Optimizing Ridesharing Services for Airport Access

Lei Feng, Elise Miller-Hooks, Paul Schonfeld, and Matthew Mohebbi

This paper addresses the problem of optimally routing and scheduling airport shuttle vehicles that offer pickup and drop-off services to customers through ridesharing. This problem, which is a version of the dial-a-ride problem, is formulated as a mixed integer program. For the solution, an exact approach applying constraint programming in a column generation framework and the adaptation of two existing heuristics are proposed. Implementations of the mathematical program and proposed solution approaches for three operational policies are presented. The performance of the proposed approaches in regard to computational requirements and solution quality was evaluated in a real-world case study involving service records for one service day out of Washington, D.C., Dulles International Airport in Chantilly, Virginia, provided by an actual airport shuttle service provider. Results show that the heuristics provide good approximations to the exact solution.

This study was motivated by the need for tools supporting efficient resource management at a shuttle service offering ridesharing services to and from two major airports in the Washington, D.C., area. In its outbound operations, the shuttle service has a fleet of vehicles that pick up customers from airport arrival doors and drop them at customer-chosen destinations. The vehicles also provide inbound services in which they pick up customers at multiple origins outside the airport and drop them at the airport's departure doors. Customers request services by phone, online, or at a kiosk in the airport or hotel. Each request includes information on the number of passengers, pickup location and time, drop-off location and time, or both. A single request can be for a one-way trip (outbound or inbound) or a round-trip (outbound and inbound). Each request results in a trip from the arrival door of the airport to the trip's destination or from the trip's origin to the departure door of the airport. Thus, requests can be made in advance or may arise dynamically on the same day of service. One or more trips are served by one vehicle through a route that is defined by a circuit traveled by a vehicle starting from and ending at the holding lot in the airport. Each vehicle may have multiple routes during a shift.

Reduced total passenger miles traveled resulting from ridesharing and efficiently designed routes can increase the profitability of the service provider and aid in diminishing traffic congestion and related negative externalities, including environmental pollution. Thus, an optimization model is proposed for the problem of determining a set of routes and schedules that meet service quality, resource, labor, and vehicle capacity constraints while minimizing total cost in relation to vehicular use and total wages in the context of airport ridesharing services. This problem, which is a version of the dial-a-ride (DAR) problem, is called the problem of airport access ridesharing (AAR) here.

The AAR problem is considered under three different operational policies, which are illustrated in Figure 1. Policy 1 handles outbound and inbound trips separately, assigning different sets of vehicles to each. This policy is under consideration because currently the demand for outbound service far outweighs the demand for inbound service. Policy 2 handles outbound and inbound trips simultaneously, permitting a single vehicle to drop off outbound and pick up inbound customers at all points along the route. Under Policy 3, all outbound trips must be dropped off before the same vehicle starts picking up inbound trips. This last policy gives preference to outbound customers, the majority of the company's actual customers. While one can show that Policy 2 will always produce the most efficient routes for the operator, other policies must be considered under certain contractual agreements or passenger service policies.

The AAR problem is a difficult combinatorial optimization problem; the number of possible solutions for it grows exponentially with increasing problem size. Obtaining even a single feasible solution by hand can be quite challenging. Yet, efficient use of limited resources is key to providing profitable, quality service that, by contract, meets service level agreement requirements. Thus, for real-world problem instances, tools to support the identification of feasible and optimal or near-optimal solutions can be crucial. An exact solution algorithm and two heuristics are proposed to solve the AAR problem. The exact solution applies a constraint programming–based column-generation (CPCG) approach (*1*). The first heuristic is a variant of the sequential insertion heuristic proposed by Jaw et al. for a related DAR problem (*2*). The second is adapted from Solomon's work on the vehicle routing problem with time windows (VRPTW) (*3*). Performance of the proposed heuristics is compared in a case study involving data from one day's operation of the Supreme Airport Shuttle fleet at one airport. The solution approaches were implemented and adapted to the three operational polices. Results from runs of the algorithms are analyzed and compared on the basis of various performance measures.

RELATED LITERATURE

The AAR problem shares several characteristics of a variety of established optimization problems. First, the AAR problem is related to the VRPTW problem (*4*). In the VRPTW problem, a vehicle must arrive at each customer within a given time window. In instances in

L. Feng, E. Miller-Hooks, and P. Schonfeld, Department of Civil and Environmental Engineering, University of Maryland, College Park, 1173 Glenn L. Martin Hall, College Park, MD 20742. M. Mohebbi, IT Curves and Supreme Airport Shuttle, Inc., 8201 Snouffer School Road, Gaithersburg, MD 20879. Corresponding author: E. Miller-Hooks, elisemh@umd.edu.

Transportation Research Record: Journal of the Transportation Research Board, No. 2467, Transportation Research Board of the National Academies, Washington, D.C., 2014, pp. 157–167.
DOI: 10.3141/2467-17

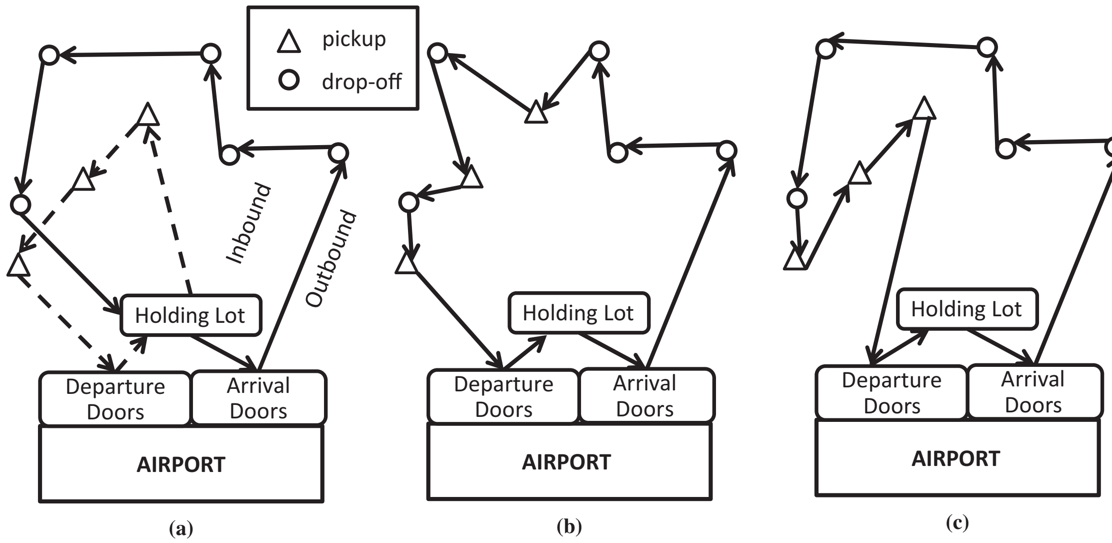

FIGURE 1 Three operational policies: (*a*) Policy 1, (*b*) Policy 2, and (*c*) Policy 3.

which soft time window constraints are permitted, a penalty for early or late arrival may be incurred. [See Dantzig and Ramser (*5*), Derochers et al. (*6*), Prescott-Gagnon et al. (*7*), Braysy and Gendreau (*8, 9*), and Dumas et al. (*10*) for comprehensive reviews of optimization algorithms for the VRPTW.] The AAR problem similarly has time windows; however, these constraints are hard. Thus, any solution that violates these constraints is infeasible. The AAR problem differs from the VRPTW problem by also including maximum ride time constraints needed to control the time spent by each passenger traveling in the vehicle, as well as maximum shift durations. Moreover, customer stops are paired in the AAR problem, since each customer has a pair of pickup and drop-off locations, and the pickup must be completed before the drop-off. This requirement creates additional precedence constraints. Such pairing and precedence of customers are captured in a generalization of the VRPTW problem, the pickup and delivery problem with time windows (PDPTW) (*10*).

As in the AAR problem, in the PDPTW problem the origin of each request must precede its destination on each vehicle tour, and both locations must be visited by the same vehicle. Among the PDPTW problems in the literature, those that address the DAR problem are most relevant (*2*). Comprehensive surveys of optimization algorithms on the PDPTW problem are provided in Wallace (*11*) and Cordeau and Laporte (*12*), and on the DAR problem in Parragh et al. (*13*) and Berbeglia et al. (*14*). The DAR problem involves passenger transportation between paired pickup and delivery points and takes user convenience into account. The AAR problem can be viewed as a special case of the DAR problem with one-to-many and many-to-one operations. See Gribkovskaia and Laporte for a discussion of this variant for a single vehicle (*15*). The AAR problem with Policy 1 or 2 can be treated as a PDPTW problem; however, Operational Policy 3 requires that inbound movements not begin until outbound movements are complete. This variant does not appear to have been addressed previously.

The vehicle routing problem with backhauls and time windows (VRPBTW) specifically captures the one-to-many and many-to-one characteristics of the AAR problem. The VRPBTW problem involves line-haul and backhaul operations. In the line-haul operations, the loading of goods onto a vehicle is completed at one or more depots, and goods are transported to one or more destinations. In backhaul operations, once line-haul operations are complete (or partially complete), goods are loaded at the line-haul destinations or other convenient locations and transported to the depot, the destination of the backhaul deliveries. A comprehensive survey of algorithms and applications for the VRPB problem, including the VRPBTW, is given in Parragh et al. (*16*). Following the classification scheme in Parragh et al., the AAR problem using Policy 1 can be viewed as two separate DAR problems, one addressing the outbound trips and the other the inbound trips (*16*). The AAR problem using Policy 2 can be defined as a vehicle routing problem with simultaneous delivery and pickup and time windows. The AAR problem using Policy 3 can be classified as a vehicle routing problem with clustered backhauls and time windows (VRPCBTW). The VRPCBTW problem does not deal with pairing and precedence of service points, a crucial characteristic of the AAR problem.

Limited works in the literature address the specifics of the VRPBTW problem and its vehicle routing problem with simultaneous backhauls and time windows (VRPSBTW) and VRPCBTW variants. The VRPBTW problem has been formulated as a mixed integer linear program, but only relatively small instances can be solved to optimality. An exact solution approach for the VRPSBTW problem is presented in Angelelli and Mansini (*17*). Specifically, a column generation framework is proposed in which the problem is decomposed into a master problem (MP) and subproblem (SP). The MP is formulated as a set covering problem, and branch-and-price is proposed for the solution of the SP. The solution of the SP supplies a feasible route for inclusion in the set of possible routes considered in the MP. The largest instance solved to optimality had 20 customers. Yano et al. formulated the VRPCBTW problem as a set partitioning problem (*18*). They proposed an exact solution method based on branch-and-bound to generate optimal tours, each with a maximum of four line-haul and four backhaul customers. For the same problem with possibly more than four customers, Gélinas et al. proposed an exact algorithm based on column generation for solving a similar set partitioning formulation of the VRPCBTW problem (*19*). The algorithm found optimal solutions for problems with up to 100 customers.

While promising, exact solution methods cannot be applied to typical problems of a size seen in real-world operations. Thus, numerous works have proposed heuristics for these problems. The majority of these heuristics include construction and improvement schemes based on classical greedy methods. More powerful methods based on metaheuristics have been proposed. For example, Dethloff proposed an extension of the cheapest insertion heuristic for the VRPSBTW problem (*20*). Thangiah et al. proposed a heuristic for solution of the VRPCBTW problem (*21*). In the construction phase, the insertion procedure of Kontoravdis and Bard proposed for the VRPTW problem is used to obtain initial solutions (*22*). Then, the initial solutions are improved through the application of λ-interchanges and 2-opt* exchanges in the improvement phase. Duhamel et al. (*23*) used an insertion procedure proposed in Solomon (*3*) for initial solution construction, but proposed a tabu search heuristic for the improvement phase. Zhong and Cole presented an augmented objective function for the VRPCBTW problem, in which violations of time windows and capacity and line-haul-backhaul precedence constraints are penalized (*24*). The cluster-first, route-second method is used for initial route construction, and intra- and interroute operators are described for use in improving the tours. Metaheuristics, such as tabu search (*23*), genetic algorithms (*25*), and ant colony optimization (*26*), have also been proposed.

In the next section, a general formulation is presented that can be used to solve all three variants of the AAR problem. This formulation combines aspects of previously proposed formulations for the PDPTW and VRPBTW problems. It fills the need for a formulation of a PDPTW problem with line-haul and backhaul operations or the VRPCBTW problem with additional pairing and precedence constraints, that is, an AAR problem with Policy 3. The proposed formulation does not rely on a complete enumeration of the feasible tours, which is a requirement of the set partitioning formulation of the VRPCBTW problem and set covering approach for the VRPSBTW problem. The formulation given next includes additional constraints specific to an application involving passengers as opposed to cargo, including maximum passenger ride times and restrictions on idling with passengers onboard.

PROBLEM FORMULATION

In this section, the AAR problem is formulated. The formulation is preceded by the introduction of notation. Additional adaptations needed for different operational policy implementations are given.

Notation

n_O = number of outbound trips and requests;
n_I = number of inbound trips and requests;
P_O = set of outbound pickup nodes located at arrival door, $P_O = \{1, \dots, n_O\}$;
P_I = set of inbound pickup nodes, $P_I = \{n_O + 1, \dots, n_O + n_I\}$;
D_O = set of outbound drop-off nodes, $D_O = \{n_O + n_I + 1, \dots, 2n_O + n_I\}$;
D_I = set of inbound drop-off nodes located at departure door; $D_I = \{2n_O + n_I + 1, \dots, 2n_O + n_I\}$
P = set of all pickup nodes, $P = P_O \cup P_I$;
D = set of all drop-off nodes, $D = D_O \cup D_I$;
V = set of available vehicles;
q_i = demand and supply at node i; for pickup nodes, $q_i > 0$, $\forall i \in P$; for drop-off nodes, $q_i < 0$, $\forall i \in D$; for the holding lot, $q_0 = q_{2n_O+2n_I+1} = 0$;
e_i = earliest service time at node i, that is, start of time window;
l_i = latest service time at node i, that is, end of time window;
s_i = service duration or dwell at node i;
c_{ij}^v = cost to travel from node i to node j with vehicle v;
t_{ij}^v = travel time from node i to node j with vehicle v;
Q^v = capacity of vehicle v;
T^v = shift duration of vehicle and route v;
R_i = maximum ride time of request i; and
M = arbitrary large number.

Decision Variables

$$x_{ij}^v = \begin{cases} 1 & \text{if arc}(i, j) \text{ is traversed by vehicle } v \\ 0 & \text{otherwise} \end{cases}$$

L_i^v = load of vehicle v leaving node i,
A_i^v = arrival time of vehicle v at node i,
B_i^v = time of beginning service of vehicle v at node i, and
y = auxiliary binary variable.

With this notation, the AAR problem can be modeled on a digraph $G = (N, A)$, where N is the set of all nodes, $N = P \cup D \cup \{0, 2n_O + 2n_I + 1\}$ and A is the set of directed arcs, $A = \{(i, j): i, j \in N, i \neq 2n_O + 2n_I + 1, j \neq 0, i \neq j\}$.

General AAR Problem Formulation

The general formulation of the AAR problem builds on the existing formulations for the VRPBTW problem (*16*) and PDPTW problem (*27*).

$$\min \sum_{v \in V} \sum_{(i,j) \in A} c_{ij}^v \cdot x_{ij}^v + \sum_{v \in V} \sum_{j \in P} c_w \cdot \left(B_j^v - A_j^v\right) \tag{1}$$

subject to

$$\sum_{v \in V} \sum_{j:(i,j) \in A} x_{ij}^v = 1 \qquad \forall i \in P \tag{2}$$

$$\sum_{j:(i,j) \in A} x_{ij}^v - \sum_{j:(n_O+n_I+i,j) \in A} x_{n_O+n_I+i,j}^v = 0 \qquad \forall i \in P; v \in V \tag{3}$$

$$\sum_{j:(0,j) \in A} x_{0j}^v = 1 \qquad \forall v \in V \tag{4}$$

$$\sum_{j:(i,j) \in A} x_{ji}^v - \sum_{j:(i,j) \in A} x_{ij}^v = 0 \qquad \forall i \in P \cup D; v \in V \tag{5}$$

$$\sum_{i:(i,2n_O+2n_I+1) \in A} x_{i,2n_O+2n_I+1}^v = 1 \qquad \forall v \in V \tag{6}$$

$$-M\left(1 - x_{ij}^v\right) \leq B_j^v - B_i^v - s_i - t_{ij}^v \leq M\left(1 - x_{ij}^v\right) \qquad \forall (i, j) \in A; v \in V \tag{7}$$

$$-M\left(1 - x_{ij}^v\right) \leq A_j^v - A_i^v - t_{ij}^v \leq M\left(1 - x_{ij}^v\right) \qquad \forall (i, j) \in A; v \in V \tag{8}$$

$$L_0^v = L_{2n_O+2n_I+1}^v = 0 \qquad \forall v \in V \tag{9}$$

$$-M\left(1 - x_{ij}^v\right) \leq L_j^v - L_i^v - q_j \leq M\left(1 - x_{ij}^v\right) \qquad \forall (i, j) \in A; v \in V \tag{10}$$

$$e_i - A_i^v < M(1-y) \qquad \forall i \in N; v \in V \tag{11}$$

$$-(L_i^v - q_i) \le M \cdot y \qquad \forall i \in N; v \in V \tag{12}$$

$$\max(0, q_i) \le L_i^v \le \min(Q^v, Q^v + q_i) \qquad \forall i \in N; v \in V \tag{13}$$

$$\max(e_i, A_i^v) \le B_i^v \le l_i \qquad \forall i \in N; v \in V \tag{14}$$

$$B_i^v + t_{i,n_O+n_I+i}^v \le B_{n_O+n_I+i}^v \qquad \forall i \in P; v \in V \tag{15}$$

$$B_{2n_O+2n_I+1}^v - B_0^v \le T^v \qquad \forall v \in V \tag{16}$$

$$B_{n_O+n_I+i}^v - (B_i^v + s_i) \le R_i \qquad \forall i \in P; v \in V \tag{17}$$

$$x_{ij}^v \in \{0,1\} \qquad \forall (i,j) \in A; v \in V \tag{18}$$

The objective function (Equation 1) minimizes total routing cost while penalizing waiting times. The variable C_w is the unit cost of vehicle waiting. The cost c_{ij}^v in the function is expressed in Equation 19, which includes costs related to vehicle travel distance and time.

$$c_{ij}^v = C_d * d_{ij} + C_t * t_{ij}^v \tag{19}$$

where C_d and C_t are unit costs of vehicle travel distance and travel time, respectively, and d_{ij} is the distance between nodes i and j.

Constraints 2 and 3 ensure that every node is visited exactly once and pickup and drop-off nodes associated with a particular request are visited by the same vehicle, respectively. Each route starts and ends at a holding lot as required in Constraints 4 and 6, respectively. Constraints (Equation 5) enforce flow conservation. Constraints 7 and 8 and 9 and 10 guarantee consistency between time and load variables, respectively. Constraints 11 and 12 ensure that a vehicle does not idle while carrying passengers. Capacity and time window constraints are imposed by Inequalities 13 and 14, respectively. Constraints (Equation 15) force the pickup node to be visited before the drop-off node, for each request. The maximum route duration is restricted in Constraints 16. The passenger maximum ride time constraints are specified in Inequalities 17, followed by integrality constraints expressed by Constraints 18. Constraints 2, 4 through 6, 13, and 15 are used in formulations of Parragh et al. (*16*) and Ropke and Cordeau (*27*). Constraints 7, 9, and 15 are included in the formulation given in Parragh et al. (*16*), while Constraints 3 and 14 are included in the formulation of Ropke and Cordeau (*27*). Constraints 8, 10 through 12, 16, and 17 are unique to the AAR problem. If all vehicles are identical, superscript v in c_{ij}^v, t_{ij}^v, Q^v, and T^v can be eliminated for the AAR problem formulation.

The formulation is designed to be general and directly applicable for Policy 2. Small adaptations are required for the application of Policies 1 and 3, as described next.

Adaptation for Policy 1

To apply the formulation in which Policy 1 is implemented, the problem can be posed as two separate DAR problems, one for outbound trips and the other for inbound trips. To specify the DAR problem for outbound trips, the AAR problem formulation can be applied by presetting certain variables. Specifically, $n_I = 0$, $P_I = D_I = \phi$ for the outbound problem and $n_O = 0$, $P_O = D_O = \phi$ for the inbound problem.

Adaptation for Policy 3

For the AAR problem under Policy 3, additional constraints (Equation 20) are required to ensure that each vehicle drops off its outbound passengers before picking up its inbound passengers. This requirement can be implemented by restricting arcs between inbound and outbound customer location nodes. That is, no arc can exist in a route directly connecting any inbound pickup location to an outbound drop-off location. This restriction precludes any tour from allowing a sequence in which an inbound request is served before all outbound drop-offs are completed.

$$x_{ij}^v = 0 \qquad \forall i \in P_I; j \in P_O \cup D_O; v \in V \tag{20}$$

The AAR problem is difficult to solve directly since the number of decision variables increases exponentially with increasing problem size (number of nodes to be visited). The proposed formulation was implemented directly in the IBM ILOG CPLEX package on a personal computer with Intel(R) CPU 3.10 GHz and 4.0 GB RAM. The required computation time was exceptionally long. In a reduced version of the problem instance with only 10 outbound trips and 10 available vehicles, the solution was obtained after more than 6 h, which is unacceptable in practice. Thus, in the next section, an alternative exact solution method is proposed.

EXACT SOLUTION METHOD

A CPCG solution methodology is proposed for the exact solution of the AAR problem. A column generation mechanism is used in which the AAR problem is decomposed into an MP and an SP. A restricted linear relaxation of the MP (LMP) is solved for a given route set. Optimal dual variables associated with the requests served are determined in solving the SP. The dual variables are used to calculate the reduced costs associated with the tours (columns). At each iteration, a tour with negative reduced cost is added to the route set of the MP. Solution of the SP is obtained through a constraint programming methodology. An overview of the proposed CPCG methodology is provided in Figure 2.

The procedure starts by feeding the LMP, modeled as a set covering problem, with a feasible solution consisting of a set of vehicles, each of which serves one request. The SP is a constrained shortest path problem. Solution of the SP produces a route with negative reduced cost. This route is added to the route set (or column set) used in the next iteration in which the solution of the LMP is repeated. This process terminates when the solution of the SP does not produce a route that is not already included in the column pool with negative reduced cost. With the final column pool (route set), the integer MP, a set partitioning problem, is solved. The route obtained from the solution of the integer MP is reassembled through a proposed heuristic procedure to generate the final solution. Details associated with the MP, SP, and heuristic reassemble procedure are provided next.

Master Problem

Assume that all vehicles are identical, and let Ω denote the set of feasible routes satisfying Constraints 3 through 17. For each route, $r \in \Omega$

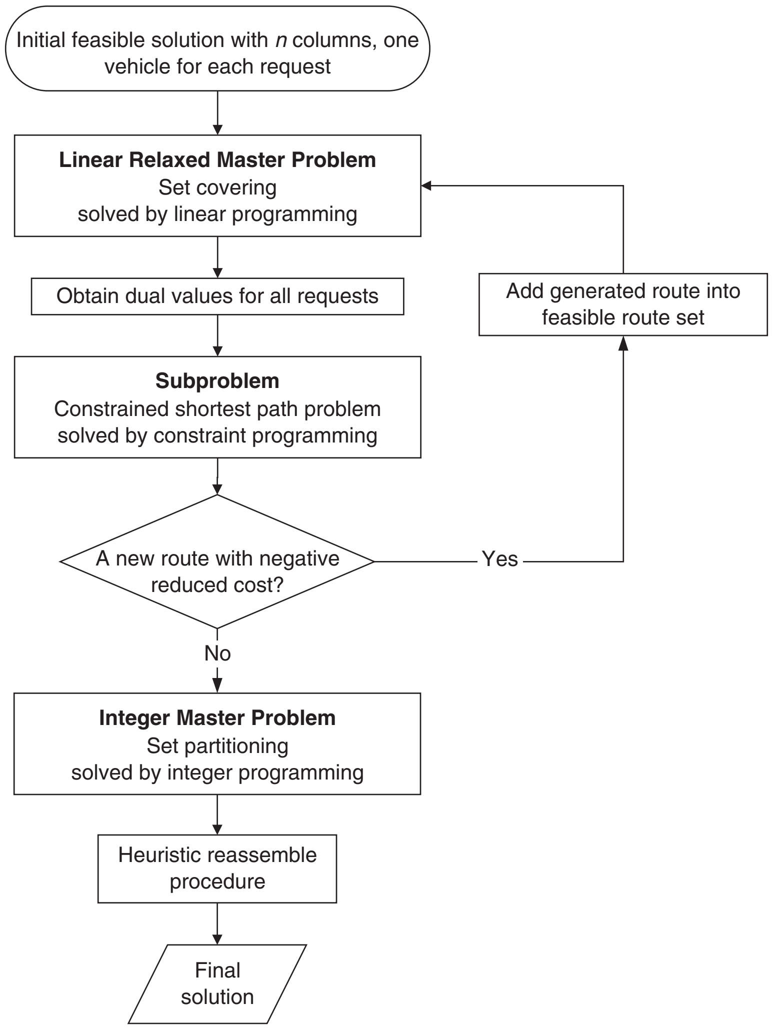
Initial feasible solution with *n* columns, one vehicle for each request
Linear Relaxed Master Problem
Set covering
solved by linear programming
Obtain dual values for all requests
Add generated route into feasible route set
Subproblem
Constrained shortest path problem
solved by constraint programming
A new route with negative reduced cost?
Yes
No
Integer Master Problem
Set partitioning
solved by integer programming
Heuristic reassemble procedure
Final solution

FIGURE 2 Flowchart of exact solution method.

let c_r be the cost of the route and a_{ir} be a binary constant indicating whether or not a node $i \in P$ is visited by route r. Let y_r be a binary variable equal to 1 if route $r \in \Omega$ is selected, and 0 otherwise. The AAR problem can be reformulated as the following set partitioning problem (MP-SPP):

$$\min \sum_{r \in \Omega} c_r \cdot y_r \tag{21}$$

subject to

$$\sum_{r \in \Omega} a_{ir} y_r = 1 \qquad \forall i \in P \tag{22}$$

$$y_r \in \{0, 1\} \qquad \forall r \in \Omega \tag{23}$$

The objective (Equation 21) minimizes the cost of the chosen routes. Constraints (Equation 22) ensure that every request is served once.

It is impractical to enumerate all feasible routes in Ω explicitly. Instead, as is typical, only a subset, $\Omega' \subset \Omega$, is considered. This subset is expanded iteratively by adding a route with negative reduced cost through solution of the SP. The reduced cost of a route is expressed by Equation 24.

$$\hat{c}_{ij} = \begin{cases} c_{ij} - \pi_i & \forall i \in P; (i, j) \in A \\ c_{ij} & \forall i \in N \backslash P; (i, j) \in A \end{cases} \tag{24}$$

where π_i is the dual value associated with the ith constraint (Equation 22). The LMP is given next:

$$\min \sum_{r \in \Omega'} c_r \cdot y_r \tag{25}$$

subject to

$$\sum_{r \in \Omega'} a_{ir} y_r \geq 1 \qquad \forall i \in P \tag{26}$$

$$y_r \geq 0 \qquad \forall r \in \Omega' \tag{27}$$

This relaxation allows every request to be served more than once rather than only once. Constraints (Equation 27) relax integrality constraints.

Subproblem

The SP formulation is

$$\min \sum_{(i,j)\in A} rc_{ij} \cdot x_{ij} + \sum_{v\in V}\sum_{j\in P} C_w \cdot \left(B_j - A_j\right) \qquad (28)$$

subject to Constraints 3 through 17.

Constraint Programming Model for SP

In the constraint programming approach, each decision variable has a domain. For example, in the SP, the domain of each arc, x_{ij}, is $\{0, 1\}$. Similarly, the domain of load L_i is $\{0, 1, \dots, Q\}$. Initially, the search space contains all combinations of the values in the domains of all decision variables. To avoid exploring the entire search space, the constraint programming approach first removes inconsistent values from the domains of the variables involved in each constraint. Then a search strategy (depth first, width first, or multistart) is applied to guide the search for a solution in the reduced search space. The search process can be viewed as traversing a tree, in which the root is the starting point, a leaf node is a combination of values in the reduced search space, and each branch represents a move (branching) in the search. A solution is a set of value assignments to the decision variables such that each variable is assigned to exactly one value from its domain. Together these values satisfy all constraints and minimize the objective function. Each leaf node is evaluated to determine whether it will produce a feasible solution.

Two measures are suggested for speeding up the process of finding a feasible solution: (*a*) eliminate ineligible decision variable settings from the initial search space, wherein those decisions that include starting from the end depot, ending at the starting depot, having self-loops, or violating Constraints 13 through 15 are excluded, and (*b*) set branching limits for the route generation process. As mentioned in Irnich and Desaulniers, in the context of column generation, optimality of the SP is necessary only to prove that no negative reduced cost routes exist in the last iteration and that feasible solutions to the SP are sufficient for preceding iterations (*28*). Thus, a lower branching limit (10^6) is used for these nonfinal iterations, while higher branching limits (10^8) are applied in the last iteration.

Heuristic Reassembly Procedure

The optimal solution to the LMP is obtained when there are no remaining routes with negative reduced cost to the SP. Unfortunately, this solution is not always integer valued. A branching scheme was proposed in Dumas et al. to address this issue through adding arc flow constraints to the SP and resolving it (*10*). This process is repeated for each branching decision taken in the MP. The following observations are made:

1. The MP starts with a feasible solution in which one vehicle serves one request.
2. Each iteration generates a single unique route.
3. The newly generated routes that are selected by solution of the MP-SPP are always a subset of the newly generated routes that are selected by solution of the set covering problem.
4. The solution to the MP-SPP always includes one or more initial feasible routes.

Since solution from the MP-SPP provides useful information, the following heuristic applies:

Step 1. Calculate the value of V = route cost and number of requests served for each route selected by the MP-SPP.
Step 2. Select the route r with maximum V. Try to extract the first unvisited request on route r, and insert it into the best feasible position on one of the other routes, r'. If this procedure decreases the total cost, update r and r', and go to Step 1. Otherwise, mark this request as visited, and move to the next request in r. If all requests in r have been visited, stop.
Step 3. Repeat Step 1.

HEURISTIC SOLUTION APPROACHES

Two heuristics proposed in the literature were modified for the solution of the AAR problem. An overview of each is given first, followed by the modifications required to address the three variants of the AAR problem.

Jaw's Heuristic

The first heuristic considered for solution to the AAR problem is the sequential insertion procedure originally proposed by Jaw for the DAR problem in Jaw et al. (*2*). The algorithm processes each request in an unrouted request list (URL) in sequence and assigns each request to a vehicle until the URL is exhausted. The main steps of Jaw's sequential insertion procedure are summarized as follows:

Step 1. Sort the URL by the requested pickup times in increasing order. Create a route from and back to the depot. Set $r = 1$.
Step 2. Select the first unrouted request u from the URL. Find all feasible insertion positions in all existing routes, 1 to r.
 i. If a feasible insertion position is found, assign the request u to the route r^* with minimum insertion cost, and update route r^*.
 ii. If no feasible insertion position exists, create a new route from the depot to request u, and add a return to the depot. Set $r = r + 1$.
 Delete u from URL.
Step 3. Repeat Step 2 until the URL is empty.

The additional insertion cost to route r of inserting request u is calculated as the difference between the total cost of route r after the insertion minus its cost before the insertion. This calculation is expressed in Equation 29.

$$\sum_{i,j\in \text{new}^r} c_{ij}^r - \sum_{i,j\in \text{old}^r} c_{ij}^r \qquad (29)$$

where new^r denotes route r after insertion of request u and old^r denotes route r before insertion of request u.

Solomon's Insertion I1 Heuristic

The second heuristic considered here, Insertion I1 (I1), was proposed by Solomon for the VRPTW (*3*). The I1 heuristic constructs routes one at a time. For the first created route, a tour is developed from the depot to a "seed" request, which returns to the depot. Remaining requests are considered for insertion in the route. The cost of insertion of all remaining unrouted requests is computed. The request with the minimum insertion cost that can feasibly be inserted is selected. Insertion of additional requests is considered until no remaining unrouted request can be feasibly inserted. A new route is then created. The process is repeated until all requests have been included in a tour. At each iteration in which a new route is created, the remaining unrouted request with the minimum value of $-\alpha d_{0i} + \beta l_i$, $0 \le \alpha \le 1$, $i \in D_O$ for outbound trips and $i \in P_I$ for inbound trips is selected as the seed. Trips that are far from the depot and have an earlier deadline are, thus, favored in choosing the request.

The main steps of I1 can now be summarized:

Step 1. Initialize $r = 0$.

Step 2. Set $r = r + 1$. Select the seed request u^* with the minimum value of $-\alpha d_{0i} + \beta l_i$ from the URL for inclusion in route *r.* Add u^* to route *r,* and delete it from the URL. If the URL is empty, stop.

Step 3. For each remaining unrouted request *u* in the URL, find the feasible insertion position in route *r,* if a feasible insertion exists, that minimizes the additional insertion cost (Equation 30).

(i) If a feasible insertion exists, select request u^* with the minimum additional insertion cost (Equation 30), and insert this request at its best feasible insertion position in route *r.* Update route *r,* and delete u^* from the URL.

(ii) If there is no feasible insertion of any unrouted request in route *r,* go to Step 2.

Step 4. Repeat Step 3 until the URL is empty.

The two heuristics are quite similar, but differ in one fundamental aspect relating to the choice of a feasible insertion position for the unrouted requests. In Jaw's heuristic, for each selected unrouted request *u,* its best insertion position within all constructed routes is evaluated and the insertion is made accordingly. When a request cannot be feasibly inserted in any existing route, a new route is constructed. The request is inserted in the new route. The next unrouted request from a list that was not yet tested will be considered for inclusion in this expanded set of constructed routes. In heuristic I1 this evaluation is conducted only over the most recently constructed route. The list of unrouted requests must be considered, and any possible insertions must be made in that route before insertion in another route is considered.

Both heuristics as described can be used directly on the AAR problem with Policies 1 and 2. For Policy 3, however, feasibility is further restricted by outbound and inbound trip separation requirements. Both heuristics can be adapted to deal with this additional constraint. Specifically, modifications are made when the best feasible insertion position is chosen for each unrouted request: if the selected unrouted request *u* is an outbound trip, its drop-off location must be inserted before the pickup of the first inbound trip, given that there are inbound trips in the current route. Likewise, if the unrouted request *u* is an inbound trip, its pickup location must be inserted after the drop-off location of the last outbound trip, assuming there is an outbound trip in the current route.

Checking Solution Feasibility

Both heuristics ensure that the problem constraints associated with time windows, precedence and pairing, maximum ride time, shift duration for drivers, and vehicle capacities are satisfied during the insertion process. An insertion of a request in a route is feasible only if it does not lead to the violation of any of these constraints by the inclusion of this request. Moreover, its inclusion should not create other violations of these constraints for other requests already included in the route. The implementation of these constraints during this process is important and is described next.

Time Window Constraints

Time window feasibility is maintained in a route if the insertion of a new request does not push the vehicle arrival time at any node *i* past its latest service time l_i. While a vehicle without a passenger onboard is permitted to arrive at a pickup node earlier than its earliest service time e_i, thus incurring an additional waiting cost, no vehicle is permitted to idle while carrying passengers. The procedure proposed in Jaw et al. is applied for the calculations of the earliest service time, e_i, and latest service time, l_i (*2*).

To ensure that time window constraints are met, one must check that e_i and l_i fall within each request's time window for each *i* in the route and for requests considered for inclusion.

Precedence and Pairing Constraints

For any insertion of a new request, precedence and pairing constraints are ensured by simultaneously inserting the pickup and drop-off locations associated with a single request in the route. The pickup location must precede the drop-off location.

Maximum Ride Time Constraints

For each considered insertion of a request, the insertion must not cause a violation in constraints (Equation 17), whether directly for the request or for other requests already inserted in the route. The maximum ride time R_i is a function of direct (shortest path) ride time DRT_i. Here, a piecewise linear function is applied (*30*):

$$R_i = \begin{cases} 3 \cdot \mathrm{DRT}_i & \text{if } \mathrm{DRT}_i \le 30 \\ 2 \cdot \mathrm{DRT}_i & \text{if } 30 \le \mathrm{DRT}_i \le 60 \\ \mathrm{DRT}_i + 30 & \text{if } \mathrm{DRT}_i > 60 \end{cases} \tag{30}$$

Shift Duration Limit for Drivers Any insertion of a new request cannot extend the route duration over the shift duration limit for a driver, as expressed by constraints (Equation 16). Thus, shift duration must be assessed for each insertion of a request.

Vehicle Capacity Constraints Any insertion of a new request must adhere to capacity constraints (Equation 13). Thus, no insertion is made if its inclusion will cause the vehicle to exceed its capacity. This situation must be assessed at each potential insertion location because the number of requests handled at any point in time changes over the route duration.

NUMERICAL EXPERIMENTS

Experiment Design

To investigate the efficiency of the proposed solution approach, the solution methods were tested on a real-world problem instance. The test case involved service records for one service day in January 2012 out of Washington, D.C., Dulles International Airport (IAD). The case

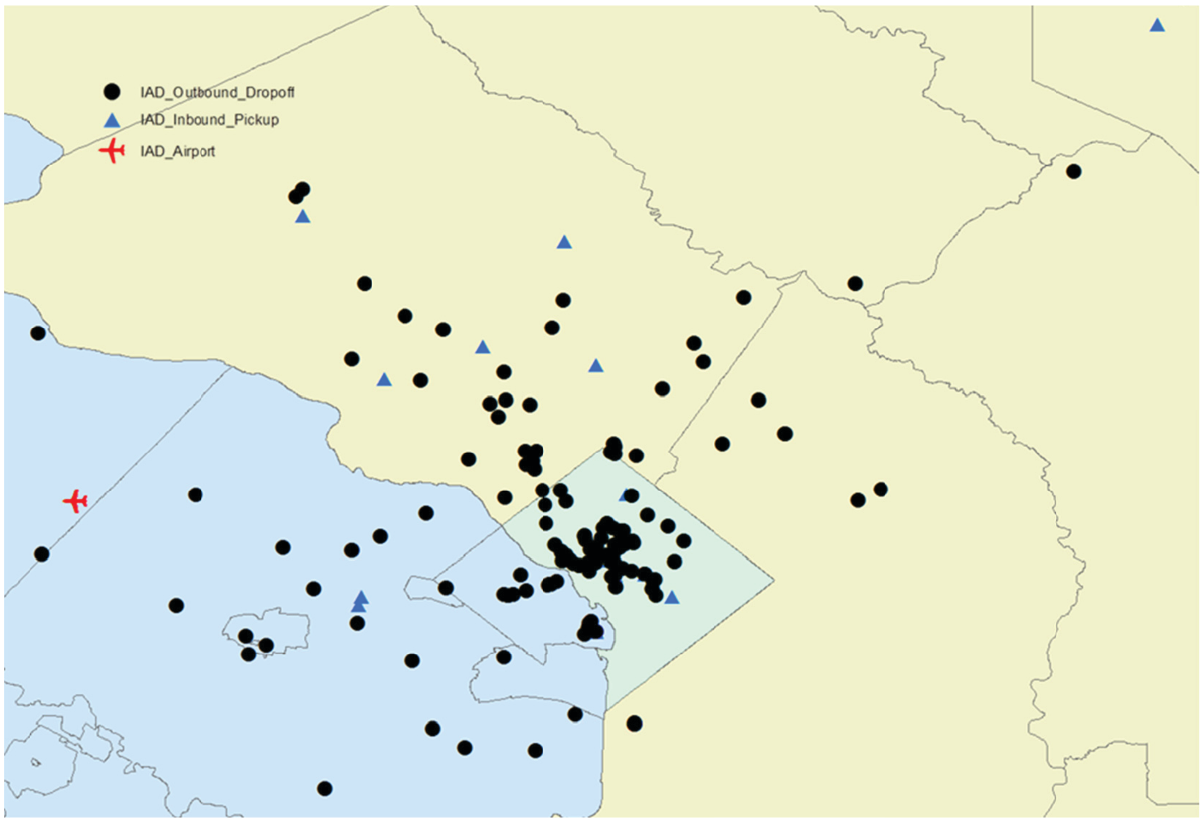

FIGURE 3 Distributions of pickup and drop-off locations.

included 164 outbound requests involving 212 passengers and 22 inbound requests involving 41 passengers. For each request, detailed information, including desired pickup time, number of passengers, latitude and longitude of pickup and drop-off locations, and assigned vehicle index, was also included. All requests were served by a fleet of identical vehicles. Figure 3 shows the partial distributions of the requested pickup (inbound) and drop-off (outbound) locations. The service area covers Maryland, the District of Columbia, Virginia, and Pennsylvania. Distances and travel times between pairs of customer locations were calculated through the OD cost matrix tool in the Network Analyst Toolbox of ArcGIS. The travel time is computed on the basis of the shortest distance and speed limits.

Parameters of the model and the algorithms are presented in Table 1. The solution methods were implemented in Visual C++ 2010 and run on a personal computer with Intel(R) CPU with 3.10 GHz and 4.0 GB RAM. The SP associated with the exact solution method was solved by the C++ Concert Technology of Constraint Programming solver in the IBM ILOG CPLEX.

Algorithm Performance

The CPCG approach was tested on cases with 10, 20, and 30 requests under the most general policy, Policy 2. The computation time increases exponentially with an increasing number of customers. Thus, solution of problem instances with significantly more than 30 customers is precluded. The results are compared with those obtained through the adapted Jaw's algorithm in Table 2. The numbers in parentheses are outbound and inbound requests, respectively. Results show that the maximum gap between the exact solution and the adapted Jaw's algorithm is approximately 7% (with 20 requests), but the computation time is about 1/1200 of that of the CPCG approach.

Policy Performance

Computation results obtained by applying the two heuristics for each of the three operational policies are shown in Table 3. From Table 3,

TABLE 1 Parameters of Proposed Model and Algorithms

Parameter	Explanation	Value
C_t	Unit cost of time	\$0.54/min
C_d	Unit cost of distance	\$0.72/mi
C_w	Unit cost of vehicle waiting time	\$0.23/min
Q	Vehicle capacity	7 passengers
V	Maximum fleet size	30
s	Identical service time	3 min
T	Shift duration	10 h
α, β	Weight parameters	$\alpha = 0.8$, $\beta = 0.2$
TW	Prespecified maximum deviation from desired time	45 min

TABLE 2 Comparison of Results from the CPCG Approach and Adapted Jaw's Algorithm

	CPCG Approach by Number of Requests			Adapted Jaw's Algorithm by Number of Requests		
Performance	10 (7 + 3)	20 (14 + 6)	30 (22 + 8)	10 (7 + 3)	20 (14 + 6)	30 (22 + 8)
Total cost ($)	314.9	581.1	839.6	314.9	625.0	853.4
Number of vehicles used	2	2	4	2	3	4
Computational time (s)	360.5	2040.2	8940.3	0.7	3.5	7.6

two significant conclusions can be reached. First, Jaw's heuristic outperforms I1. For all three policies, the computation time required by Jaw's heuristic is only 14% to 25% of that required by I1. The longer I1 computation time can be explained by the requirement for assessing the insertion of every unrouted request when each route is constructed. For each of the three operational policies, the total cost of the routes built through Jaw's heuristic is below that developed by I1. This finding may be the result of the seed selection process of I1, in which the farthest unrouted request is selected for inclusion. The long distance to this request may lead to longer empty vehicle miles and, thus, increased route duration and total cost. Moreover, for each of the three operational policies, the routes built through Jaw's heuristic have higher utility factors (higher average occupancy, lower passenger miles, and higher average utilization) than those from I1.

Second, both heuristics reveal that Policy 2 provides the best performance in relation to the number of needed vehicles, idle time, empty and loaded driving time or miles traveled, and total cost. Policy 3 provides the second-best performance, and Policy 1 the worst performance. That Policy 2 provides the best performance is unsurprising and can be shown theoretically, because it is the least constrained of the three variants. From run results of Jaw's heuristic, Policy 2 requires the fewest vehicles, lowest idle time, lowest empty vehicle miles, and lowest total cost of the three policies. Accordingly, Policy 2 has the highest vehicle utilization rate of the three. The vehicle utilization rate of Policy 3 is significantly above that

TABLE 3 Performance Comparisons of Two Heuristics

	Policy 1 Performance by Heuristic						Policy 2 Performance		Policy 3 Performance	
	Jaw's Heuristic			Solomon's Heuristic						
Performance Measure	Outbound	Inbound	Total	Outbound	Inbound	Total	Jaw's Heuristic	Solomon's Heuristic	Jaw's Heuristic	Solomon's Heuristic
Number of vehicles	17	4	21	17	4	21	17	19	20	21
Total idle time[a] (min)	1,051	159	1,211	856	255	1,110	929	1,430	1,201	1,603
Total DH1Time[b] (min)	0	215	215	0	350	350	52	18	151	49
Total DH1Mile[c] (mi)	0	200	200	0	327	327	49	17	141	46
Total DH2Time[d] (min)	933	0	933	1,151	0	1,151	562	1,029	723	1,161
Total DH2Mile[e] (mi)	871	0	871	1,074	0	1,074	524	961	674	1,083
Total EDTime[f] (min)	1,696	516	2,213	2,223	548	2,771	1,537	1,802	1,861	2,141
Total EDMile[g] (mi)	1,583	482	2,065	2,075	511	2,586	1,434	1,682	1,737	1,998
Total LDTime[h] (min)	3,082	586	3,668	2,798	590	3,388	3,669	3,424	3,690	3,599
Total LDMile[i] (mi)	2,877	547	3,424	2,611	550	3,162	3,424	3,195	3,444	3,359
Route duration (min)	6,883	1,414	8,298	6,947	1,557	8,505	7,319	7,877	7,953	8,615
Average occupancy (number of passengers)	1.5	1.1	1.3	1.2	0.8	1.0	1.6	1.3	1.4	1.1
Average passenger miles	25.9	25.6	25.7	26.9	30.5	28.7	24.7	25.7	25.3	25.8
Average utilization[j]	7.6	6.1	7.3	6.9	6.1	6.7	9.0	7.5	7.7	7.1
Total cost ($)	6,034	1,372	7,406	6,282	1,437	7,719	6,523	6,662	7,005	7,325
CPU Time (s)		27.1			173.1		37.4	274.8	28.6	116.0

[a]Sum of all waiting times incurred by vehicle along its route.
[b]Empty driving time from depot to first pickup.
[c]Empty driving distance from depot to first pickup.
[d]Empty driving time from last dropoff to ending depot.
[e]Empty driving distance from last dropoff to ending depot.
[f]Driving time without passengers on board.
[g]Driving distance without passengers on board.
[h]Driving time with one or more passengers on board.
[i]Driving distance with one or more passengers on board.
[j]Total LDTime/(24 $*$ Number of Vehicles).

of Policy 1 but below that of Policy 2. This difference in vehicle utilization rate is caused by requirements for ordering outbound and inbound operations with Policies 1 and 3.

To assess the value of this optimization-based approach, solutions obtained from the heuristics were compared against manually derived routes used to deploy the vehicle fleet on the date of the case study. In actual operations on the date of service, Policy 2 was used. From records maintained for that date, 37 vehicles were used. Stringing the vehicle routes together where feasible would permit completion by as few as 28 vehicles. Many of the routes did not comply with maximum ride time constraints, and several violated constraints that prohibit waiting with a passenger onboard. Of course, violations were addressed during actual operations. In comparison, results from the proposed heuristic for the AAR problem under Policy 2 required only 17 vehicles to serve the same requests. This result is an approximately 60% improvement in vehicle utilization.

CONCLUSIONS AND FUTURE WORK

The AAR problem is formulated here as a mixed integer program. Three implementations corresponding to three different operational policies under consideration in practice are investigated. Exact and heuristic solution procedures are proposed. The performance of the proposed solution approaches is compared in a case study involving data from one day's operation of an actual service provider. No exact solution could be obtained for the full version of the case study, but an exact solution was obtained for a reduced version with 30 customer requests. In a comparison the results of the adapted Jaw's algorithm were within 7% of the exact solution and required only 1/1,200 the computation time. In the original case study, the adapted Jaw's algorithm outperformed the second proposed heuristic. Thus, the heuristic is an effective and efficient approach for addressing the AAR problem, yielding significantly better results than routes and schedules determined manually.

The proposed methodologies are also relevant for other routing and scheduling applications involving one-to-many-to-one operations, such as passenger feeder problems involving subscription services, school bus routing, carpooling, and express collection and delivery. Other possible applications arise in reverse logistics operations, including the delivery of full and collection of empty bottles between a manufacturer and retailers. However, the reverse logistics problem is simpler than the airport shuttle, because the goods to be transported are identical. Thus, every unit to be picked up can equally satisfy customer demand.

While the heuristics described here provide good results with low computational effort, more sophisticated heuristics may provide improved solutions. Both described heuristics are construction heuristics. Thus, constructed routes can be improved through the application of improvement operators, such as λ-interchange, 2-opt* exchange, trip exchange, and trip reinsertion. A cluster-first, route-second methodology may also address this myopic nature. Clustering can be based on temporal and spatial characteristics of the pickup and drop-off locations. The authors are currently investigating these and other improvements.

In a dynamic setting, new requests may be received on short notice while some vehicles are en route. The operator must quickly insert these new requests into previously constructed routes and schedules. In the airport operations of the case study, most inbound requests are known in advance but almost all outbound trips arise dynamically. A fast algorithm to find a good feasible insertion for the new requests is required. The authors are working to extend the developed model and solution methodologies to operations with uncertainties in travel and service times, as well as dynamic requests.

ACKNOWLEDGMENTS

This work was funded by the Maryland Industrial Partnership and IT Curves.

REFERENCES

1. Junker, U., S.E. Karisch, N. Kohl, B. Vaaben, T. Fahle, and M. Sellmann. A Framework for Constraint Programming Based Column Generation. *Proc., 5th International Conference on Principles and Practice of Constraint Programming,* Springer, 1999.
2. Jaw, J.J., A.R. Odoni, H.N. Psaraftis, and N.H.M. Wilson. A Heuristic Algorithm for the Multivehicle Advance Request Dial-a-Ride Problem with Time Windows. *Transportation Research Part B: Methodological,* Vol. 20, No. 3, 1986, pp. 243–257.
3. Solomon, M.M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research,* Vol. 35, No. 2, 1987, pp. 254–265.
4. Kolen, A.W., A.R. Kan, and H. Trienekens. Vehicle Routing with Time Windows. *Operations Research,* Vol. 35, No. 2, 1987, pp. 266–273.
5. Dantzig, G.B., and J.H. Ramser. The Truck Dispatching Problem. *Management Science,* Vol. 6, No. 1, 1959, pp. 80–91.
6. Desrochers, M., J. Desrosiers, and M. Solomon. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research,* Vol. 40, No. 2, 1992, pp. 342–354.
7. Prescott-Gagnon, E., G. Desaulniers, and L.M. Rousseau. A Branch-and-Price-Based Large Neighborhood Search Algorithm for the Vehicle Routing Problem with Time Windows. *Networks,* Vol. 54, No. 4, 2009, pp. 190–204.
8. Braysy, I., and M. Gendreau. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science,* Vol. 39, No. 1, 2005, pp. 104–118.
9. Braysy, I., and M. Gendreau. Vehicle Routing Problem with Time Windows, Part Ii: Metaheuristics. *Transportation Science,* Vol. 39, No. 1, 2005, pp. 119–139.
10. Dumas, Y., J. Desrosiers, and F. Soumis. The Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research,* Vol. 54, No. 1, 1991, pp. 7–22.
11. Wallace, N.E. *Optimization in Dial-a-Ride System Analysis: A Comparison of Recent Modelling and an Expected Value Model.* Highway Research Institute, University of Michigan, Ann Arbor, 1978.
12. 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.
13. Parragh, S., K. Doerner, and R. Hartl. A Survey on Pickup and Delivery Problems Part Ii: Transportation Between Pickup and Delivery Locations. *Journal für Betriebswirtschaft,* Vol. 58, No. 2, 2008, pp. 81–117.
14. Berbeglia, G., J.F. Cordeau, I. Gribkovskaia, and G. Laporte. Static Pickup and Delivery Problems: A Classification Scheme and Survey. *Top,* Vol. 15, No. 1, 2007, pp. 1–31.
15. Gribkovskaia, I., and G. Laporte. One-to-Many-to-One Single Vehicle Pickup and Delivery Problems. In *The Vehicle Routing Problem: Latest Advances and New Challenges* (B. Golden, S. Raghavan, and E. Wasil, eds.), Springer, New York, 2008, pp. 359–377.
16. Parragh, S., K. Doerner, and R. Hartl. A Survey on Pickup and Delivery Problems Part I: Transportation Between Customers and Depot. *Journal für Betriebswirtschaft,* Vol. 58, No. 1, 2008, pp. 21–51.
17. Angelelli, E., and R. Mansini. The Vehicle Routing Problem with Time Windows and Simultaneous Pick-up and Delivery. In *Quantitative Approaches to Distribution Logistics and Supply Chain Management* (A. Klose, M.G. Speranza, and L.N. Van Wassenhove, eds.), Springer-Verlag, Berlin, 2002, pp. 249–267.
18. Yano, C.A., T.J. Chan, L.K. Richter, T. Culter, K.G. Murty, and D. McGettigan. Vehicle Routing at Quality Stores. *Interfaces,* Vol. 17, No. 2, 1987, pp. 52–63.

19. Gélinas, S., M. Desrochers, J. Desrosiers, and M. M. Solomon. A New Branching Strategy for Time Constrained Routing Problems with Application to Backhauling. *Annals of Operations Research,* Vol. 61, No. 1, 1995, pp. 91–109.
20. Dethloff, J. Vehicle Routing and Reverse Logistics: The Vehicle Routing Problem with Simultaneous Delivery and Pick-Up. *Or Spektrum,* Vol. 23, No. 1, 2001, pp. 79–96.
21. Thangiah, S. R., J. Y. Potvin, and T. Sun. Heuristic Approaches to Vehicle Routing with Backhauls and Time Windows. *Computers and Operations Research,* Vol. 23, No. 11, 1996, pp. 1043–1057.
22. Kontoravdis, G., and J. F. Bard. A Grasp for the Vehicle Routing Problem with Time Windows. *ORSA Journal on Computing,* Vol. 7, No. 1, 1995, pp. 10–23.
23. Duhamel, C., J.-Y. Potvin, and J.-M. Rousseau. A Tabu Search Heuristic for the Vehicle Routing Problem with Backhauls and Time Windows. *Transportation Science,* Vol. 31, No. 1, 1997, pp. 49–59.
24. Zhong, Y., and M. H. Cole. A Vehicle Routing Problem with Backhauls and Time Windows: A Guided Local Search Solution. *Transportation Research Part E: Logistics and Transportation Review,* Vol. 41, No. 2, 2005, pp. 131–144.
25. Tasan, A. S., and M. Gen. A Genetic Algorithm Based Approach to Vehicle Routing Problem with Simultaneous Pick-up and Deliveries. *Computers and Industrial Engineering,* Vol. 62, No. 3, 2012, pp. 755–761.
26. Paraphantakul, C., E. Miller-Hooks, and S. Opasanon. Scheduling Deliveries with Backhauls in Thailand's Cement Industry. In *Transportation Research Record: Journal of the Transportation Research Board, No. 2269,* Transportation Research Board of the National Academies, Washington, D.C., 2012, pp. 73–82.
27. Ropke, S., and J. F. Cordeau. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science,* Vol. 43, No. 3, 2009, pp. 267–286.
28. Irnich, S., and G. Desaulniers. Shortest Path Problems with Resource Constraints. In *Column Generation* (G. Desaulniers, J. Desrosiers, and M. M. Solomon, eds.), Springer, New York, 2005, pp. 33–65.

Publication of these findings does not imply endorsement by Maryland Industrial Partnership or by IT Curves.

The Travel Analysis Methods Section peer-reviewed this paper.