AMERICAN JOURNAL OF MATHEMATICAL AND MANAGEMENT SCIENCES

SCHEDULING LARGE-SCALE ADVANCE-REQUEST DIAL-A-RIDE SYSTEMS

Harilaos N. Psaraftis

Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

SYNOPTIC ABSTRACT

This paper examines the scheduling of large-scale advance-request dial-a-ride systems, describes two algorithms that have been developed recently to solve problems in this area, and provides analysis and insights into how these algorithms are expected to perform under various operational scenarios and in comparison with one another. The algorithms examined are the GCR (Grouping/Clustering/Routing) and ADARTW (Advanced Dial-A-Ride with Time Windows) procedures. The paper gives an overview of both algorithms, emphasizes the differences in their operational scenarios, describes computational experience with both procedures and includes worst-case considerations for ADARTW. Extensions and directions for further research are also discussed.

Key Words and Phrases: Scheduling with Time Windows; Vehicle Routing; Dial-A-Ride Systems.

1986, VOL. 6, NOS. 3 & 4, 327-367
0196-6324/86/030327-41 $11.20

1. INTRODUCTION

This paper examines the scheduling of large-scale advance-request dial-a-ride systems, describes two algorithms that have been developed recently in the above context, and provides analysis and insights into how these algorithms are expected to perform in various operational scenarios and in comparison with one another.

The problem we are concerned with is that of routing and scheduling a fleet of vehicles to serve customers who have to be picked up from specified origins and delivered to specified destinations. Our operating scenario assumes that all of the requests for service are received well in advance of the actual time of vehicle dispatching (say, the day before at the latest). It also assumes that each customer has specified a desired pickup time or a desired delivery time (but not both) and that the vehicle operating agency has adopted some guidelines so as to ensure adequate quality of service to all customers. Such guidelines, also known as customer service guarantees (or service quality constraints), are adopted so as to avoid "intolerable" deviations from each customer's desired (pickup or delivery) times and/or excessively circuitous routes that would force a customer to spend "too long" a time on board the vehicle (as compared to that customer's direct ride time). As we shall see, such customer service guarantees are typically translated into "time-windows" for each customer's pick up and/or delivery times, in a fashion that will be explicitly defined later. At the same time, the scheduler would also like to achieve a good utilization (or productivity) for the fleet. Given that such a goal is generally in conflict with the objective of an acceptable quality of service, the problem is to find the best way to utilize available resources so that both goals are "satisficed".

The generic problem described above is known by several

names in the Transportation/Operations Research literature, such as "multi-vehicle many-to-many advance-request dial-a-ride problem", "subscriber dial-a-ride problem", "dial-a-ride problem with desired pickup or delivery times", "dial-a-ride problem with time windows", and so on. This problem is to be contrasted with the equivalent "immediate-request" problem, in which customer requests are received and processed at the same time as the actual time of vehicle dispatching (that is, in real time). Most real-world dial-a-ride agencies operate a "mixed" service, where both types of requests (in varying proportions) can be handled simultaneously. In this paper we shall be concerned with a pure advance reservation configuration only.

There has been an abundance of algorithmic development efforts in this area in the recent past. Wilson and Weissberg (1976) and Wilson and Colvin (1977) developed heuristic algorithms for the dial-a-ride system of Rochester, New York. Sexton (1979) and Bodin and Sexton (1982) developed single vehicle and multi-vehicle approximate algorithms which they applied to the subscriber dial-a-ride system of Baltimore, Maryland. Psaraftis (1980) described an exact approach for the single-vehicle immediate-request problem, and modified this approach for the equivalent problem with time windows (Psaraftis (1983a)). He also developed several polynomial-time heuristics for the single-vehicle immediate-request problem (Psaraftis (1983b, 1983c)). (A polynomial-time algorithm is one whose running time is bounded by a polynomial function of the size of the problem). Hung, Chapman, Hall and Neigut (1982) developed a procedure (also known as the "Neigut/NBS" algorithm) for the multi-vehicle time window problem, and Roy, Chapleau, Ferland, Lapalme and Rousseau (1983) described an algorithm for the same version of the problem. Finally, the current author and his colleagues developed two different multi-vehicle advance-request dial-a-ride algorithms during the past 2-3 years: in Jaw, Odoni,

Psaraftis and Wilson (1982), the "Grouping/Clustering/Routing" (or GCR) algorithm was described, while in Jaw, Odoni, Psaraftis and Wilson (1984) the "Advanced Dial-A-Ride algorithm with Time Windows" (or ADARTW) was introduced.

The focus of this paper is on the two most recent large-scale dial-a-ride algorithms, GCR and ADARTW. Specifically, Section 2 gives an overview of both algorithms and illustrates the main similarities and differences with regard to the operating scenario, the treatment of time constraints, the solution approach, and other features. Section 3 describes some recent, encouraging computational experience with the two procedures, including some comparisons of the two using the same data set. Section 4 concentrates on ADARTW, by presenting some worst-case considerations and related scenarios. The analysis identifies the features in the structure of ADARTW that are likely to cause undesirable schedules in some (rare) cases. Finally, Section 5 contains some brief concluding remarks on directions for further research, including an extension to the "mixed" demand case.

2. OVERVIEW OF THE GCR AND ADARTW APPROACHES

Although the real-world problems for which GCR and ADARTW have been designed are essentially the same, there are some differences in the way "reality" is modeled in each of the two approaches. In general, conceptual terms, the GCR approach assumes that a customer would be satisfied if he is picked up or delivered "reasonably close" to his specified pickup or delivery time, whereas the ADARTW approach provides a much stricter definition of what constitutes an acceptable level of service. Moreover, the ADARTW approach assumes that the dispatcher considers vehicle fleet size as a decision variable, whereas in GCR fleet size is treated as a given parameter. The rest of this section discusses these and other more subtle differences in

detail. For reference purposes, a list of symbols and abbreviations used throughout the text appears in Table 1, and a summary of assumptions and features of GCR and ADARTW is listed in Table 2.

2.1 Type-P and Type-D Customers. Both approaches assume that each customer requesting service has specified either a desired pickup time (DPT) or a desired delivery time (DDT), but not both. Such an assumption is reasonable in view of the fact that customers using the system are likely to be time-constrained only on one end of their trip (usually the delivery end during the morning and the pickup end during the afternoon). ADARTW actually goes further by assuming that each DPT is the earliest time the corresponding customer can be picked up (EPT) and that each DDT is the latest time the corresponding customer can be delivered (LDT). Both approaches allow a mix of both categories of customers (from now on referred to as type-P and type-D respectively) in the same problem. Also, both approaches can be modified in a straightforward way to allow a customer to specify both a DPT and a DDT. In fact (also see Section 2.2) both approaches derive, each in a different way, "equivalent" delivery (pickup) times - or time windows - for each type-P (type-D) customer. These derivations would be bypassed if a customer happens to specify both a DPT and a DDT, and these values would be used directly by the algorithms. Of course, in this case separate checks should be applied to ascertain whether the computer-specified DPT and DDT values constitute a feasible requirement (also see Section 2.2).

2.2 Time Constraints and Service Guarantees. Perhaps the most important difference between the operating scenarios assumed by the two approaches is the treatment of time constraints and the corresponding customer service guarantees. For each type-P(D)

TABLE 1: List of Symbols and Abbreviations

a,b	:	Constants Specifying MRT as Function of DRT	LDT	:	Latest Delivery Time
			LPT	:	Latest Pickup Time
ADARTW	:	Advanced Dial-A-Ride (Algorithm) with Time Windows	m	:	Index for Vehicles
			MC_{ij}	:	Marginal Insertion Cost of Customer i into Work Schedule of Vehicle j
ADT	:	Actual Delivery Time			
APT	:	Actual Pickup Time	$MINCOST_i$	:	Minimum of $COST_{ij}$ (w.r. to j)
ART	:	Actual Ride Time	MRT	:	Maximum Ride Time
AVF:	:	Active Vehicle Fleet	n	:	Number of Customers, Size of UCL
BVF	:	Backup Vehicle Fleet	S	:	Work Schedule of Vehicle
$c_1,\ldots,c_8$	:	Objective Function Coefficients in ADARTW	SW_i	:	System Workload Index for Customer i
			T	:	Constant Used to Calculate SW_i
CF	:	Conversion Factor in CCR	type-D	:	Customer Having Specified a DPT
$COST_{ik}$	:	Minimum Marginal Insertion Cost of Customer i into Work Schedule of Vehicle j	type-P	:	Customer Having Specified a DDT
			UCL	:	Unprocessed Customer List
			VC_i	:	Cost of Vehicle Resources Due to Inserting Customer i
d_{ij}	:	Direct Distance from i to j			
D(i,j)	:	Direct Trip Time from i to j	x_i	:	Pickup or Delivery Time Deviation of Customer i
DDT	:	Desired Delivery Time	y_i	:	Excess Ride Time of Customer i
DPT:	:	Desired Pickup Time	w	:	Time Window Width
DRT	:	Desired Ride Time	v_i	:	Additional Slack Vehicle Time Due to Inserting Customer i
DT	:	Time Group Length in CCR			
DU_i, DU_j	:	Disutility to Customer i	z_i	:	Additional Active Vehicle Time Due to Inserting Customer i
EDT	:	Earliest Delivery Time			
EPT	:	Earliest Pickup Time	z_{ADARTW}	:	Objective Function Value Produced by ADARTW
GCR	:	Grouping/Clustering/Routing (Algorithm)	z_H	:	Objective Function Value of Heuristic H
i,j,k	:	Indices for Points, Customers, Clusters	z_{OPT}	:	Optimal Value of Problem
K	:	Pool Size	z_R	:	Reference (Maximum) Objective Value
ln	:	Natural Logarithm	O(n)	:	Growing at Least as a Linear Function of n

TABLE 2: Summary of assumptions and features of GCR and ADARTW algorithms and their operating scenarios. Asterisks (*) denote nonbinding assumptions that can be relaxed without major modifications in the logic of the algorithm.

ASSUMPTION/FEATURE	GCR ALGORITHM	ADARTW ALGORITHM
Demand Environment	Pure Advance Request	Pure Advance Request
Customer Desired Service Times	Each customer specifies either a desired pickup time (DPT) or a desired delivery time (DDT) but not both	Same as GCR. In addition, each customer's DPT (DDT) is equal to that customer's earliest pickup (latest delivery) time EPT (LDT).
Customer Type Mix	Both DPT and DDT-specified customers can exist in same problem	Same as GCR
Customer Service Guarantees	Soft: Attempts (with no guarantees) to service each customer within a time interval of specified duration. Non-empty vehicles cannot idle.	Hard: Accepts upper bounds on each customer's deviation from desired (pickup or delivery) time and on each customer's ride time. Non-empty vehicles cannot idle.
Customer Dwell Times	Zero	Zero (*)
Distance Matrix	Symmetric	General
Vehicle Fleet Size	A User-Input. Can vary in time.	A Decision Variable. Can vary in time.
Vehicle Types	Identical	Identical (*)
Vehicle Capacity Constraint	No	Yes
Options if Service Guarantees Cannot Be Met With Available Resources	Problem never infeasible. If service deteriorates, scheduler may either deny service or iterate with more vehicles on a trial/error basis.	Can either deny service to some customers or add more vehicles at a cost.
Objective Function	Not explicitly defined. Attempts to equalize vehicle workload, maximize total vehicle productivity and minimize route circuity.	Minimizes a prescribed function of customer disutility and vehicle resource utilization. User-calibrated via eight coefficients
Solution Method	Grouping-Clustering-Routing: Customer decomposition in time; seed customer selection; simultaneous assignment of other customers; look-ahead capability; use of a routing algorithm as subroutine.	Sequential Insertion: Includes fast screening test for feasible insertions and optimization of customer assignments.
Worst-Case Performance	Not Applicable	Worst-Case Relative Error: ∞ no Worst-Case Absolute Error: Unknown
Average Case Performance	Not Applicable	Unknown
Computational Complexity	$O(mn + n^2)$ for n customers and m vehicles on the average	$O(n^4)$ in worst case
Maximum Problem Size Solved	2,617 customers	2,617 customers
CPU Time for Max. Problem VAX 11/730	6-7 mins.	12 mins.
Computer Language used	FORTRAN	PL1
Extension to "Mixed" Demand Case	Difficult	Easier

customer, the GCR algorithm simply derives an equivalent desired delivery (pickup) time DDT(DPT), as a function of that customer's desired pickup (delivery) time DPT(DDT) and his/her direct ride time DRT as follows (see also Figure 1):

$$\text{For type-P customers, } DDT = DPT + CF * DRT \quad (1)$$

$$\text{For type-D customers, } DPT = DDT - CF * DRT \quad (2)$$

where CF (for "conversion factor") is a calibration parameter (≥ 1) to account for possible circuity in that customer's ride. (A high value of CF means that the difference DDT-DPT is greater than DRT, hence the customer may arrive to his/her destination via a circuitous route) GCR then divides the entire time horizon of the problem into adjacent "time groups" (see Figure 1). Each time group has a duration of DT, with DT being another calibration parameter. Sensitivity on the selected values of CF and DT is reported in Section 3.

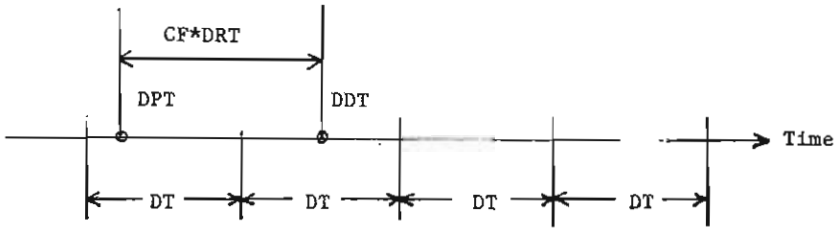

Figure 1: Time groups in GCR algorithm

In GCR, customer service guarantees (and corresponding time constraints) are "soft" in the sense that the algorithm will attempt to service (pickup or deliver or both) each customer only within the time group that encompasses his DPT or DDT, without directly considering that customer's individual desired time. For instance, if time groups are set up every 30 minutes starting

at 8:00 a.m., an attempt will be made to pick up a customer who has specified a DPT of 10:37 a.m. between 10:30 a.m. and 11:00 a.m., with the projected actual pickup time being quoted to him well in advance of actual vehicle dispatching.

The operating scenario of ADARTW is rather different (see Figure 2).

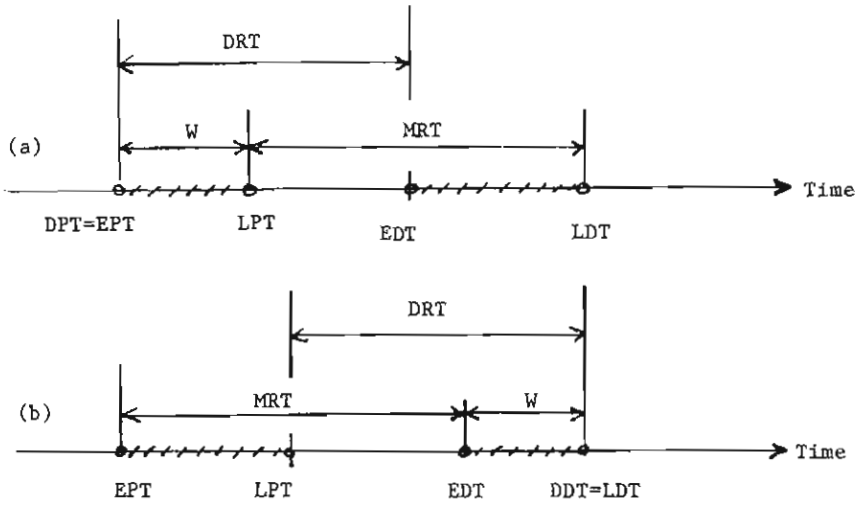

Figure 2: Time windows in ADARTW algorithm, (a) for type-P customers, (b) for type-D customers.

For each type-P(D) customer ADARTW derives three types of times, which, together with that customer's EPT (LDT), constitute a pair of time windows (one for pickup and one for delivery) for that customer. These times are defined as follows:

For type - P customers (EPT specified, see Figure 2a):

$$\text{Latest pickup time} \quad LPT = EPT + W \quad (3)$$

$$\text{Earliest delivery time} \quad EDT = EPT + DRT \quad (4)$$

Latest delivery time LDT = LPT + MRT. (5)

For type - D customers (LDT specified, see Figure 2b):

Earliest delivery time EDT = LDT - W (6)
Latest pickup time LDT = LDT - DRT (7)
Earliest pickup time EPT = EDT - MRT, (8)

where W is a user-specified time window width and MRT is defined as the customer's maximum ride time. MRT is a function of a customer's DRT. A typical, but not binding functional relationship is MRT = a +b*DRT, with a≥0 and b≥1 being user inputs.

Customer service guarantees in ADARTW are "hard", in the sense that each customer's actual ride time (ART) and actual pickup time (APT) (or actual delivery time (ADT), as appropriate) must satisfy the following constraints:

for all customers: DRT≤ART≤MRT (9)
for type-P customers only: EPT≤APT≤LPT (10)
for type-D customers only: EDT≤ADT≤LDT (11)

where LPT in (10) and EDT in (11) are defined by (3) and (6) respectively.

It is easy to check that if (9) and (10) ((9) and (11)) are satisfied for a type-P(D) customer, then that customer's ADT (APT) must lie between EDT(EPT) and LDT (LPT) as defined by (4) and (5) ((8) and (7)) respectively (the opposite is not necessarily true).

If we allow a customer to specify both a DPT and a DDT, then both these values would be used as such in the GCR algorithm, that is, the conversion implied by (1) or (2) would not be used. However, in this case we should also require that the ratio

(DDT-DPT)/DRT be greater than 1.0 (otherwise the request would be infeasible). In ADARTW, if a customer has specified both a DPT and a DDT, we would use EPT=DPT, LDT=DDT, and derive LPT and EDT from (3) and (6) respectively. In this case, the conversions implied by (4), (5), (7) and (8) would not be used. Also, in this case the maximum ride time constraints would be automatically satisfied whenever DPT-DPT a+b*DRT.

The issue of which of the two approaches ("soft" or "hard") is more appropriate, has been and is being widely debated, both among algorithm developers and among operators and policy makers. We feel that each approach has merit, as well as drawbacks; these are discussed further in Section 5.

2.3 Rejecting Customers versus Adding More Vehicles. Since the "hard" approach in ADARTW raises the prospect of infeasibility (which does not exist in GCR due to the absence of such constraints), there are two basic user options by which this issue can be handled: Either by denying service to those customers whose service guarantees cannot be met, or by adding more vehicles (drawn from a "backup vehicle fleet"), at a cost. This cost is explicitly considered in ADARTW's objective function by a term that quantifies the "cost of vehicle resources" (see Section 2.6). Such decisions can also be made in GCR, but only at the discretion of the dispatcher. That is, whenever he feels that some customers suffer from poor quality of service, or whenever the capacity of some vehicle is exceeded (GCR does not consider vehicle capacity constraints explicitly), the dispatcher has the option to deny service to some customers, or rerun GCR with more vehicles until service quality becomes acceptable and capacity constraints are satisfied..

2.4 Optimizing Vehicle Fleet Size. The treatment of vehicle fleet size as a function of time is another major difference

between the two approaches. In GCR, the number of vehicles to be operated throughout the day is supplied by the user, possibly as a function of customer demand (this number can vary over time). In ADARTW, this number is a decision variable, determined so as to optimize a prescribed objective function which includes vehicle resource costs (see also equation (16) below).

2.5 Objective Function and Solution Method/GCR. The ultimate objective in GCR is to maximize total vehicle productivity (in terms of passengers serviced per vehicle hour). However, the design of the algorithm is such that the above objective is never explicitly considered by the procedure. Instead, GCR produces a set of routes and schedules by attempting to optimize a set of surrogate measures, defined in such a way that the resulting solution is likely to achieve good productivity and a good quality of service. The general solution method of GCR, along with the performance measures that the algorithm attempts to optimize at each step, is summarized as follows (for complete details see Jaw, Odoni, Psaraftis and Wilson (1982)).

STEP 1 (Grouping): Based on each customer's DPT and DDT (one of them being input and the other calculated by (1) or (2)), and on user inputs DT and CF, divide the time horizon into equal and consecutive time groups, and then assign customers to those groups, according to the interval into which their DPT's or DDT's fall. This is just a decomposition of the problem by time, and no optimization is involved.

STEP 2 (Clustering): For each time group defined in Step 1, further subdivide all customers of that time group into "clusters", and then assign a vehicle to service (pickup, deliver, or both) all customers in each cluster. The mechanism by which this is done is fairly intricate (see Jaw, Odoni,

Psaraftis and Wilson (1982)) but can be described roughly as follows:

For each time group k do the following:

Step 2.1: Let m be the number of vehicles available (but thus far unassigned) in k. Identify m unassigned customers in k and proclaim each of them a "seed" customer for a new cluster in k. Do this by attempting to optimize a measure of "seed dispersion", that is, declare a customer a "seed" if he maximizes the "distance" between him and the closest seed already chosen. "Distance" between two seeds is defined as a function of the distances between origins and destinations of the corresponding customers, depending on the scenario.

Step 2.2: Assign the m available vehicles to the m seed customers obtained in Step 2.1. Do this by attempting to minimize the total cost of the assignment. The cost associated with each vehicle-seed pair is defined as the distance between the vehicle and the origin of that seed.

Step 2.3: Add all other customers in group k, one by one, to either the new clusters formed in Step 2.1, or, to already existing clusters carried from group k-1. Each cluster is allowed to grow in such a way so that "vehicle workload" is spread out evenly among all vehicles. The vehicle workload computation includes some "look ahead" features to take into account the locations of the origins and destinations of customers belonging to time group k+1.

STEP 3 (Routing): For each time group k and for each cluster identified in Step 2, use a single-vehicle routing algorithm to form a route that will service the customers of that cluster (vehicle capacity constraints may be considered in this step of the procedure). The objective here is to minimize route length

for each cluster. GCR offers four options with respect to which routing algorithm will be used in Step 3 (see Psaraftis (1980, 1983a, 1983b, 1983c)).

The main novelty (and core) of the GCR algorithm is its clustering step, whose basic purpose is to identify sets of customers that have good "directivity" characteristics, that is, have similar directional patterns of travel (e.g. "north to south", "east to west"). Such a task is very difficult in a many-to-many environment because customers whose origins are close to one another (and hence seem suitable to be picked up by the same vehicle) may have their destinations widely separated, thus making the eventual vehicle route inefficient. Step 2 of the algorithm was tested extensively and was shown to produce clusters of good "directivity".

2.6 Objective Function and Solution Method/ ADARTW. The solution method of ADARTW is completely different from that of GCR. In fact, ADARTW uses an explicitly defined objective function. The basic idea of the algorithm is to build routes and schedules by sequentially inserting each customer into the most promising "provisional" route and schedule constructed thus far. Its main novelty lies in the way by which feasible insertions are identified. The algorithm's basic version can be described roughly as follows (for complete details see Jaw, Odoni, Psaraftis and Wilson (1984)):

STEP 0 (Initialization): For all customers, set up time windows according to (3)-(8). Rank-order all customers by nondecreasing order of EPT. Put all customers into an "unprocessed customer list" (UCL). Let n = size of UCL. Set up "active vehicle fleet" (AVF) and "backup vehicle fleet" (BVF). Set up "pool size" K (K = user input ≥ 1)

STEP 1 (Candidate Customer Selection): Take the top k = min. (n, K) customers out of UCL and into "pool of customers for immediate assignment" (or, "pool"). If K = 0, END.

STEP 2 (Tentative Insertion): For all customers i in "pool", do the following:

STEP 2.1: For all vehicles j in AVF, do the following: Find all feasible ways in which i can be inserted into the work schedule of j so that (a) no service guarantees to any customers are violated; (b) no vehicle capacity constraint is violated (see also below). If none is found, examine another vehicle in AVF. If none exists, then either declare i REJECTED, or consider a vehicle j from BVF and repeat. Find the insertion of i into the work schedule of vehicle j which results in minimum marginal insertion cost (see also below). Call this cost $COST_{ij}$.

STEP 2.2: For customer i, find vehicle j*(i) for which $COST_{ij*(i)}$ $COST_{ij}$ for all j examined in Step 2.1. Call this minimum cost $MINCOST_i$. Tentatively assign i to j*(i).

STEP 3 (Best Insertion): Find customer i* in "pool" for which $MINCOST_{i}$ $MINCOST_{i}$ for all i in "pool". Permanently assign i* to j*(i*).

STEP 4 (Update): Update AVF and BVF. Update all vehicle schedules. Dump remaining customers of "pool" back into UCL. Set n = n-1 and go to Step 1.

Central to ADARTW is the concept of a "schedule block". A schedule block is a sequence of stops (pickups and deliveries) scheduled to be visited by a vehicle, with no idle time (or "slack") in between. Figure 3 shows two such schedule blocks: Block 1 represents the pickup and delivery of customers 1 and 2

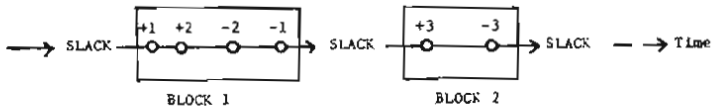

Figure 3: The concept of a schedule block (A "+" denotes pickup and a "-" denotes delivery)

(in the sequence shown) and Block 2 shows the pickup and delivery of customer 3. Slack time will generally exist between two consecutive schedule blocks. In Figure 3 a slack time between Blocks 1 and 2 means that the time it takes the vehicle to drive from the destination of customer 1 to the origin of customer 3 is less than the difference in the scheduled times of these two operations (deliver # 1 and pick up # 3). Such a schedule thus implies that the vehicle should idle at any one of these two points (or at both) for a total amount of time equal to the value of the slack.

Step 2.1 of ADARTW attempts to insert a new customer somewhere within the work-schedule of a vehicle. It does this by continually keeping track of how much each stop in each schedule block can be shifted upstream or downstream so that no constraints are violated. It also keeps track of the maximum available slack between consecutive blocks. Given a new customer and a candidate vehicle, there are three insertion options open to ADARTW: First, inserting both origin and destination within (or around) one individual schedule block. Second, creating a new schedule block by inserting these two points between two consecutive schedule blocks. And third, inserting them into two different schedule blocks. Notice that this last option implies that all slack between the two "receiving" blocks will have to disappear, since one of our operating assumptions is that the vehicle is not allowed to idle if a customer is on board (see Table 2). The basic mechanics of identifying feasible insertions for the first two options are described in Jaw, Odoni, Psaraftis and Wilson (1984)) and for all options in Jaw (1984). In this paper, we shall have an opportunity to observe the insertion

logic of ADARTW in conjunction with the worst-case considerations of Section 5.

The marginal cost MC_{ij} of a (feasible) insertion of a customer i into the work schedule S of a vehicle j (with S already including other customers k) is defined as follows (subscript j dropped from all terms for convenience):

$$MC_i = DU_i + \sum_{k \in S} (DU_k - DU_k) + VC_i \qquad (12)$$

where DU_i (DU_k) is the disutility to customer i (k) after the insertion of customer i, DU_k is the disutility of customer k before the insertion of customer i and VC_i is the cost to vehicle resources.

The functional form assumed by ADARTW for DU_i (and, by analogy for DU_k and DU_k) is a quadratic function of x_i, the pickup or delivery time deviation of customer i, and, of y_i, the excess ride time of customer i, as follows:

$$DU_i = c_1 x_i + c_2 x_i^2 + c_3 y_i + c_4 y_i^2 \qquad (13)$$

where x_i and y_i are defined as follows:

$$x_i = \begin{cases} APT_i - EPT_i & \text{if customer i is Type-P} \\ LDT_i - ADT_i & \text{if customer i is Type-D} \end{cases} \qquad (14)$$

$$y_i = ART_i - DRT_i \qquad (15)$$

and c_1 to c_4 are user-specified nonnegative constants.

A quadratic form was chosen for DU_i because it is both more general than a linear form and more realistic in representing actual customer dissatisfaction.

Finally, VC_i is assumed to be a function of z_i, the additional active vehicle time due to insertion of customer i, and of w_i, the additional slack vehicle time due to insertion of customer i, as follows:

$$VC_i = c_5 z_i + c_6 w_i + SW_i \ (c_7 z_i + c_8 w_i) \qquad (16)$$

where c_5 to c_8 are user-specified non-negative constants.

The term denoted by SW_i in (16) is called "system workload index" and is a measure of how heavy the workload of vehicle j is at the time of the insertion of customer i. The existence of this term in (16) penalizes additional vehicle time more in situations in which vehicle workload is already high. SW_i is calculated according to the following empirical formula:

$$SW_i = \frac{\text{(Number of system customers in } [EPT_i - T, EPT_i + T])}{\text{(Effective number of vehicles in above interval)}} \qquad (17)$$

with T being another constant.

We note that in case i is not inserted into two different schedule blocks (which, in fact, would happen in most cases due to MRT-constraints and penalization of excess ride time y_i), the summation in (12) need only be evaluated for customers k belonging to at most one schedule block of S.

We end this section by noting that Table 2 displays some additional (but minor) features that help highlight the difference between GCR and ADARTW.

3. COMPUTATIONAL EXPERIENCE

There has already been extensive computational experience with both GCR and ADARTW. This has included running the algorithms with both simulated and real data, performing

extensive sensitivity analyses on various parameters and, finally, comparing the two approaches on the same data set. The general assessment of these runs as far as both computational efficiency (CPU time) and performance (vehicle productivity, route circuity, customer service quality) are concerned has been quite positive for both approaches. In addition, both algorithms have dealt successfully with by far the largest-size problem database that has yet been examined (more than 2,600 customers), and can deal with even larger problems with no significant difficulty. This section summarizes these results and describes the cases on which the two algorithms were compared.

3.1 GCR Algorithm. In Jaw, Odoni, Psaraftis and Wilson (1982) a series of simulation runs of the GCR algorithm were described. These involved simulating 4 hours of service in a 6x6 square mile geographical area. Customer origins and destinations were distributed uniformly and independently over the area, and their requests were distributed over the 4-hour interval at a rate ranging from 100 to 500 customers per hour. A sensitivity analysis was made on the number of available vehicles (which ranged from 10 to 60), on the length of the time interval DT (which was chosen to be 30 or 60 minutes) and on the conversion factor CF (chosen to be 1.0 or 1.5). A 50%-50% mix of type-P and type-D customers was assumed. A general conclusion from these runs was that DT is the most important calibration parameter. In the runs examined, the best results were achieved with DT = 60 minutes whereas the performance of the algorithm generally deteriorated with DT smaller. This is probably due to the fact that GCR is less successful in linking adjacent time groups if DT is too small. Average vehicle productivity ranged from about 3.5 to more than 15 passengers per vehicle hour, depending on the scenario. Increasing the number of vehicles predictably resulted in lower productivity but in better quality of service, but did

not necessarily result in higher CPU time, due to the tradeoff between the clustering and routing steps. This tradeoff is due to the fact that a higher number of vehicles generally results in more clusters (and hence more CPU time for the clustering step) but also in fewer customers per cluster (and hence less CPU time for the routing step). In all runs, CPU time ranged from 0.50 to about 16 minutes on a VAX 11/750, again depending on the size of the problem.

3.2 ADARTW Algorithm. Similar (although smaller-scale) simulation runs were performed in Jaw, Odoni, Psaraftis and Wilson (1984)) and in Jaw (1984) for ADARTW. A 9-hour, 6x6 square mile scenario was examined in Jaw, Odoni, Psaraftis and Wilson (1984), under the same (as in Section 3.1) general assumptions regarding the distribution of origins, destinations and desired times. In all cases the total number of customers were 250, with the demand pattern throughout the period of service fluctuating between 20 and 40 requests per hour. Time windows were set to 20 or 10 minutes and the maximum ride time was set to 5 minutes plus twice the direct ride time of each customer. An initial fleet size of 4 vehicles and an option to add more vehicles if necessary was assumed. Sensitivity analysis was performed on the values of the eight objective function coefficients and on the width of the time window. A general observation from these runs was that vehicle productivity (which ranged between 3 and 4 passengers per vehicle hour) seemed to be a decreasing function of c_1, c_2, c_3, and c_4, especially c_1 (the coefficient of the linear term representing pickup or delivery deviation). Average deviation from desired (pickup or delivery) time seemed to be most sensitive to and negatively correlated with both c_1 and c_2 (the coefficient of the quadratic term associated with pickup or delivery deviation), especially in the case of a wider time window. We should emphasize however that

all of the above dependencies, although definitely identifiable, were not that pronounced. Thus, it seems (at least from these runs) that although the selection range of ADARTW's objective function coefficients is extremely broad, vehicle productivity and service attributes seem to depend less on the values of those coefficients, and more on other, problem-specific inputs, such as the time window width W, and the relationship between MRT and DRT (coefficients a and b). In that respect, a "tightly constrained" problem (W small, MRT close to DRT) predictably results in a lower productivity and in a higher number of vehicles if no customers are to be rejected. In our runs, between 10 and 14 vehicles were eventually used, this number never being a strictly monotonic function of any of the objective function coefficients. CPU time averaged about 20 seconds on the VAX 11/750.

It should be also stressed that all the runs of ADARTW reported in Jaw , Odoni, Psaraftis and Wilson (1984) referred to the version of the algorithm in which (a) the "pool size" K was set equal to 1 and (b) a customer's origin and destination could not be inserted into different schedule blocks. Larger pool sizes and the possibility of insertion into two different blocks were examined in detail in Jaw (1984). This analysis showed no significant improvement in productivity and service attributes for larger values of K, although this certainly resulted in a significant increase in CPU time. Nevertheless, we feel that a "small" value of $K \geq 1$ (say, between 5 and 10) is appropriate so as to reduce the myopia of ADARTW when $K = 1$ (an example of this is presented in Section 4). An insertion into different schedule blocks was observed to happen very rarely (for reasons that were discussed in Section 2.6) and to generally result in very small gains.

3.3 Comparison Between GCR and ADARTW. Despite the fact that the abstract problems (or, models) for which GCR and ADARTW have

been designed are different, a comparison between the two approaches is indeed worthwhile, because both eventually refer to the same real-world problem. However, significant caution must be exercised in the design of the test cases so that the comparison is fair.

We have compared the two algorithms via two cases: one using real data and one using simulated data. Before we describe these cases, we observe that since GCR is "less constrained" than ADARTW (as far as quality of service, adherence to time constraints, etc. are concerned), one would a priori expect GCR to produce solutions of higher productivity than ADARTW if both procedures were applied to the same problem. Interestingly enough, the first comparison of the two algorithms produced exactly the opposite result. In that comparison ADARTW was observed to dominate GCR both in terms of quality of service and productivity. However, the second comparison revealed that such superiority of ADARTW is not always guaranteed, in the sense that GCR may, in some cases, achieve a better productivity than ADARTW without a significant deterioration in the quality of service. Details of the two comparisons are as follows:

The first comparison involved running both algorithms with a real database. The database covered about 16 hours of operation of the flexible-route system of Rufbus GmbH Bodenseekreis, in the city of Friedrichshafen, West Germany. The database included information on 2,617 customers, of which 2,397 were type-P while the remaining 220 were type-D (this database is maintained at M.I.T.).

Running either algorithm with this database presented some initial obstacles, because the database had features that did not match the operating assumptions of the procedures. Thus, some effort was spent to "convert" the original database to one that could be processed by both algorithms. Perhaps the most drastic of those conversions was the treatment of those type-P customers

in the database who were "immediate-request" customers. For both algorithms, we converted these customers to "advance-request" customers by defining their DPT as the actual time of their request. In addition, since the distance matrix of the database was not symmetric (although asymmetries were not that pronounced), we ran the GCR algorithm with a converted distance matrix, by replacing each entry d_{ij} of the matrix with $0.5(d_{ij} + d_{ji})$. ADARTW was tested against the original matrix.

We were told that the Rufbus fleet consisted of one 33-passenger vehicle, four 9-passenger vehicles and twenty-three 17-passenger vehicles. Since both GCR and ADARTW assume identical vehicles, we ran ADARTW with a capacity of 17 for all vehicles. An "active vehicle fleet" of 10 vehicles, with the option to add more vehicles if necessary was assumed. In GCR (which does not accept capacity constraints) we externally varied the number of vehicles throughout the day with a peak of 21 vehicles. Vehicle speed was set to 15 mph for both procedures.

Finally, we were told that Rufbus schedulers tried to keep a 15-minute time window, although at times this constraint was relaxed to 60 minutes to avoid rejecting "immediate-request" customers. We somewhat arbitrarily translated this policy to the following parameters: For GCR, we set DT = 30 minutes and CF = 1.0. For ADARTW, we set W = 15 minutes, and MRT = 5 mins + 1.5*DRT. We selected the objective function coefficients of ADARTW as follows: $c_1=3$, $c_7=1$, $c_8=0.8$, and all others zero. Finally, we set the parameter T in (17) equal to 60 minutes.

The above description highlights some of the difficulties involved in testing any generic dial-a-ride algorithm on a real-life database, and the risks of comparing two procedures which have been based on different assumptions. Given the nature of some of the conversions, we have reservations about comparing actual Rufbus schedules (which refer to a "mixed" demand case) with the results of GCR or ADARTW. However, we are more

confident as far as the legitimacy of the comparison between GCR and ADARTW on this "converted" database is concerned. This comparison is displayed in Table 3 (Rufbus scheduling results are also displayed, for illustration purposes only).

TABLE 3: Comparison between GCR and ADARTW on the Rufbus "converted" database. Rufbus results are shown for illustration purposes only. All times are in minutes. Productivity is in passengers per vehicle hour. By convention, early deliveries (pickups) and late pickups (deliveries) have time deviations greater (less) than or equal to zero respectively.

	GCR	ADARTW	RUFBUS
Vehicles used	21	17	28
Vehicle productivity	10.53	12.06	8.87
Average deviation (late pickup or early delivery)	17.59	6.60	11.9
Average deviation (early pickup or late delivery)	-13.31	0	N.A.
Average ride time ratio	2.3	1.54	N.A.
CPU time (VAX 11/750)	7	12	N.A

We observe that both algorithms produce reasonable schedules with regard to quality of service and productivity levels. We also observe that, for this example, and with the exception of CPU time, ADARTW clearly dominates GCR on every count, that is, with respect to both quality of service and productivity. This dominance is perhaps surprising in view of the fact that the problem solved by ADARTW is more constrained than the one solved by GCR.

The second comparison was made based on a simulated database of only 25 customers (see Table 4). Customer origins and destinations were uniformly and independently distributed on a

TABLE 4: Simulated database of 25 customers. All X and Y coordinates are in hundredths a mile. All desired pickup and delivery times are in hours and minutes.

Customer (type)	Origin Coord. X	Origin Coord. Y	Dest. Coord. X	Dest. Coord. Y	DPT	DDT
1 (D)	180	437	30	582		7:42
2 (D)	380	288	442	198		7:37
3 (D)	412	490	488	404		7:45
4 (P)	27	413	169	516	7:02	
5 (D)	197	29	205	153		7:56
6 (P)	326	490	324	166	7:14	
7 (D)	62	543	182	14		8:33
8 (D)	163	77	254	293		8:10
9 (P)	394	389	29	144	7:21	
10 (P)	31	66	283	278	7:23	
11 (D)	377	420	332	97		8:33
12 (D)	528	50	285	98		8:39
13 (P)	465	406	408	144	7:48	
14 (P)	472	73	93	34	7:51	
15 (D)	212	161	183	474		8:53
16 (D)	243	106	589	436		9:11
17 (D)	253	569	4	547		8:55
18 (D)	89	427	375	128		9:19
19 (D)	448	111	243	298		9:10
20 (D)	362	57	577	337		9:17
21 (D)	333	104	102	251		9:13
22 (D)	352	483	428	259		9:16
23 (P)	491	537	238	584	8:28	
24 (D)	348	33	520	259		9:27
25 (P)	27	194	33	299	8:40	

6x6 square mile area. Eight of those customers were type-P and 17 were type-D. Desired pickup (or delivery) times were uniformly distributed within approximately three hours of service. We ran both algorithms with 4 vehicles and a vehicle speed of 15mph. In GCR we used DT=60 minutes and CF=1.0, and in ADARTW W=30 minutes and MRT=5 minutes + 2*DRT. Finally, the objective function coefficients of ADARTW were set to $c_7 = 1$, $c_8 = 0.8$, and all others zero. Table 5 shows the results of the comparison. We can see that, in this example, GCR outperforms ADARTW in terms of total vehicle time, average ride time ratio and vehicle productivity, without using more vehicles. A further examination of the schedules (see Jaw (1984)) reveals that GCR delivers only two type-D customers more than 30 minutes earlier than their DDT's and all type-P customers less than 30 minutes later than their DPT's. Furthermore, no actual ride time in the GCR schedule exceeds the maximum ride time constraints of ADARTW.

TABLE 5: Comparison between GCR and ADARTW on the simulated database. All times are in minutes. Productivity is in passengers per vehicle hour. By convention, early deliveries (pickups) and late pickups (deliveries) have time deviations greater (less) than or equal to zero respectively.

	GCR	ADARTW
Vehicles used	4	4
Total vehicle time	360	389
# of late deliveries (out of 17)	3	0
Average deviation given late delivery	-16.3	0
Average deviation given early delivery	16.6	15.6
# of early pickups (out of 8)	3	0
Average deviation given early pickup	-9	0
Average deviation given late pickup	19.2	19.4
Average ride time ratio	1.33	1.49
Vehicle productivity	4.17	3.86
CPU Time (VAX 11/750)	1.12	1.4

Of course, the last comparison can be criticized on the grounds that ADARTW might conceivably obtain better results by an appropriate choice in its objective function coefficients. However, given that such choice is sometimes not clear, we believe that the comparison has demonstrated the premise that GCR can outperform ADARTW in some cases.

Due to small scale (and conceivably contrived nature) of this last comparison, it is clear that additional computational experience with both procedures is necessary in order to shed more light on what types of instances are likely to be more "favorable" to one of the algorithms as opposed to the other. However, based on our experience with both procedures to date, and on the fact that ADARTW possesses more features that would fit a given real-world situation (such as capacity constraints, general distance matrix, etc.), we conjecture ADARTW is likely to outperform GCR in most realistic operational situations and hence, is better suited for implementation in such situations. Still, we regard GCR as a viable tool for quickly producing reasonable solutions which could be used either in planning situations or in operational situations on a preliminary basis. In the latter case, GCR could be used to provide good starting solutions (e.g. customer clusters) that could be further improved with the help of a post-optimization routine (for instance, similar in spirit with the "swapper" algorithm of Bodin and Sexton (1982)).

The GCR and ADARTW codes have been written in FORTRAN IV and PL1 respectively, and their approximate lengths are 3000 lines for GCR and 2000 lines for ADARTW. These codes are maintained at M.I.T. (inquiries should be addressed to the author).

4. WORST-CASE CONSIDERATIONS IN ADARTW

Worst-case analyses of insertion heuritics have already been performed for several routing problems to-date. For instance, in

Rosenkrantz, Sterns and Lewis (1974), the worst-case error ratio of both the nearest insertion and the cheapest insertion methods, as applied to a "triangle-inequality" Traveling Salesman Problem of n nodes, was shown to be equal to 1.0. The worst-case error ratio of a heuristic H as applied to a minimization problem P is defined as the maximum, over all possible instances of P, values of the ratio $(Z_H - Z_{OPT})/Z_{OPT}$ where Z_H is the value of the objective Z of the problem when solved by H, and Z_{OPT} is its optimal value. (Such a definition implies that the two methods mentioned above produce tours which are at most 100% longer than the optimal tour, since in this case $(Z_H - Z_{OPT})/Z_{OPT} = 1.0$). For the same version of the Traveling Salesman Problem it is also known that both the arbitrary insertion and the farthest insertion methods exhibit a worst-case error ratio of 2*ln(n) - 0.84 (see also Golden, Bodin, Doyle and Stewart (1980)). More recently, a worst-case analysis of some heuristics for the vehicle routing and scheduling problem with time windows was carried out by Solomon (1983). This analysis showed that a variety of heuristics (including, but not limited to, insertion methods) exhibit an $\Omega(n)$ worst-case error ratio for n customers as far as the number of vehicles used, the total distance traveled, and the total schedule time are concerned (that is, the worst-case error ratio of these heuristics was shown to be at least equal to a linear function of n).

The pertinent question here is, whether any kind of similar analysis can be carried out for ADARTW (this issue is "academic" for GCR, since that approach does not use an explicitly defined objective function). A first observation is that such a task would be extremely difficult, given the nontrivial nature of the problem (regarding both its objective function and its constraints) and the nontrivial design of the algorithm. Jaw (1984) posed the question whether the $\Omega(n)$ behavior reported in Solomon (1983) might be true also for ADARTW, but left the question open.

Another observation that can be made is that analyzing heuristics in terms of their worst-case relative errors (that is, in terms of their worst-case error ratios) may make little sense in certain problems: for a minimization problem P and a heuristic H, if there are instances of P for which $Z_{OPT} = 0$ and $Z_H > 0$, the worst-case error ratio of H becomes $+\infty$. This may lead to erroneous conclusions regarding the merits of H, particularly if the absolute error $Z_H - Z_{OPT}$ is small.

It is important to recognize that whereas such a "singular" ($Z_{OPT}=0$) behavior is rare in most vehicle routing problems (in which typically Z measures such quantities as total distance traveled, number of vehicles, etc., and hence $Z_{OPT} > 0$), this is not necessarily the case for the problem solved by ADARTW, due to the form of the objective function considered. Indeed, by an appropriate choice in the coefficients c_1 to c_8 (which are parts of the input that defines a specific problem instance), it is relatively easy to envision cases for which $Z_{OPT} = 0$. Such cases would correspond to objective functions measuring total customer disutility only (at least one of c_1, c_2, c_3, c_4 being nonzero and $c_5 = c_6 = c_7 = c_8 = 0$), in which the minimum achievable value of Z might go all the way down to zero. This means that if there exists such a case ($Z_{OPT} = 0$) for which $Z_{ADARTW} > 0$, the worst-case error ratio of ADARTW is $+\infty$.

Not surprisingly, such a case can be constructed. Despite the fact that the use of the ratio $(Z_{ADARTW} - Z_{OPT})/Z_{OPT}$ becomes meaningless when $Z_{OPT} = 0$, we shall present one such case below, so as to increase our understanding of how ADARTW behaves, and to indicate when and why it may behave poorly. We also discuss some alternative worst-case criteria later in this section.

Consider three requests on the Euclidean plane, as shown in Figure 4. Assume there is only one vehicle (of unit speed), initially located at point O. Assume also that two of the requests (customers 2 and 3) have specified desired (earliest) pickup times equal to $\sqrt{2}/2 = 0.71$ and $1 + \sqrt{2}/2 = 1.71$

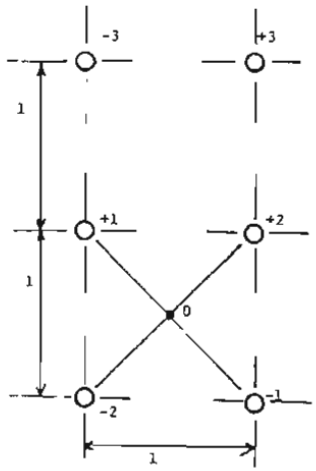

Figure 4: Customer and vehicle locations in example.

respectively. Customer 1 has specified a desired (latest) delivery time equal to $5 + \sqrt{2}/2 = 5.71$. Finally, assume that the time window W for this problem has been set equal to 6, and that the maximum ride time for all customers is equal to 4 (independent of direct ride time). It can be seen that the service guarantees for this problem are rather loose. Still, assume that we would like to obtain the lowest possible total customer disutility. For this problem, assume that the disutility of each customer is just equal to that customer's deviation from his desired time (pickup or delivery, as appropriate). That is, assume that the only nonzero objective function coefficient is $c_1 = 1$.

Given the above data, Table 6 displays the calculated

TABLE 6: Time windows for 3 customers of example (asterisked times are inputs; all other times are calculated).

Customer	Type	EPT	LPT	EDT	LDT
1	D	-4.29	4.29	-0.29	5.71*
2	P	0.71*	6.71	2.12	10.71
3	P	1.71*	7.71	2.71	11.71

(according to (3) - (8)) pickup and delivery time windows for all three customers.

A cursory investigation reveals that the optimal solution to this problem corresponds to the route (0, +2, +3, -3, +1, -2, -1) shown in Figure 5. The corresponding schedule is shown in Table 7. This solution is optimal because the total deviation from desired time is zero ($Z_{OPT} = 0$), the lowest possible. Notice also that there is no slack in the vehicle schedule.

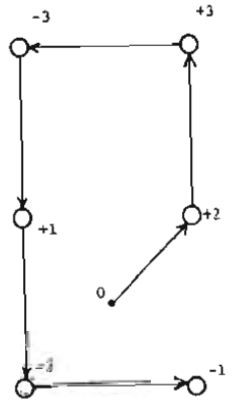

Figure 5: Optimal route (see also Table 7

Table 7: Optimal Schedule (see also Figure 5)

Stop	Arrival Time	Departure Time	Idle Time	Desired Time	Time Deviation	Ride Time
0	0	0	0	-	-	-
+2	0.71	0.71	0	0.71	0	-
+3	1.71	1.71	0	1.71	0	-
-3	2.71	2.71	0	-	-	1
+1	3.71	3.71	0	-	-	-
-2	4.71	4.71	0	-	-	4
-1	5.71	-	-	5.71	0	2

ADARTW solves the above problem as follows: If the "pool size" K is set equal to one, the procedure starts by first

considering customer 1, the customer with the lowest EPT. Since he is a type - D customer, ADARTW schedules him/her in such a way that his/her delivery time deviation is as small as possible, that is, zero. The resulting route and schedule are shown in Figure 6a and Table 8a respectively. Note that there is a single schedule block ("Block 1") in the schedule, which is pushed as late in time as possible to achieve zero delivery time deviation for customer 1. This necessitates a slack period of 3.59, during which the vehicle idles at 0. Up to this iteration, $Z_{ADARTW} = 0$.

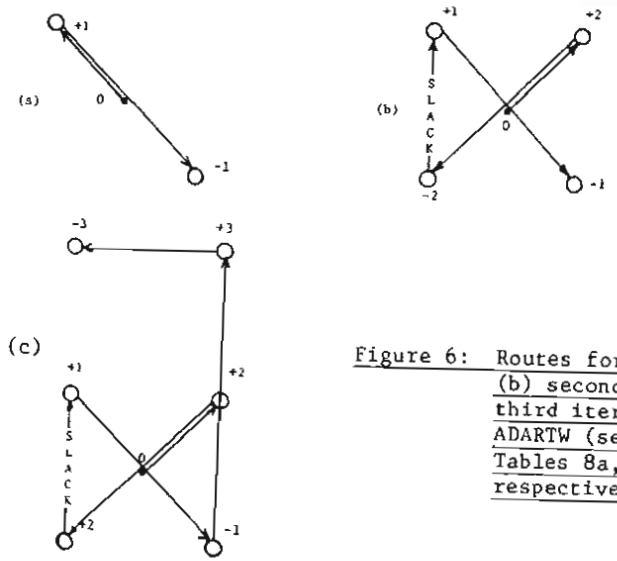

Figure 6: Routes for (a) first, (b) second, and (c) third iterations of ADARTW (see also Tables 8a, 8b, and 8c respectively).

TABLE 8a: Schedule at first iteration of ADARTW (also see Figure 6a)

Stop	Arrival Time	Departure Time	Idle Time	Desired Time	Time Deviation	Ride Time
0	0	3.59	3.59	-	-	-
+1	4.29	4.29	0	-	-	-
-1	5.71	-	-	5.71	0	1.41

ADARTW next considers customer 2. The best insertion results in a route and schedule shown in Figure 6b and Table 8b respectively. The schedule consists now of two blocks. Block 1 remains as before, that is, pushed as late in time as possible. The new block (which includes the pickup and delivery of customer 2) is Block 2, which is pushed as early in time as possible, to

TABLE 8b: Schedule at second iteration of ADARTW (also see Figure 6b)

Stop	Arrival Time	Departure Time	Idle Time	Desired Time	Time Deviation	Ride Time
0	0	0	0	-	-	-
+2	0.71	0.71	0	0.71	0	-
-2	2.12	3.29	1.17	-	-	1.41
+1	4.29	4.29	0	-	-	-
-1	5.71	-	-	5.71	0	1.41

achieve a zero pickup time deviation for customer 2 (who is of type P). There is a residual slack of 1.17 between these two blocks, representing the total amount of available idle time inbetween. As before, and up to this iteration, $Z_{ADARTW} = 0$.

Perhaps the most crucial observation at this point is that since the delivery of customer 2 already precedes the pickup of customer 1, there is no way that this relative order can be reversed in subsequent insertions. In other words, we can already see that it would be impossible for ADARTW to insert customer 3 so that it would eventually produce the optimal route and schedule of Figure 5 and Table 7. Still, the possibility of obtaining an alternate optimal route and schedule cannot be ruled out before all possible insertions of customer 3 are evaluated.

Unfortunately, there are not that many feasible insertions for customer 3, even if insertion into two different schedule blocks is attempted. Inserting both +3 and -3 into (including before or after) Block 2 is infeasible, primarily because customer 3's EPT is rather late (1.71) and because the slack between Blocks 1 and 2 is rather tight (1.17). The same is true

regarding an insertion into two different blocks. The only possible insertion seems to be the one immediately after Block 1 (see Figure 6c and Table 8c). Such an insertion would entail a pickup time deviation of 6 for customer 3, if the delivery time deviation of customer 1 is to remain at zero. This proves that for this example $Z_{ADARTW} = 6$ while $Z_{OPT} = 0$, that is, ADARTW is not successful in identifying the optimal solution.

TABLE 8c: Schedule at third (and final) iteration of ADARTW (also see Figure 6c).

Stop	Arrival Time	Departure Time	Idle Time	Desired Time	Time Deviation	Ride Time
0	0	0	0	-	-	-
+2	0.71	0.71	0	0.71	0	-
-2	2.12	3.29	1.17	-	-	1.41
+1	4.29	4.29	0	-	-	-
-1	5.71	5.71	0	5.71	0	1.41
+3	7.71	7.71	0	1.71	6	-
-3	8.71	-	-	-	-	1

ADARTW fails to produce the optimal route and schedule in this example for a number of reasons, due both to certain features of the instance itself and to some features of the algorithm. Regarding the instance itself, one could possibly identify the combination of loose service guarantees (W = 6, MRT = 4), coupled with a mix of type-P and type-D customers as conducive to the type of behavior displayed above. Regarding the algorithm, it is clear that the fact that the "pool size" K was set equal to 1, and the fact that ADARTW has no "resequencing" capability ultimately forced the algorithm to insert customer 3 after Block 1.

It can be seen also that the above two features can cause ADARTW to mistakenly declare a feasible problem instance as infeasible. Indeed, it is easy to check that such would be the outcome in the example examined if the width of the time window W were equal to 4 instead of 6. In such a case, the optimal route and schedule would remain unchanged, but ADARTW would be unable

to service customer 3 without an additional vehicle.

As a brief aside, one might wonder how would the GCR algorithm perform on such a "perverse" instance. That would depend, to a significant extent, on the values of the parameters DT and CF of that procedure. It is straightforward to check that if CF = 1.5, and if time groups of DT = 2 are set up starting at t = 0, the GCR algorithm would produce the optimal route and schedule for this example. Of course, other choices in DT and CF might make GCR fail to identify the optimal solution (an example for this case is DT = 3, CF = 1.0).

This analysis has been to a certain extent unfair to ADARTW, because the criterion that has been used (worst-case error ratio) is less than perfectly suited for this problem. One could actually imagine even more extreme cases, such as a 1,000-customer problem in which $Z_{OPT} = 0$ and in which ADARTW achieves a zero time deviation for 999 customers and a time deviation of 1 for one customer. It would be clearly nonsensical to state that for this case ADARTW performs infinitely poorly. In addition, the analysis is unfair because it refers to pathological cases which are unlikely to occur in the real world. Indeed, it is not uncommon for a heuristic whose worst-case performance is poor to behave decently in practice; perhaps the most typical example in this context is the k-interchange heuristic of Lin and Kernighan (1973) for the TSP, which, in spite of an arbitrarily poor worst-case behavior (Papadimitriou and Steiglitz (1978)), is known to be one of the best heuristics devised for the TSP to date. Our real-world computational experience with ADARTW, which has been very encouraging thus far, tends to confirm the disparity between worst-case and average-case behavior of the algorithm.

As far as the worst-case criterion used is concerned, we note here that there is considerable room for further analysis if ADARTW's performance is evaluated otherwise. Error ratios can still be used if the objective function coefficients are

restricted to certain values that would guarantee $Z_{OPT} > 0$. For instance, an interesting question is what happens, in terms of worst-case performance, if the only nonzero coefficient is c_5. That is, the issue is what is ADARTW's worst-case performance in terms of total vehicle active time (or, equivalently, total vehicle distance traveled). The worst that could happen here might be a case in which ADARTW is forced to use n vehicles for n customers traveling from the same origin to the same destination, whereas the optimal solution recommends only one vehicle for these customers. Thus far, we have been unable to construct such a pathological instance if the only nonzero coefficient is c_5.

Error ratios can also be used if they compare the heuristic error to the maximum possible error rather than the optimal value of Z. A number of researchers (see for instance Fisher (1980)) have suggested the use of the ratio $(Z_H - Z_{OPT})/(Z_R - Z_{OPT})$ in lieu of the traditional error ratio $(Z_H - Z_{OPT}/Z_{OPT})$ for minimization problems, where Z_R is a suitably chosen reference value. Ideally, Z_R should be the maximum value of Z. Unfortunately, this maximum value is usually no easier to identify than Z_{OPT}, so Z_R is instead taken to be some upper bound on the maximum value of Z that is presumably easier to obtain. In terms of the previous example ($c_1 = 1$ and all others zero), it can be seen that the maximum possible total time deviation is equal to the time window width multiplied by the total number of customers, that is, $Z_R = nW$ (in this case $Z_R = 18$). If this is the case, the solution error $Z_{ADARTW} - Z_{OPT} = 6$ seems quite reasonable as compared to the maximum possible error $Z_R - Z_{OPT} = 18$.

A final direction in the worst-case analysis of ADARTW is to abandon the use of error ratios altogether and focus on absolute errors instead (that is, use $Z_{ADARTW} - Z_{OPT}$ as the error criterion). The presence of hard constraints implies that upper bounds on the total customer deviation from desired (pickup or delivery) time, on the total excess ride time, and on the total vehicle active time are equal to nW, $\sum_{i=1}^{n} (MRT_i - DRT_i)$ and $\sum_{i=1}^{n} (D(depot, +i) + MRT_i)$ respectively, if no customers are to be rejected. However, these bounds are likely to be loose because they fail to take into account the nature of the algorithm and other problem inputs (such as vehicle capacity). Whether these (and possibly other) bounds can be tightened by exploiting the special structure of the problem and the algorithm remains to be seen.

5. CONCLUDING REMARKS

This paper has reviewed two heuristic algorithms developed by the author and his colleagues for the multi-vehicle advance-request Dial-A-Ride problem, and has summarized computational experience with the procedures to date. Both algorithms seem to be quite efficient computationally, and have solved the largest (to our knowledge) instances in this problem class to date. On the basis of this computational experience, we feel that both procedures can be useful in the implementation of an advance-request Dial-A-Ride system. We feel the GCR algorithm can be best used as a fast planning tool and/or as a device to produce good starting solutions in an operational situation, whereas the ADARTW algorithm can form the basis of an operational scheduling system that would assist the dispatcher in the actual execution of the schedule. Of course, additional refinements in both procedures, and continuing computational experience with them are necessary to both shed more light on their performance, and, ultimately, enhance that performance even more.

We end this paper by discussing several important issues on the scenarios under which the procedures of the paper are likely to be implemented.

We mentioned earlier that the issue of whether customer service guarantees (or, the corresponding time constraints) should be "hard" or "soft" in an advance-request environment is a debatable one. The "hard" approach is certainly more appealing

from a policy (or even a public relations) standpoint, plus, lends itself more easily to analysis by quantitative techniques. In addition, there are certainly cases in which a customer has to be picked up (or delivered) within a prescribed time window. However, the "hard" approach also has some drawbacks: First, it opens the door to "infeasible solutions" or to solutions which are erroneously declared as infeasible (as seen in Section 4). How would a dispatcher handle an "infeasible solution" in practice? Rejecting customers may not be an allowable alternative. Similarly, adding more vehicles to make the problem feasible may be far less implementable an option than the discussion thus far would seem to suggest. This is certainly true in a "mixed" demand scenario where the dispatcher might simply be unable to add vehicles upon request. Thus, it may happen that in such situations the best alternative for the dispatcher would be to make the solution feasible by relaxing the constraints. This could possibly involve rerunning the algorithm, and/or calling some or all of the customers to renegotiate new service parameters. Leaving aside for the moment the fact that a "hard" constraint that is subsequently relaxed is, by definition, "soft", one can also see that there are a multitude of other public perception problems that could occur under these (or similar) circumstances. This is particularly true if the (supposedly "hard") constraints under which the algorithm operates have been advertised as such in public.

Extending an advance-request dial-a-ride algorithm into the equivalent "mixed" demand case is a well-motivated task, since pure advance-request systems are either nonexistent or very rare. Such an extension would be much more difficult to implement in the GCR algorithm than in ADARTW. Indeed, the design of ADARTW would make the real-time insertion of immediate-request customers into an advance-request schedule obtained (say) the previous day, seem "straightforward". In fact, certain facets of the insertion problem become easier for immediate-request customers (for

instance, schedules can no longer be shifted earlier in time upon appearance of an immediate request, and that would alleviate some of the computational burden of the procedure). However, as alluded to above, there are a number of issues that merit serious attention before such an extension is implemented. Given that it might be very difficult, or even impossible, to add new vehicles so as to make the problem feasible each time an immediate request appears, one would likely have to adopt a different system of service guarantees for real-time requests. In fact, it is exactly because of such issues that Rufbus policy makers decided to consider time constraints as "soft" rather than "hard" (see the description in Section 3.3). Short of scrapping the idea of "hard" time constraints altogether, it might make sense to use it for advance-request customers only, and adopt softer service guarantees for real-time customers.

Finally, there are a number of other ideas that can be implemented to further enhance ADARTW. For instance, one could easily modify the algorithm to account for nonzero dwell times, or for specialized vehicles (for special categories of customers such as wheelchair customers, etc.). One could also easily add a post-optimization module such as k-interchange to provide a "resequencing" capability to the algorithm, or even a capability to "swap" customers among vehicles. Whether the last two measures would substantially enhance the performance of ADARTW is an open question at this point.

ACKNOWLEDGEMENTS

This work was supported in part by the Paratransit Integration Program of the Urban Mass Transportation Administration, U.S. Department of Transportation. The assistance of Ed Neigut, the project's technical monitor, on the formulation of the problem solved by ADARTW is gratefully acknowledged. Sincere thanks are also due to Horst Gerland of Rufbus, Wolfgang Kratschmer of Dornier, and others in their team for making their database

available, and for many fruitful discussions. Last, but not least, the author would like to thank his project colleagues, Jang-Jei Jaw, Amedeo Odoni, and Nigel Wilson, without whom this work would not have been completed, and Bruce Golden, an anonymous referee and the Editor for their comments.

REFERENCES

Bodin, L.D. and Sexton, T.R. (1982). The Multi-Vehicle Subscriber Dial-A-Ride Problem. Working Paper No. 82-005, Department of Management Science and Statistics, College of Business and Management, University of Maryland, College Park, Maryland 20742.

Fisher, M.L. (1980). Worst-Case Analysis of Heuristic Algorithms. Management Science 26, 1-16.

Golden, B., Bodin, L.D., Doyle, T., Stewart, W., Jr. (1980). Approximate Traveling Salesman Algorithms. Operations Research 28, 694-711.

Hung, H.K., Chapman, R.E., Hall, W.G., Neigut, E. (1982). A Heuristic Algorithm for Routing and Scheduling Dial-A-Ride Vehicles. ORSA/TIMS National Meeting, San Diego, California.

Jaw, J.J. (1984). Heuristic Algorithm for Multi-Vehicle, Advance-Request Dial-A-Ride Problems. Ph.D. Thesis, Department of Aeronautics and Astronautics, M.I.T., Cambridge, Massachusetts 02139.

Jaw, J.J., Odoni, A.R., Psaraftis, H.N., and Wilson, N.H.M. (1982). A Heuristic Algorithm for the Multi-Vehicle Many-to-Many Advance-Request Dial-A-Ride Problem. Working Paper MIT-UMTA-82-3, M.I.T., Cambridge, Massachusetts 02139.

Jaw, J.J., Odoni, A.R., Psaraftis, H.N., and Wilson, N.H.M. (1984). A Heuristic Algorithm for the Multi-Vehicle Advance-Request Dial-A-Ride Problem with Time Windows. Transportation Research (8), to appear.

Lin, S., Kernighan, B.W. (1973). An Effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Research 21, 498-516.

Papadimitriou, C.H., Steiglitz, K. (1978). Some Examples of Difficult Traveling Salesman Problems. Operations Research 26, 434-443.

Psaraftis, H.N. (1980). A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-A-Ride Problem. Transportation Science 14, 130-154.

Psaraftis, H.N. (1983a). An Exact Algorithm for the Single Vehicle Many-to-Many Dial-A-Ride Problem with Time Windows. Transportation Science 17, 351-357.

Psaraftis, H.N. (1983b). k-Interchange Procedures for Local Search in a Precedence-Constrained Routing Problem. European Journal of Operational Research 13, 391-402.

Psaraftis, H.N. (1983c). Analysis of an $O(N^2)$ Heuristic for the Single Vehicle Many-to-Many Euclidean Dial-A-Ride Problem. Transportation Research 17B, 133-145.

Rosenkrantz, D.J., Stearns, R.E., Lewis, P.M. (1974). Approximate Algorithms for the Traveling Salesperson Problem. Proceedings of the 15th Annual IEEE Symposium of Switching and Automata Theory, 33-42.

Roy, S., Chapleau, L., Ferland, J., Lapalme, G., and Rousseau, J.M. (1983). The Construction of Routes and Schedules for the Transportation of the Handicapped. Working Paper, Publication No. 308, Centre de Recherche sur les Transports, University of Montreal, Case Postale 6128 - Succursale A, Montreal, PQ, Canada H3C 3J7.

Sexton, T.R. (1979). The Single Vehicle Many-to-Many Routing and Scheduling Problem. Ph.D. Thesis, State University of New York at Stony Brook, New York 11738.

Solomon, M.M. (1983). On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints. Working Paper 84-02, Graduate School of Business Administration, Northeastern University, Boston, Massachusetts 02115.

Wilson, N.H.M., Weissberg, H. (1976). Advanced Dial-A-Ride Algorithms Research Project: Final Report. Report R76-20. Department of Civil Engineering, M.I.T. Report R-76-20, Cambridge, Massachusetts 02139.

Wilson, N.H.M., Colvin, N.H. (1977). Computer Control of the Rochester Dial-A-Ride System. Report R77-31. Department of Civil Engineering, M.I.T. Report R77-31, Cambridge, Massachusetts 02139.

Received 11/84; Revised 8/25/86.