A Decision Support System for the Bimodal Dial-A-Ride Problem

Ching-Fang Liaw, Chelsea C. White, III, *Fellow, IEEE*, and James Bander, *Member, IEEE*

***Abstract*— A bimodal dial-a-ride problem (BDARP) considered in this paper is a dial-a-ride problem that involves two transportation modes: paratransit vehicles and fixed route buses. Riders in such a system might be transferred between different transportation modes during the service process. The motivation of this research is that by efficiently coordinating paratransit vehicles with fixed route buses we can improve the accessibility and efficiency of a dial-a-ride system. In this paper, we design a decision support system (DSS) which automatically constructs efficient paratransit vehicle routes and schedules for the BDARP. This DSS has been tested using actual data from the Ann Arbor Transportation Authority (AATA) in Ann Arbor, MI. The results show that this DSS produces an average increase of 10% in the number of requests that can be accommodated and an average decrease of 10% in the number of paratransit vehicles required, as compared to the manual results where no fixed route buses are involved.**

I. Introduction

THE dial-a-ride system considered in this paper is concerned with routing and scheduling paratransit vehicles in order to satisfy requests from customers for transportation. Such demand-responsive transportation systems have been established in many localities. Typical target populations are people who are elderly or handicapped and who need transportation between their homes and special locations, such as hospitals, clinics, and work. Our focus is on individuals who are handicapped. Each request for service in such a system has a pickup point (origin), a delivery point (destination), and generally either a desired pickup time or a desired delivery time or both. Because of the implied precedence relationships (the pickup must precede the delivery) and the specified service time constraints, the development of effective vehicle routes and schedules in this type of situation can be complex (see [2], [3], [5], [6], and [12]).

The number of the elderly and handicapped using dial-a-ride systems is constantly increasing, placing pressure on such systems to improve their efficiency and, in part due to the Americans with Disabilities Act, accessibility. A dial-a-ride system is typically a highly subsidized program. Hence, operation cost reduction is also an important objective. Dial-a-ride systems are invariably more expensive per passenger mile and more convenient for the individual rider than a fixed route bus system. The intent of this research is to study how a dial-a-ride system can be efficiently integrated with a fixed route bus system so as to reduce total system cost and improve total system accessibility for the prospective rider without significant reduction in individual rider convenience.

The problem addressed will be called the bimodal dial-a-ride problem (BDARP), i.e., a dial-a-ride problem (DARP) that involves both paratransit vehicles and fixed route buses. Riders in such a system might be transferred between different transportation modes during the service process. So as not to significantly degrade rider convenience, we assume that each rider can experience at most two transfers during the entire trip from origin to destination, those being transfers from a paratransit vehicle to a fixed route bus or from a fixed route bus to a paratransit vehicle. Hence, no transfers are allowed between two vehicles of the same mode, e.g., between two paratransit vehicles or between two fixed route buses.

In this paper, we report on the development of a DSS for the automatic construction of paratransit vehicle routes and schedules for a BDARP.

II. Problem Description

Several versions of dial-a-ride service exist today, giving rise to several types of what we call the DARP. The version we are interested in is the advance-request DARP with desired delivery times. "Advance-request" means that all the requests for service are received well before the time of vehicle dispatching, say, one day in advance. A desired delivery time at the destination is specified by each request. For a given day, we have a set of requests for service, a paratransit vehicle fleet, a set of pieces of work (to be defined below) for paratransit vehicle drivers, and a bus system with fixed routes and schedules. A solution to the BDARP is a service configuration which combines paratransit vehicle rides with fixed route bus rides. In the following, the terms "solution" and "service configuration" are interchangeable. We seek a service configuration that maximizes the service capacity (the number of requests serviced per unit time) while satisfying the service time constraints specified by riders. For a fixed service capacity, we seek the minimum number of pieces of work needed and for these pieces of work, we seek routes and schedules for the paratransit vehicles that minimize the total travel distance.

A. Definitions and Assumptions

1) *Requests:* Each rider request involves: an origin, a destination, a given number of riders to be transported, and a desired delivery time. For each request i, the following data are assumed available:

Manuscript received July 20, 1994; revised December 23, 1995.

C.-F. Liaw is with the Department of Industrial Engineering, Chaoyang Institute of Technology, Taichung, Taiwan, ROC.

C. C. White, III and J. Bander are with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109-2117 USA (e-mail: ccwiii@umich.edu).

Publisher Item Identifier S 1083-4427(96)04762-5.

O_i: origin of request i;
D_i: destination of request i;
$WH(i)$: number of riders in wheelchairs of request i;
$RD(i)$: total number of riders of request i;
$DT(i)$: desired delivery time of request i.

2) *Pieces of Work:* A piece of work is part or all of a driver workday during which a driver and a paratransit vehicle are available to carry out transportation service. Hence, we have for each piece of work a departure point, an arrival point, and a work period. The number of pieces of work available and the assignment of drivers to vehicles is determined in advance. Paratransit vehicles have capacity limits on both the number of wheelchairs and the total number of riders. For each piece of work j, we have the following information:

$DP(j)$: the departure point for piece of work j;
$AP(j)$: the arrival point for piece of work j;
$ST(j)$: the starting time for piece of work j;
$FT(j)$: the finishing time for piece of work j;
$WC(j)$: wheelchair capacity for piece of work j;
$PC(j)$: passenger capacity for piece of work j.

3) *Distances and Times:* For paratransit vehicles, actual distances between the various stops are not known *a priori*, but can be computed using some minimum path algorithm, e.g., Dijkstra's algorithm or the A^* algorithm. However, estimates of these distances, e.g., Euclidean distance, are assumed available. We assume that the time for paratransit vehicles to travel from stop i to stop j is equal to the dwell time at stop i, $DW(i)$, plus the actual distance from stop i to stop j divided by the average speed of the paratransit vehicles (ASP). Define:

$D^*(i, j)$: the actual distance between stop i and stop j;
$D(i, j)$: the estimated distance between stop i and stop j;
$T^*(i, j)$: the actual travel time from stop i to stop j; $[T^*(i, j) = DW(i) + D^*(i, j)/ASP]$
$T(i, j)$: the estimated travel time from stop i to stop j.

Throughout this paper we assume that for any two stops i and j, the following two conditions hold: i) $D(i, j) \leq D^*(i, j)$ and ii) $T(i, j) \geq T^*(i, j)$. That is, $D(i, j)$ is an optimistic estimate of $D^*(i, j)$ while $T(i, j)$ is a pessimistic estimate of $T^*(i, j)$. In particular, we assume that $T(i, j) = DW(i) + D(i, j)/RASP$ where $RASP$ ($\leq ASP$) is a reduced average speed of the paratransit vehicles such that $D(i, j)/RASP \geq D^*(i, j)/ASP$. Furthermore, we assume that if stop i is a stop associated with request k, the dwell time $DW(i)$ is then given by $DW(i) = BS + WH(k) \times ATW + [RD(k) - WH(k)] \times ATP$, where BS = the basic setup time for a stop, ATW = the (averaged) time required for handling a rider in a wheelchair, and ATP = the (averaged) time required for handling a rider not in a wheelchair.

4) *Time Windows:* The transportation service provider specifies the constraints on the quality of service provided to riders. The quality of service for request i is usually specified by the following parameters: the maximum allowable deviation from the desired delivery time of request i (MD_i) and the maximum allowable excess riding time of request i (ME_i). That is, for each request i,

a) the desired delivery time—the actual delivery time $\leq MD_i$ and
b) the actual riding time—the direct riding time $\leq ME_i$,

where the direct riding time is the time needed to traverse a shortest path from the origin to the destination of request i. Note that we treat the desired delivery time as a "hard" constraint; that is, the actual delivery time for each request can not be later than its desired delivery time. We remark that the maximum allowable excess riding time (ME) can either be defined as a constant or a function of the direct riding time; e.g., $ME = 0.5 \times$ (the direct riding time).

We now introduce the time window concept, which allows us to deal with the service quality constraints. Each request involves two stops: an origin and a destination. For each of these stops, we define a time window. A time window associated with a stop is the time interval within which this stop must be visited. These time windows enable the constraints on the deviation from the desired delivery time to be satisfied, and they also help restrict excess riding time.

For each request i, we define:

$[L(O_i), U(O_i)]$: the time window associated with the origin of request i;
$[L(D_i), U(D_i)]$: the time window associated with the destination of request i.

We now show how these time windows are determined. Consider a request i with origin O_i, destination D_i, and desired delivery time $DT(i)$. Then,

$$\begin{aligned} U(D_i) &= DT(i); \\ L(D_i) &= U(D_i) - MD_i \\ &= DT(i) - MD_i; \\ U(O_i) &= U(D_i) - T^*(O_i, D_i) \\ &= DT(i) - T^*(O_i, D_i); \\ L(O_i) &= L(D_i) - T^*(O_i, D_i) - ME_i \\ &= DT(i) - MD_i - T^*(O_i, D_i) - ME_i; \end{aligned}$$

where $T^*(O_i, D_i)$ is the actual direct riding time from O_i to D_i. Since $T^*(O_i, D_i)$ is not known in advance, we use the estimate $T(O_i, D_i)$ to substitute for $T^*(O_i, D_i)$ in the formulas above. We remark that in this definition of time windows, the constraint on the maximum allowable excess riding time is slightly relaxed. More specifically, for request i the maximum allowable excess riding time is now increased from ME_i to $ME_i + MD_i + T(O_i, D_i) - T(O_i, D_i)$.

B. Mathematical Formulation

We now present a mathematical programming formulation for a simplified version of the bimodal dial-a-ride problem (BDARP), where the actual distances and travel times between various stops are assumed known. For simplicity, we assume only a single depot and a homogeneous fleet of paratransit vehicles. That is, all of the driver pieces of work have the same departure and arrival points (the depot), and all of the paratransit vehicles have the same capacity limits. Furthermore, we assume each request involves a single rider and a single wheelchair. In the following formulation, the service capacity is assumed to be fixed; i.e., the number of requests to be serviced is a constant. We minimize the number of pieces of work needed to service all of these requests, and for each piece of work, we find the route and schedule which minimizes the total travel distance.

Notice that each fixed bus route is serviced by several different buses (or route blocks) for a given day, each of which corresponds to a specific starting time. By a block of a fixed route, we mean a fixed route bus that begins its service on that route at a specific starting time. For ease of representation, we assume that the requests, the pieces of work, the fixed routes, the blocks of each fixed route, and the intermodal transfer points on each fixed route are all numbered in advance. Let:

$S = \{1, 2, \cdots, n\}$ is the set of requests, where n is the number of requests;

$R = \{1, 2, \cdots, r\}$ is the set of fixed bus routes, where r is the number of fixed routes;

$F(k) =$ the number of blocks of fixed route $k \in R$;

$E_j^k = \{E_j^k(1), E_j^k(2), \cdots, E_j^k(d_j^k)\}$ is the ordered set of intermodal transfer points on the jth block of fixed route $k \in R$, $1 \le j \le F(k)$; d_j^k is the number of intermodal transfer points on the jth block of fixed route $k \in R$; $E_j^k(m)$ is the mth intermodal transfer point on the jth block of fixed route $k \in R$, $1 \le m \le d_j^k$

$V = \{1, 2, \cdots, v\}$ is the set of pieces of work, where v is the number of pieces of work;

$WC = \{WC(1), \cdots, WC(v)\}$ = the wheelchair capacity for pieces of work;

$PC = \{PC(1), \cdots, PC(v)\}$ = the passenger capacity for pieces of work.

The following binary variables are defined to handle the connection between paratransit vehicle rides and fixed route bus rides. That is, these binary variables determine: 1) whether or not to utilize the fixed route bus system for each request for service and 2) if the decision is to utilize the fixed route bus system, the entry and exit points on the selected bus route. Define:

$$Z_i = \begin{cases} 1 & \text{if request } i \text{ is serviced with at least one transfer} \\ 0 & \text{Otherwise} \end{cases}$$

$$Y_{ik} = \begin{cases} 1 & \text{if request } i \text{ is serviced with transfers using fixed route bus } k \in R \\ 0 & \text{Otherwise} \end{cases}$$

$$U_{ijk}^m = \begin{cases} 1 & \text{if the } m\text{th intermodal transfer point on the } j\text{th block of fixed route } k \text{ is used as an entry point by request } i \\ 0 & \text{Otherwise} \end{cases}$$

$$V_{ijk}^m = \begin{cases} 1 & \text{if the } m\text{th intermodal transfer point on the } j\text{th block of fixed route } k \text{ is used as an exit point by request } i \\ 0 & \text{Otherwise.} \end{cases}$$

The relationships between these variables are as follows:

$$Z_i = 0 \Rightarrow \sum_{k=1}^{r} Y_{ik} = 0 \qquad \forall i \in S$$

$$Z_i = 1 \Rightarrow \sum_{k=1}^{r} Y_{ik} = 1 \qquad \forall i \in S$$

$$Y_{ik} = 0 \Rightarrow \begin{cases} \sum_{j=1}^{F(k)} \sum_{m=1}^{d_j^k} U_{ijk}^m = 0 \\ \sum_{j=1}^{F(k)} \sum_{m=1}^{d_j^k} V_{ijk}^m = 0 \end{cases} \qquad \forall i \in S; \quad \forall k \in R$$

$$Y_{ik} = 1 \Rightarrow \begin{cases} \sum_{j=1}^{F(k)} \sum_{m=1}^{d_j^k} U_{ijk}^m = 1 \\ \sum_{j=1}^{F(k)} \sum_{m=1}^{d_j^k} V_{ijk}^m = 1 \end{cases} \qquad \forall i \in S; \quad \forall k \in R$$

We now define variables which are used to handle the routing and scheduling issues of paratransit vehicles. Let:

$(O_i, D_i) =$ the origin and destination of request i;

$O^+ = \{O_i : i \in S\}$ = the set of origins;

$D^- = \{D_i : i \in S\}$ = the set of destinations;

$[L(O_i), U(O_i)] =$ the pickup time interval for request i;

$[L(D_i), U(D_i)] =$ the delivery time interval for request i;

$EN_i =$ the entry point of request i;

$EX_i =$ the exit point of request i;

$TB[E_j^k(m)] =$ the scheduled time at which the designated bus arrives at $E_j^k(m)$;

$W^1 = \{(O_i, D_i): i \in S, Z_i = 0\}$ = the set of origin $\rightarrow$ destination trips;

$W^2 = \{(O_i, EN_i): i \in S, Z_i = 1, O_i \neq EN_i\}$ = the set of origin $\rightarrow$ entry point trips;

$W^3 = \{(EX_i, D_i): i \in S, Z_i = 1, EX_i \neq D_i\}$ = the set of exit point $\rightarrow$ destination trips;

$M = W^1 \cup W^2 \cup W^3 = \{(p^+, p^-)\}$

= the set of all trips needed to be serviced by v pieces of work.

Note that

$$EN_i = \begin{cases} E_j^k(m) & \text{if } U_{ijk}^m = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

and

$$EX_i = \begin{cases} E_j^k(m) & \text{if } V_{ijk}^m = 1 \\ \emptyset & \text{otherwise} \end{cases}.$$

Each trip $p \in M$ requires transportation from an origin p^+ to a destination p^-, where an origin might actually be an exit point and a destination might actually be an entry point. We write $M^+ = \{p^+ \colon p \in M\}$ for the set of origins and $M^- = \{p^- \colon p \in M\}$ for the set of destinations. In fact,

$$\begin{aligned} M^+ = &\{O_i \colon i \in S,\, Z_i = 0\} \\ &\cup \{O_i \colon i \in S,\, Z_i = 1,\, O_i \neq EN_i\} \\ &\cup \{EX_i \colon i \in S,\, Z_i - 1,\, EX_i \neq D_i\}; \\ M^- = &\{D_i \colon i \in S,\, Z_i = 0\} \\ &\cup \{D_i \colon i \in S,\, Z_i = 1,\, EX_i \neq D_i\} \\ &\cup \{EN_i \colon i \in S,\, Z_i = 1,\, O_i \neq EN_i\}. \end{aligned}$$

In the formulation, we use a network structure $G = (N, A)$ which is defined as follows. The set of nodes in G is given by $N = \{0\} \cup M^+ \cup M^-$ where 0 denotes the depot. The set of arcs in G is given by $A = (\{0\} \times M^+) \cup I \cup (M^- \times \{0\})$ where $I \subseteq (M^+ \cup M^-) \times (M^+ \cup M^-)$ represents the set of arcs corresponding to "feasible" movements between origins and destinations. Let:

$t_0(l)$ = the departure time for piece of work l at the depot;

$t_1(l)$ = the arrival time for piece of work l at the depot;

$t(p)$ = the departure time at node p $\forall\, p \in M^+ \cup M^-$;

$L_t(p)$ = the number of total riders in the vehicle arriving at node p $\forall\, p \in M^+ \cup M^-$;

$L_w(p)$ = the number of riders in wheelchairs in the vehicle arriving at node p $\forall\, p \in M^+ \cup M^-$;

$D^*(p, q)$ = the distance from node p to node q $\forall\, p, q \in N$;

$T^*(p, q)$ = the travel time from node p to node q $\forall\, p, q \in N$;

C_{pq} = the cost of using arc (p, q) $\forall\, (p, q) \in A$;

The time windows associated with nodes in $M^+ \cup M^-$ are defined as follows:

For each node $p^+ \in M^+ \cap O^+$, i.e., $p^+ = O_i$ for some $i \in S$, the time window associated with it is $[L(p^+), U(p^+)] = [L(O_i), U(O_i)]$.

For each node $p^- \in M^- \cap D^-$, i.e., $p^- = D_i$ for some $i \in S$, the time window associated with it is $[L(p^-), U(p^-)] = [L(D_i), U(D_i)]$.

For each node $p^+ \in M^+ \backslash O^+$, i.e., $p^+ = EX_i$ for some $i \in S$, the time window associated with it is $[L(p^+), U(p^+)] = [TB(EX_i), \min\{TB(EX_i) + MWT, U(D_i) - T^*(EX_i, D_i)\}]$ where MWT is the maximum waiting time allowed at a intermodal transfer point.

For each node $p^- \in M^- \backslash D^-$, i.e., $p^- = EN_i$ for some $i \in S$, the time window associated with it is $[L(p^-), U(p^-)] = [\max\{L(O_i) + T^*(O_i, EN_i), TB(EN_i) - MWT\}, TB(EN_i)]$.

In this formulation, the flow variables (X_{pq}^l) and the objective function coefficients (C_{pq}) are defined as follows:

$$X_{pq}^l = \begin{cases} 1 & \text{if arc } (p, q) \text{ is used by piece of work } l \\ 0 & \text{otherwise} \end{cases}$$

$$C_{pq} = \begin{cases} \Delta + D^*(p, q) & \text{if } p = \{0\} \\ D^*(p, q) & \text{otherwise} \end{cases}$$

where Δ is a large fixed constant to ensure that the number of pieces of work is minimized.

In summary, the simplified BDARP can be described as the following linear mixed integer program:

$$\text{MIN} \sum_{l \in V} \sum_{(p, q) \in A} C_{pq} X_{pq}^l \tag{1}$$

S.T.

$$Z_i = 0 \Rightarrow \sum_{k=1}^{r} Y_{ik} = 0 \qquad \forall i \in S \tag{2}$$

$$Z_i = 1 \Rightarrow \sum_{k=1}^{r} Y_{ik} = 1 \qquad \forall i \in S \tag{3}$$

$$Y_{ik} = 0 \Rightarrow \begin{cases} \sum_{j=1}^{F(k)} \sum_{m=1}^{d_j^k} U_{ijk}^m = 0 \\ \sum_{j=1}^{F(k)} \sum_{m=1}^{d_j^k} V_{ijk}^m = 0 \end{cases} \qquad \forall i \in S; \quad \forall k \in R \tag{4}$$

$$Y_{ik} = 1 \Rightarrow \begin{cases} \sum_{j=1}^{F(k)} \sum_{m=1}^{d_j^k} U_{ijk}^m = 1 \\ \sum_{j=1}^{F(k)} \sum_{m=1}^{d_j^k} V_{ijk}^m = 1 \end{cases} \qquad \forall i \in S; \quad \forall k \in R \tag{5}$$

$$\sum_{l \in V} \sum_{q \in S} X_{pq}^l = 1 \qquad \forall p \in M^+ \tag{6}$$

$$\sum_{q \in S} X_{pq}^l - \sum_{q \in S} X_{qp}^l = 0 \qquad \forall p \in M^+ \cup M^-; \ \forall l \in V \tag{7}$$

$$\sum_{q \in S} X_{p^+q}^l - \sum_{q \in S} X_{qp^-}^l = 0 \qquad \forall p \in M; \ \forall l \in V \tag{8}$$

$$t(p^+) + T^*(p^+, p^-) \leq t(p^-) \qquad \forall p \in M \tag{9}$$

$$X_{pq}^l = 1 \Rightarrow t(p) + T^*(p, q) \leq t(q) \qquad \forall\, (p, q) \in A,\, p \neq \{0\},\, q \neq \{0\};\ \forall l \in V \tag{10}$$

$$X_{pq}^{l}=1 \Rightarrow t_0(l)+T^*(p,q) \le t(q) \quad \forall q \in M^+;\ \forall l \in V;\ p=\{0\} \tag{11}$$

$$X_{pq}^{l}=1 \Rightarrow t(p)+T^*(p,q) \le t_1(l) \quad \forall p \in M^-;\ \forall l \in V;\ q=\{0\} \tag{12}$$

$$L(p) \le t(p) \le U(p) \quad \forall p \in M^+ \cup M^- \tag{13}$$

$$ST(l) \le t_0(l) \le t_1(l) \le FT(l) \quad \forall l \in V; \tag{14}$$

$$X_{pq}^{l}=1 \Rightarrow L_t(p)-1 \le L_t(q) \quad \forall (p,q) \in A;\ \forall p \in M^-;\ \forall l \in V \tag{15}$$

$$X_{pq}^{l}=1 \Rightarrow L_w(p)-1 \le L_w(q) \quad \forall (p,q) \in A;\ \forall p \in M^-;\ \forall l \in V \tag{16}$$

$$X_{pq}^{l}=1 \Rightarrow L_t(p)+1 \le L_t(q) \quad \forall (p,q) \in A;\ \forall p \in M^+;\ \forall l \in V \tag{17}$$

$$X_{pq}^{l}=1 \Rightarrow L_w(p)+1 \le L_w(q) \quad \forall (p,q) \in A;\ \forall p \in M^+;\ \forall l \in V \tag{18}$$

$$0 \le L_t(p) \le PC-1 \quad \forall p \in M^+ \tag{19}$$

$$0 \le L_w(p) \le WC-1 \quad \forall p \in M^+ \tag{20}$$

$$\begin{aligned}
Z_i &= 0,1 && \forall i \in S\\
Y_{ik} &= 0,1 && \forall i \in S; \forall k \in R\\
U_{ijk}^{m} &= 0,1 && \forall i \in S;\ \forall k \in R; 1 \le j \le F(k);\ 1 \le m \le d_j^k\\
V_{ijk}^{m} &= 0,1 && \forall i \in S;\ \forall k \in R; 1 \le j \le F(k);\ 1 \le m \le d_j^k\\
X_{pq}^{l} &= 0,1 && \forall (p,q) \in A;\ \forall l \in V\\
L_t(p) &\ge 0, \text{ integer} && \forall p \in M^+ \cup M^-\\
L_w(p) &\ge 0, \text{ integer} && \forall p \in M^+ \cup M^-\\
t(p) &\ge 0 && \forall p \in M^+ \cup M^-.
\end{aligned}$$

Constraints (2)–(5) establish the relationships between the paratransit rides and the fixed route bus rides. Constraints (6) and (7) are the flow conservation constraints. Constraints (8) ensure that the origin and destination for each trip are visited by the same piece of work. Constraints (9) model the precedence relationship between origins and destinations. Constraints (10)–(14) describe the compatibility requirement between the routes and schedules, while constraints (15)–(20) guarantee no violation of the paratransit vehicle capacity.

III. Methodology

In this section, we present a real time DSS for the BDARP. This DSS consists of two subsystems: an on-line DSS and an off-line DSS. Consider the following operating scenario. Prior to the evening before the day that services are to occur, telephone information agents (TIA's) receive calls requesting service. (We remark that in reality there are may be a variety of communication media, e.g., kiosks, that can receive the request for service. For simplicity, we will focus on the TIA as the recipient for the request for service.) Each request specifies a pickup point, a delivery point and a desired delivery time. The on-line DSS is used to determine whether there exists a fixed route bus that can be used to help service the request. If yes, the on-line DSS identifies the possible entry and exit points on this selected bus route, and then schedules the associated paratransit vehicle trips. Scheduling paratransit vehicle trips involves determining pickup times at the pickup point and at the exit point of the selected bus route and identifying the pieces of work of paratransit vehicles at both ends (if necessary) of these trips. The customer on the phone is then told when he/she will be picked up, what fixed route bus he/she will take, when he/she will get off the bus in order to be picked up by the second paratransit vehicle. If a request cannot be accommodated, then that request is put on stand-by. One important feature of this on-line operation is that it insures the existence of a feasible solution to the BDARP.

We remark that the resulting feasible and initial solution is likely to be substantially suboptimal since it does not, and cannot, consider subsequent requests for service. Also, it is likely that there are feasible solutions in which fewer requests would be put on stand-by; i.e., the service capacity can be increased. Therefore, effort will be made to honor stand-by requests by the off-line operation, which is a procedure to improve the initial solution provided by the on-line DSS.

We now assume that there is no obligation to take further appointments for the following day. We then have the following information. For each piece of work, we have a list of (origin, destination, and time windows), where an origin might actually be an exit point out of a certain bus route and a destination and might be an entry point into a certain bus route. We then use a more numerically intense DSS, the off-line DSS, to identify a service configuration that is more preferred than the initial service configuration determined by the on-line DSS. This might involve changing the current entry and exit points, changing the pickup times at the origins, adding some or all of the stand-by requests to the service, and reassigning pieces of work to the generated trips. Those riders whose pick-up times are changed, and those riders who are removed from the stand-by list are then recontacted by the TIA's.

A. On-Line Decision Support System

Basically, for each request i, the on-line DSS performs the following three tasks.

Task 1: Determine the paratransit vehicle trips to service request i.
Find a *proper* fixed route bus that can be utilized to help service request i. If one such fixed route bus is found, identify the possible entry and exit points on this bus route; i.e., EN_i and EX_i, and then generate the necessary paratransit vehicle trips. In this case, we will generate at most two paratransit vehicle trips: $O_i \rightarrow EN_i$ and $EX_i \rightarrow D_i$. Otherwise, use a single paratransit vehicle trip, $O_i \rightarrow D_i$, to service request i.

Task 2: Determine the time windows for the paratransit vehicle trips generated in Task 1.

Task 3: Schedule the paratransit vehicle trips generated in Task 1.

We now explore the solution methods for attacking these three tasks. We remark that since actual distances and hence actual travel times between the various stops are not known, these three tasks are performed based on estimated distances and travel times between the various stops.

Task 1: Since our objective is to maximize service capacity and minimize the travel distance for paratransit vehicles, it seems reasonable that we would like to utilize the fixed route bus system as much as possible. (Here, we assume that the operating capacity of the fixed route bus system is far below its capacity limit, and hence the increased service will not violate any capacity constraint on the fixed route bus.) That is, if there exists a proper fixed route bus which can be used to help carry riders from their origin to their destination, we would like to put riders onto, and get them off of, this fixed route bus at certain intermodal transfer points so that the required paratransit vehicle trips are as short as possible, as long as the delivery time constraint is not violated. The criterion and approach for determining a proper fixed route bus to service a request i are outlined as follows.

Criterion: Choose a proper fixed route bus that will require the shortest paratransit vehicle trips to service request i. If the estimated total travel distance of the required paratransit vehicle trips $[D(O_i, EN_i) + D(EX_i, D_i)]$ is greater than a value which is a function of $D(O_i, D_i)$, e.g., $\theta \times D(O_i, D_i)$ where θ is a constant, then service this request with a single paratransit vehicle trip $O_i \rightarrow D_i$ (thus, no fixed route bus trip would be involved).

Approach: A simple procedure for determining a proper fixed route bus to service request i, assuming that intermodal transfer points, origins, and destinations are all geocoded, is given below:

1) Draw two small circles centered at O_i and D_i, respectively.
2) Enlarge these two circles until there is at least one bus route which passes through both circles and satisfies the following two conditions: i) The direction of the bus route is consistent with the direction from O_i to D_i. ii) At least one intermodal transfer point on this bus route is contained within each circle.
3) Choose one of these bus routes arbitrarily.

Once the fixed route bus is chosen, the entry point (EN_i) is selected arbitrarily from those intermodal transfer points contained within the circle centered at O_i. Similarly, the exit point (EX_i) is selected arbitrarily from those intermodal transfer points contained within the circle centered at D_i.

Task 2: If the output of Task 1 is a single origin $\rightarrow$ destination paratransit vehicle trip; i.e., no fixed route bus is involved, then the time windows for both origin and destination can be determined as shown in Section II-A. Otherwise, the time windows for an entry point and an exit point are determined as follows.

Consider a request i with an entry point EN_i and an exit point EX_i. Let $TB(EN_i)$ and $TB(EX_i)$ be the scheduled times at which the designated bus will arrive at EN_i and EX_i, respectively. We remark that the times $TB(EN_i)$ and $TB(EX_i)$ are unique since Task 1 not only chooses a proper fixed route bus but also specifies its starting time; i.e., the round number of the chosen fixed route bus is also determined in Task 1. Let MWT denote the maximum acceptable length of time for a rider to stay at a bus stop before he/she is picked up by a fixed route bus or a paratransit vehicle. Then the time window for EN_i is given by $[\text{Max}\,\{L(O_i) + T(O_i, EN_i), TB(EN_i) - MWT\}, TB(EN_i)]$. Similarly, the time window for EX_i is given by $[TB(EX_i), \text{Min}\,\{TB(EX_i) + MWT, U(D_i) - T(EX_i, D_i)\}]$. If either of these time windows is invalid (a time window $[a, b]$ is said to be invalid if $a > b$), then we have to redo Task 1 in order to find a new entry point, a new exit point, and/or a new fixed route bus.

Task 3: Task 3 includes assigning the required paratransit vehicle trips to pieces of work and determining new routes and schedules for those affected pieces of work. Hence, it is a trip insertion operation. In the following we present a simple procedure for trip insertion, similar to that proposed by Jaw *et al.* [6].

This procedure inserts one trip at a time into the current schedule for some piece of work. Central to this procedure is: a search for a feasible insertion of the trip into the current schedule of some piece of work. An insertion of a trip into the schedule of a piece of work is feasible only if the following two conditions are satisfied:

1) It does not lead to any violation of the capacity constraints specified for that piece of work.
2) It does not lead to any violation of the time window constraints for the trip and for all other trips already assigned to that piece of work.

The processing of a trip i goes as follows. For each piece of work j, determine $DIST_j$, the minimum additional travel distance incurred when trip i is inserted into the current schedule of piece of work j. If it is infeasible to insert trip i into the current schedule of piece or work j, let $DIST_j = +\infty$. If at least one feasible insertion is found, then insert trip t into the current schedule of piece of work j^* for which $DIST_{j^*} \leq DIST_j$ for all j. Otherwise, declare trip i a "rejected trip." (Note that for those requests resulting in rejected trips we can either put them on stand-by or redo Task 1 in order to try to determine new entry and/or exit points and hence generate new paratransit trips.)

We now discuss how to identify a feasible insertion of a trip into the current schedule of a piece of work. For each trip t, let:

1) "$+t$" and "$-t$" represent the departure point and arrival point of trip t, respectively;
2) $[L(+t), U(+t)]$ and $[L(-t), U(-t)]$ be the time windows associated with points $+t$ and $-t$, respectively;
3) $SH(+t)$ and $SH(-t)$ be the scheduled times at which the designated paratransit vehicle will visit points $+t$ and $-t$, respectively.

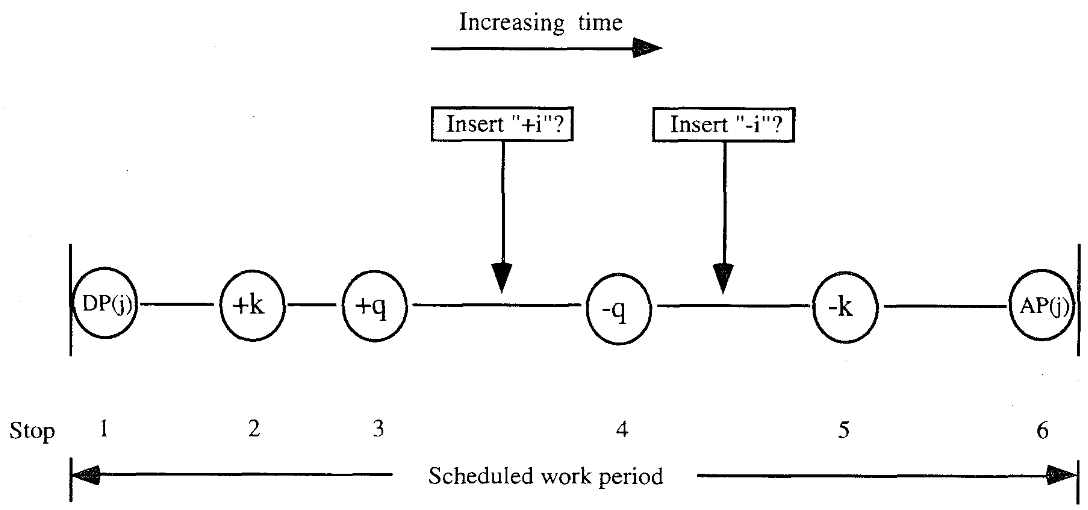

Fig. 1. Insertion of a trip i into an existing vehicle schedule.

For each trip t, a set of necessary and sufficient conditions for feasibility is then given by

$$L(+t) \leq SH(+t) \leq U(+t)$$

and

$$L(-t) \leq SH(-t) \leq U(-t).$$

Let us consider the problem of searching for a feasible way in which trip i can be inserted into the current schedule of piece of work j. We adopt the convention that the departure point of piece of work j, $DP(j)$, and the arrival point of piece of work j, $AP(j)$, are, respectively, the first and last stops on the current schedule of piece of work j. Also, the time windows associated with stops $DP(j)$ and $AP(j)$ are assumed to be $\{ST(j),\, FT(j) - T[DP(j),\, AP(j)]\}$ and $\{ST(j) + T[DP(j),\, AP(j)],\, FT(j)\}$, respectively. Suppose that there are p stops on the current schedule of piece of work j and that we number these successive stops in order from 1 to p. [Stop 1 is $DP(j)$ and stop p is $AP(j)$.] For convenience let us now drop the indication of whether a particular stop on the current schedule of piece of work j is a pick-up or a delivery and use $L(t)$, $SH(t)$, and $U(t)$ to represent the earliest, scheduled, and latest visit time, respectively, for stop t, $1 \leq t \leq p$. The scheduled visit times $SH(t)$ are defined as follows:

$$SH(t+1) = \text{Max}\,[SH(t) + T(t,\, t+1),\, L(t+1)], \quad 1 \leq t \leq p-1$$

where $T(t,\, t+1)$ is the estimated travel time from stop t to stop $t+1$. The initial value of $SH(1)$ can be set equal to any number between $L(1)$ and $U(1)$. For each stop t on the current schedule of piece of work j, we compute the following four quantities:

$$SH(t+1) = \text{Max}\,[SH(t) + T(t,\, t+1),\, L(t+1)], \quad 1 \leq t \leq p-1$$

where$T(t,\, t+1)$ is the estimated travel time from stop$t$ to stop $t+1$. The initial value of$SH(1)$ can be set equal to any number between$L(1)$ and $U(1)$. For each stop t on the current schedule of piece of work j, we compute the following four quantities:

1) $$BUP(t) = \underset{1\leq k\leq t}{\text{Min}}\,[SH(k) - L(k)];$$

2) $$BDOWN(t) = \text{Min}\,(U(1) - SH(1),\, \underset{2\leq k\leq t}{\text{Min}}\,\{U(k) - \text{Min}\,[SH(k-1) + T(k-1,\, k),\, SH(k)]\})$$

3) $$AUP(t) = \underset{t\leq k\leq p}{\text{Min}}\,[SH(k) - L(k)];$$

4) $$ADOWN(t) = \underset{t\leq k\leq p}{\text{Min}}\,\{U(k) - \text{Min}\,[SH(k-1) + T(k-1,\, k),\, SH(k)]\}.$$

(We assume that piece of work j is not in operation during the intervals $[ST(j),\, SH(1)]$ and $[SH(p),\, FT(j)]$.)

As pointed out by Jaw *et al.* [6], the real intuitive meanings associated with the four quantities defined above are as follows: $BUP(t)[BDOWN(t)]$ represents the maximum amount of time by which every stop preceding but not including stop $t+1$ can be advanced [delayed] without violating the time window constraints. Similarly, $AUP(t)[ADOWN(t)]$ represents the maximum amount of time by which every stop following but not including stop $t-1$ can be advanced [delayed]. Basically, the quantities BUP, $BDOWN$, AUP, and $ADOWN$ indicate by how much, at most, each segment of the scheduled work period can be displaced in order to accommodate an additional stop. As an example, in Fig. 1 (where an attempt is made to insert the pick-up of trip i between the third and the fourth stop of the current schedule assuming that the capacity constraints are not violated), the extra time required to visit $+i$ between $+q$ and $-q$ is given by $EXTRA = T(+q,\, +i) + T(+i,\, -q) - T(+q,\, -q)$. If $EXTRA \leq BUP(3) + ADOWN(4)$, then it is feasible to insert "$+i$" into the current schedule at the point indicated without violating any time window constraints for those trips already in the current schedule (trips q and m). If this is the case, then we have to determine the new scheduled visit times for the stops on the current schedule. Suppose that $BUP^*[\leq BUP(3)]$ is the amount of time by which stops 1, 2, and 3 [i.e., stops $DP(j)$, $+k$, and $+q$] are advanced. Similarly, suppose that $ADOWN^*[\leq ADOWN(4)]$ is the amount of time by which stops 4, 5, and 6 [i.e., stops $-q$, $-k$, and $AP(j)$] are delayed. To maintain feasibility, we must have $EXTRA \leq BUP^* + ADOWN^*$. Then, the scheduled visit times are updated as follows:

1) new $SH(t) = \text{old}\, SH(t) - BUP^*$ for $t = 1, 2, 3$;
2) new $SH(t) = \text{old}\, SH(t) + ADOWN^*$ for $t = 4, 5, 6$;
3) $SH(+i) = \text{Max}\,[\text{new}\, SH(+q) + T(+q, +i),\ L(+i)]$.

If the insertion of "$+i$" is feasible; i.e., $L(+i) \leq SH(+i) \leq U(+i)$, it is then necessary to check whether it is also feasible to insert "$-i$" at the point indicated in Fig. 1, using the new scheduled visit times resulting from the insertion of "$+i$." This second check is performed exactly the same way as the check of the insertion of "$+i$."

B. Off-Line Decision Support System

Because of the combinatorial complexity of the BDARP, it may take a possibly prohibitive amount of computational time to find a globally optimal solution for large problems. Hence, the solution strategy adopted for the off-line DSS is to find an approximate solution in an acceptable amount of computation time, using an approximation algorithm. From the system optimization point of view, the on-line DSS has generated a feasible initial solution (or service configuration) to the BDARP. Given this initial service configuration, the off-line DSS will use an iterative improvement procedure, based on simulated annealing, in order to attempt to determine a more efficient service configuration. Also, after reconstructing the service configuration in a more efficient way, the off-line DSS will try to increase service capacity by putting those requests on stand-by into service.

The solution procedure for the off-line DSS is outlined in Fig. 2. The core single vehicle dial-a-ride algorithm used in Step 2 of this procedure will be presented in Section IV.

To apply an iterative improvement procedure, we first need to define a service configuration, a cost function, and a generation mechanism. A cost function assigns a real number to each service configuration, and a generation mechanism is a device to generate a transition from a service configuration to another one by a small perturbation. The generation mechanism defines a neighborhood $NH(s_i)$ for each service configuration s_i, which consists of all service configurations that can be reached from s_i in one transition. The iterative improvement procedure can now be described as follows. Starting with a given service configuration, a sequence of iterations is generated, where each iteration consists of a possible transition from the current service configuration to a service configuration selected from the neighborhood of the current service configuration. If this neighboring service configuration has a lower cost measure, the current service configuration is replaced by this neighbor. Otherwise, another neighbor is selected and compared for its cost measure. The procedure terminates when a service configuration is obtained whose cost is no worse than any of its neighbors or when its execution time reaches a certain pre-specified limit.

Step 1: Determine an initial feasible service configuration. This includes determining the possible fixed bus route (and hence entry and exit points) to be used in serving each request, generating the associated paratransit vehicle trips, and then allocating these paratransit vehicle trips to available pieces of work. (This step is supported by the on-line DSS.)

Step 2: Compute the cost measure for the initial feasible service configuration. That is, find the optimal route and schedule for each piece or work using the single vehicle dial-a-ride algorithm, and then determine the total cost measure of these pieces of work. (This step is explored in Section IV.)

Step 3: Attempt to find a new feasible service configuration with reduced cost measure. That is, attempt to reconstruct the service configuration so that total cost measure is decreased. When a new service configuration is found with reduced total cost measure, perform the reconstruction forming new service configuration, and continue Step 3. (This step is investigated later in this section.) Otherwise, go to Step 4.

Step 4: Apply the on-line operation to those requests on stand-by. That is, for each request on stand-by, determine the required paratransit vehicle trips and attempt to insert them into the current schedules of pieces of work. (The on-line operation was discussed in Section III.A.)

Fig. 2. The solution procedure for the off-line DSS.

The formal definitions of a service configuration, a cost function, and a generation mechanism for the BDARP under consideration are given as follows.

1) *Service Configuration:* We associate each request i with the following data: $[BR_i, RN_i, EN_i, EX_i, VO_i, VD_i]$ where BR_i = the fixed route bus that is used in the service of request i, RN_i = the round number associated with bus BR_i, EN_i = the entry point of request i, EX_i = the exit point of request i, VO_i = the piece of work designated for the trip $O_i \rightarrow EN_i$, and VD_i = the piece of work designated for the trip $EX_i \rightarrow D_i$. We call such data a "string." For example, if request i is associated with a string [5 2 3 6 2 3], it means that a) the 2nd round of the 5th fixed route bus is used in the service of request i, b) the 3rd intermodal transfer point on this bus route is used as an entry point, c) the 6th intermodal transfer point on this bus route is used as an exit point, d) the 2nd piece of work is designated for the trip $O_i \rightarrow EN_i$, and e) the 3rd piece of work is designated for the trip $EX_i \rightarrow D_i$. If the service of request i does not involve any fixed route bus, the string associated with request i will take the form $[0\,0\,0\,0\,j\,j]$, where j is the number of the piece of work designated for the trip $O_i \rightarrow D_i$. A service configuration is a set of strings, each of which corresponds to a request.
2) *Cost Function:* Given a service configuration, each piece of work has associated with it a list of trips to be serviced. The total cost measure of this service configuration is the sum of the cost measures of all pieces of work. The cost measure of a piece of work j

is defined as follows:

$$fixed\,cost \times \sigma_j + unit\,cost \times d_j$$

where

$$\sigma_j = \begin{cases} 1 & \text{if at least one trip is assigned} \\ & \text{to piece of work } j \\ 0 & \text{otherwise} \end{cases}$$

and d_j = the total distance traveled by piece of work j to service assigned trips. Note that to minimize the total number of pieces of work used, we define the fixed cost of a piece of work to be a huge number. If a service configuration is infeasible (e.g., there exists one piece of work which cannot finish the assigned trips within its work period), then its cost is assumed to be ∞.

3) *Generation Mechanism:* A service configuration consists of a set of strings, where each string contains six elements. A transition from a service configuration to another service configuration is made by changing the value of one or more of these elements. For example, consider a service configuration with a single string [7 3 2 6 4 1]. Two new service configurations which can be reached from this service configuration in one transition are [7 3 3 6 4 1] and [7 3 2 6 4 2]. The first one is formed by changing the entry point and the second one is formed by changing the piece of work designated for the exit point $\rightarrow$ destination trip.

In the off-line DSS, the technique chosen for performing the iterative improvement procedure is simulated annealing (SA) ([7], [8]). The SA algorithm generates new service configurations with some probabilistic rules and either accepts or rejects them depending on their relative cost measure. SA sometimes accept service configurations that increase cost measure if the result of a certain random acceptance rule is positive. The probability of acceptance is controlled by a so-called temperature parameter, TP.

Let C denote the cost function. Given an initial service configuration, s_1, with cost measure $C(s_1)$, and an initial temperature $TP = TP_1$, the SA algorithm changes the service configuration and the temperature with each iteration, $t = 1, 2, \cdots$, as follows: A *generate* function chooses a candidate service configuration to which to make transition from the current service configuration, s_t, an *accept* function determines whether the candidate should be accepted, and an *update* function changes the temperature from TP_t to TP_{t+1}.

We now define formally these three functions for the BDARP.

1) *Generate Function:* Let S be the set of requests. Initially, a request i is chosen from S randomly. Then an random integer k between 0 and 4 is selected to determine the type of the perturbation to be made to the current service configuration. Specifically,

a) if $k = 0$, BR_i and hence NR_i are changed;
b) if $k = 1$, EN_i is changed;
c) if $k = 2$, EX_i is changed;
d) if $k = 3$, VO_i is changed;
e) if $k = 4$, VD_i is changed.

Each of these changes is made by replacing the old value with another randomly selected value.

2) *Accept Function:* The most commonly used accept function for SA is the one proposed by Kirkpatrick *et al.* [7]. This accept function is based on the well known Boltzmann's probability density distribution. If the cost measure of the new service configuration, $C(s_{t+1})$, is less than the cost measure of the old service configuration, $C(s_t)$, the new service configuration is accepted with probability one. Otherwise, it is accepted with probability:

$$\exp\left[-\frac{C(s_{t+1}) - C(s_t)}{TP_t}\right].$$

3) *Update Function:* The update function defines the "cooling process" which reduces the "temperature" gradually. The function is defined as follows: $TP_{t+1} = \alpha TP_t$ where $0 < \alpha < 1$. We have typically chosen α to be in the range [0.50, 0.95].

IV. Single Vehicle Dial-a-Ride Problem

The problem to be considered in this section is the time-constrained single vehicle dial-a-ride problem with unknown but computable traveling distances. That is, we have a set of trips, each of which has an origin, a destination, a given number of wheelchairs to be carried, and a given number of riders to be carried. Time windows are specified at both origin and destination for each trip. A single vehicle with given starting location, finishing location, work period, and capacity limits on both number of wheelchairs and number of total riders, is available to service these trips. We seek a route for this vehicle that minimizes the total travel distance needed to service all of the trips, while satisfying the specified time window constraints. The distance for traveling from one stop to another is unknown *a priori*, although can be computed using some minimum path algorithm, e.g., Dijkstra's algorithm or the A^* algorithm.

If we compute all the travel distances between any two stops in advance, then this problem reduces to the normal single vehicle dial-a-ride problem and hence can be solved by using those existing methods ([1], [4], [11], [13], and [14]). However, this may be computationally inefficient. We develop a heuristic search algorithm to solve this problem with the intent of reducing the required computational effort.

Our approach is based on the heuristic search algorithm BA^*, as presented in [9], for finding a minimum path in an OR-graph that requires arc-cost determination. The performance criterion is the total travel distance, which we will refer to as "cost." In the following, we are going to use the term cost instead of distance. Also, we assume that even though the vehicle can arrive at a stop at time $t' < t''$ where t'' is the earliest time at which this stop can be visited, service at this stop cannot begin until time t''.

The first step in our approach is to construct an *enumeration tree* (an OR-tree) for the single vehicle dial-a-ride problem. This defines the implicit graph or the state-space graph for the following search algorithm. The enumeration tree is used

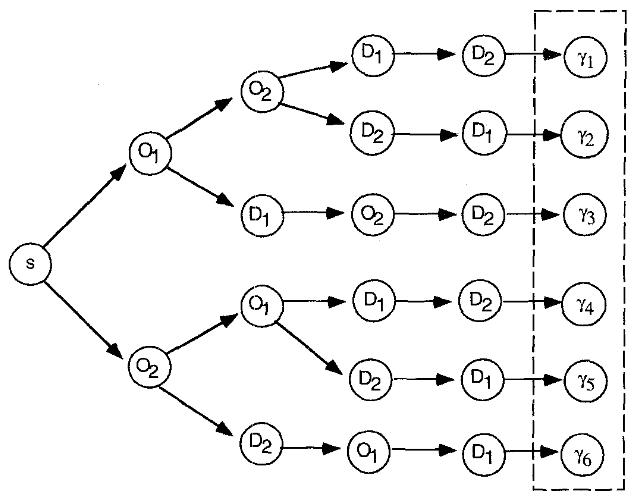

Fig. 3. The enumeration tree of a single vehicle dial-a-ride problem with two trips.

to explicate the precedence relation; that is, the origin must be visited before the destination for each rider. For example, the enumeration tree for a single vehicle dial-a-ride problem with two trips is shown in Fig. 3, where O_i and D_i are the origin and destination of trip i, $i = 1, 2$. The start node s represents the starting location of the vehicle and $\{\gamma_1, \gamma_2, \cdots, \gamma_6\}$ are the dummy goal nodes representing the finishing location of the vehicle. Every solution path in this enumeration tree corresponds to a vehicle route which visits all stops and satisfies the precedence relation. Notice that all of the solution paths in this enumeration tree have the same depth. This feature will be exploited to define an informative heuristic function, to be discussed in Section IV-D.

A. Notations and Definitions

Suppose there are m trips to be serviced. Let $M = \{O_1 \rightarrow D_1, O_2 \rightarrow D_2, \cdots, O_m \rightarrow D_m\}$ be the set of all trips where O_i and D_i are the origin and destination of trip i, $i = 1, 2, \cdots, m$. For each trip i, let:

$[L_0(i), U_0(i)]$ = the time window specified at the origin of trip i;

$[L_1(i), U_1(i)]$ = the time window specified at the destination of trip i;

$X(i)$ = the number of wheelchairs associated with trip i;

$Y(i)$ = the number of riders associated with trip i.

Let:

Q_w = the vehicle's wheelchair capacity;

Q_t = the vehicle's rider capacity;

$[T_0, T_1]$ = the work period of the vehicle;

Z_0 = the starting location of the vehicle;

Z_1 = the finishing location of the vehicle.

Assume the original street network is represented by an OR-graph $G = (N, A, s, \gamma)$ where N = the set of nodes in G, A = the set of arcs in G, $s \in N$ = the start node in G, representing the starting location of the vehicle, Z_0, and $\gamma \in N$ = the goal node in G, representing the finishing location of the vehicle, Z_1. Let $SO \subseteq N = \{O_1, O_2, \cdots, O_m\}$ be the set of origin nodes in G, $SD \subseteq N = \{D_1, D_2, \cdots, D_m\}$ be the set of destination nodes in G, and $\mathbf{I} = SO \cup SD = \{O_1, D_1, O_2, D_2, \cdots, O_m, D_m\}$ be the set of origin and destination nodes in G.

Let $G' = (N', A', s', \Gamma')$ represent the enumeration tree of this problem; that is, $s' = s$ = the start node in G', Γ' = the set of (dummy) goal nodes in G', $N' = \{s'\} \cup \mathbf{I} \cup \Gamma'$ = the set of nodes in G' and A' = the set of arcs in G'.

For each node n in G', let:

P_n = the (unique) path from s' to n in G';

$NP(P_n)$ = the set of nodes on path P_n;

$g(n)$ = the cost of the unique path from s' to n; i.e., the cost of path P_n;

$TR(n)$ = the corresponding trip number of node n; i.e., $TR(n) = i$ if node n corresponds to O_i or D_i. We define $TR(n) = 0$ if $n \in \{s'\} \cup \Gamma'$.

$[L(n), U(n)]$ = the associated time window; that is,

$$[L(n), U(n)] = \begin{cases} [T_0, T_1] & \text{if } n = s' \\ [T_0, T_1] & \text{if } n \in \Gamma' \\ \{L_0[TR(n)], U_0[TR(n)]\} & \text{if } n \in SO \\ \{L_1[TR(n)], U_1[TR(n)]\} & \text{if } n \in SD \end{cases}$$

$T(n)$ = the time the vehicle arrives at node n;

$q_w(n)$ = the number of wheelchairs in the vehicle when it leaves node n;

$q_t(n)$ = the number of riders in vehicle when it leaves node n.

B. Assumptions

Let c^*, c, and t be three functions defined on each pair of nodes in N. For each pair of nodes (n, n') in N, assume:

1) $c^*(n, n')$ is the cost of traversing a minimum path from node n to node n'. For example, $c^*(D_1, O_2)$ is the cost for traversing a minimum path from D_1 to O_2. We assume that the triangle inequality holds. Hence, if (n, n') is an arc in A, then $c^*(n, n')$ is the cost of traversing arc (n, n').
2) $c(n, n')$ is an *admissible* estimate for $c^*(n, n')$. That is, $c(n, n') \leq c^*(n, n')$ is an estimated cost for traversing a minimum path from node n to node n'. Thus, for each pair of nodes (n, n') in G', if $n' \in \Gamma'$, then we have $c(n, n') = c(n, \gamma)$ since the goal nodes in G' represent node γ in G, i.e., the finishing location of the vehicle. Furthermore, in the execution of arc-cost determination for arc (k, k') in G', $h(n) = c(n, k')$

for some intermediate node n between node k and node k', where $h(n)$ = the estimated cost for traversing a shortest path from node n to node k' in the original street network G.

3) $t(n, n')$ represents the estimated time needed for the vehicle to traverse a minimum path from node n to node n'. We define $t(n, n')$ as a function of $c(n, n')$; i.e., there exists a real-valued function Ψ such that $t(n, n') = \Psi[c(n, n')]$.

C. Modified BA^* Algorithm

We now present the modified version of the BA^* algorithm for finding a minimum cost path in the enumeration tree which satisfies the time window constraints. Since the enumeration tree is an OR-tree; i.e., each node in the tree has a unique parent, there is always a unique pointer path associated with each node in the explicit graph. Hence, the step of determining the minimum cost pointer path for each node in the explicit graph is no longer necessary, and this significantly reduces the complexity of the BA^* algorithm.

The most important feature of this modified BA^* algorithm is the node elimination operation. This node elimination operation helps limit the number of nodes generated during the search process. Hence, it not only accelerates the search process but also reduces the memory storage required by the search algorithm.

These node elimination criteria are derived from the time window constraints, the precedence constraints, and the vehicle capacity constraints. For each node n in the graph G', let $\mathbf{I}'(n) = \mathbf{I}\backslash NP(P_n)$ where "$\backslash$" is the set difference operator. That is, $\mathbf{I}'(n)$ is the set of nodes in $\mathbf{I}$ but not in $NP(P_n)$. A node n in some explicit graph of G' can be eliminated if one or more of the following criteria is satisfied:

Criterion 1: The time window constraint is violated. That is, $T(n) < L(n)$ or $T(n) > U(n)$.

Criterion 2: The vehicle capacity constraint is violated. That is, $q_w(n) > Q_w$ or $q_t(n) > Q_t$.

The following criteria are concerned with the spatial and temporal difference between node n and the nodes in the set $\mathbf{I}'(n)$. Note that for each node $n' \in \mathbf{I}'(n)$, the earliest time it can be visited is $\max\{L(n'), T(n) + t(n, n')\}$.

Criterion 3: There exists a node $n' \in \mathbf{I}'(n)$ such that $T(n) + t(n, n') > U(n')$.

Criterion 4 There exist two distinct nodes n_1 and n_2 in $\mathbf{I}'(n)$ such that $\max\{L(n_1), T(n) + t(n, n_1)\} + t(n_1, n_2) > U(n_2)$ and $\max\{L(n_2), T(n) + t(n, n_2)\} + t(n_2, n_1) > U(n_1)$.

Criterion 5 There exist three distinct node n_1, n_2, and n_3 in $\mathbf{I}'(n)$ such that $\max\{L(n_2), \max\{L(n_1), T(n) + t(n, n_1)\} + t(n_1, n_2)\} + t(n_2, n_3) > U(n_3)$ and $\max\{L(n_3), \max\{L(n_1), T(n) + t(n, n_1)\} + t(n_1, n_3)\} + t(n_3, n_2) > U(n_2)$.

The modified BA^* algorithm is given in Fig. 4. It can be seen that this algorithm possesses all of the properties that have been shown to hold for the BA^* algorithm.

```
Step 1.  Put s' on OPEN. Set $T(s')=T_0$, $q_w(s')=q_t(s')=0$.
Step 2.  If OPEN is empty, then exit with failure.
         If all of the arcs on $P_n$ have had their actual costs
         determined and $n \in \Gamma'$, then exit successfully with the
         solution equal to $P_n$.
Step 3.  Select n in OPEN such that $f(n) \le f(n')$ for all n' in OPEN.
         If all of the arcs on $P_n$ have had their actual costs
         determined and $n \notin \Gamma'$, do the following:
         (1) Remove n from OPEN and place it on CLOSED.
         (2) Set $\mathbf{I}'(n)=\mathbf{I}\backslash NP(P_n)$.
         (3) Go to step 5.
Step 4.  Otherwise, determine that actual cost of arc (n',n) which
         is the last arc in $P_n$. Set $c(n',n)=c^*(n',n)$. Recalculate
         $g(n)=g(n')+c(n',n)$ and $T(n)=\max\{L(n),T(n')+t(n',n)\}$. If one or
         more of the node elimination criteria is satisfied by node n,
         remove n from OPEN and place it on CLOSED. Otherwise,
         recalculate $f(n)=g(n)+h(n)$. Go to step 2.
Step 5.  Expand n, generating all of its successors, SCS(n), and
         attach to them pointers back to n. For every successor n" of
         n, do the following:
         (1) Set $P_{n''}=P_n \cup (n,n'')$, $g(n'')=g(n)+c(n,n'')$ and
             $T(n'')=\mathrm{Max}\{L(n''), T(n)+t(n,n'')\}$.
         (2) Set $q_w(n'') = \begin{cases} q_w(n)+X(TR(n'')) & \text{if } n'' \in SO \\ q_w(n)-X(TR(n'')) & \text{if } n'' \in SD \\ q_w(n) & \text{if } n'' \in \Gamma' \end{cases}$  and
             Set $q_t(n'') = \begin{cases} q_t(n)+Y(TR(n'')) & \text{if } n'' \in SO \\ q_t(n)-Y(TR(n'')) & \text{if } n'' \in SD \\ q_t(n) & \text{if } n'' \in \Gamma' \end{cases}$
         (3) If one or more of the node elimination criteria is
             satisfied by node n", place n" on CLOSED. Otherwise,
             estimate h(n"), calculate $f(n'')= g(n'')+h(n'')$ where $g(s')=0$,
             and place n" on OPEN.
Step 6.  Go to step 2.
```

Fig. 4. The modified BA^* algorithm.

D. Minimum Cost Spanning Tree Heuristic Estimates

As shown in [9], the modified BA^* algorithm will return a minimum cost path if the heuristic function h is admissible; i.e., $h(n) \leq h^*(n)$ for every node n in G' where $h^*(n) = c^*(n, \gamma)$. One such admissible heuristic function is $h(n) = c(n, \gamma)$ for every node n in G'; that is, let $h(n)$ be the estimated cost for traversing a minimum path from n to γ. However, this definition of the heuristic function does not exploit the knowledge we have about the special structure of all solution paths, which is that every solution path in G' must visit all of the nodes in the set $\mathbf{I}$. In this section, we utilize this special structure to develop a more informative heuristic function and hence to improve the search efficiency.

For each node n in G', $\mathbf{I}'(n)$ denotes the set of those nodes in $\mathbf{I}$ but not in $NP(P_n)$. To complete such a solution path, given we have visited nodes on path P_n, we must visit each node in $\mathbf{I}'(n)$ exactly once before returning to node γ. This observation enables us to define the minimum cost spanning tree heuristic estimate for the modified BA^* algorithm. Given a network G_0, a spanning tree in G_0 is a tree in G_0 that connects all of the nodes in G_0. Therefore, the cost of a minimum cost tree that connects the nodes in the set $\mathbf{I}'(n) \cup \{\gamma\}$ will give us an estimate of the cost-to-go from node n. Also, this estimate is optimistic since we ignore the precedence relations specified for the nodes in the set $\mathbf{I}'(n) \cup \{\gamma\}$. Hence, the heuristic estimate $h(n)$ for each node n in G' is now defined to be the estimated cost for traversing a minimum cost tree spanning the nodes in the set $\mathbf{I}'(n) \cup \{\gamma\}$. We now describe how to calculate the minimum cost spanning tree heuristic estimate.

Suppose node n' is the current node under consideration. Notice that we calculate $h(n')$, $n' \in SCS(n)$, when node n was selected from OPEN and expanded. Thus, $n' \in \mathbf{I}'(n)$. Let $N'' = \mathbf{I}'(n) \cup \{\gamma\}$ and let $G'' = (N'', A'')$ be the network that contains only nodes in N'' and arcs (k, k') such that $k \in N''$ and $k' \in N''$. The cost of an arc (k, k') in G'' is represented by $c(k, k')$. We want to determine the cost of a minimum cost spanning tree in the network G'' that is rooted at node n'. Many algorithms have been proposed for solving the minimum cost spanning tree problem. Here, we adopt the algorithm presented by Prim [10]; the overall computational complexity of which is $O(|N''|^2)$, where $|N''|$ is the number of nodes in the set N''. This is the most numerically desirable complexity bound we are aware of for problems on dense networks. Since the number of nodes in the network G'' for the application under consideration is usually not too large, we can expect this algorithm to work efficiently.

Prim's algorithm begins with the trivial spanning forest in G'' consisting of $|N''|$ isolated nodes. In each step, it selects one arc from A'' merging the forest components. Finally, after $(|N''| - 1)$ arcs are selected, all the forest components merge into a single spanning tree, which will be a minimum cost spanning tree. Note that in each step this algorithm grows only one component in the forest as a connected tree, leaving all the other components as isolated nodes. We will call the component being grown as the "tree." Nodes in the tree are labeled with predecessor indices, which are called permanent labels. Each out-of-tree node j carries a temporary label of the form (p_j, d_j), where:

$d_j = \infty$ and $p_j = 0$ if there is no arc connecting j with an in-tree node so far;

$d_j = \min\{c(i, j)$: over i an in-tree node such that $(i, j)A''\}$ and p_j is an i that attains this minimum.

This algorithm is outlined in Fig. 5.

V. Computational Experience

To gain an insight into the performance and computational efficiency of the DSS developed in this chapter, we have made a large number of tests using data associated with the fixed route system operated by AATA (Ann Arbor Transportation Authority) in Ann Arbor, Michigan. These tests can be divided into two categories:

1) tests using simulated request data;
2) tests using real request data.

All of these tests are performed on an IBM 80386 PC using TURBO PASCAL 6.0 compiler. In this section, we present some examples of these tests.

A. Tests Using Simulated Request Data

The basic information used in these tests are as follows:

Service area: Ann Arbor area (an area of 72.8 square miles);
Fixed route system: The fixed route system operated by AATA;

Initialization:	Let n'∈N" be the root node. Number the nodes in N" from 1 to |N"| with root node n' as node 1. Permanently label the root node 1 with label 0. For each node j such that (1,j)∈A" [(1,j)∉A"] temporarily label j with (1,c(1,j))[(0,∞)].
General step:	Among all out-of-tree nodes j at this stage, find one with the smallest d_j. Suppose it is r with temporary label (p_r,d_r), where d_j=Min{c(i,j): over i an in-tree node such that (i,j)∈A"}. Delete this temporary label on r and give it the permanent label p_r (its predecessor in the tree). If there are no out-of-tree nodes left, terminate. The resulting spanning tree defined by the permanent labels is a minimum cost spanning tree in G". If there are some out-of-tree nodes left, update their temporary labels as follows. For each out-of-tree node j with temporary label (p_j,d_j), change its label to (r,c(r,j)) only if (r,j)∈A" and c(r,j)<d_j. Leave the label unchanged otherwise. Then go to the next iteration.

Fig. 5. Prim's algorithm for the minimum cost spanning tree problem.

Distance metric: Euclidean (the direct travel distance between any two stops is the Euclidean distance between these two stops);
Location of request origins/destinations: Uniformly and independently distributed within the area;
Total number of available pieces of work: 20;
Depot location: AATA (all pieces of work have the same departure and arrival points);
Average paratransit vehicle speed: 15 MPH;
Time simulated: One day (from 6AM to 12PM);
Desired delivery time distribution: Uniform and independent;
Maximum deviation from the desired delivery time: 15 min;
Maximum excess riding time: 15 min + (0.5 $\times$ direct riding time);
Parameters: $MWT = 15$ min, $BS = 2.5$ min, $ATW = 1$ min, $ATP = 0.5$ min, and $\theta = 0.7$.

Table I displays results of these tests. In these tests, the computational time of the off-line DSS is about 10 min. The criteria of interest are the following: the number of requests serviced, the number of pieces of work used, and the total travel distance of paratransit vehicles (ft.).

As can be seen in Table I, the off-line DSS is quite effective in improving the initial solutions determined by the on-line DSS. For problems 1, 2, 4–7, and 9, the off-line DSS increases the number of requests serviced, while reduces both the number of pieces of work used and the total travel distance of paratransit vehicles. For problems 8, 10–14, the off-line DSS increases the number of requests serviced, while reduces the total travel distance of paratransit vehicles. Finally, for problem 3, the off-line DSS reduces both the number of pieces of work used and the total travel distance of paratransit vehicles.

B. Tests Using Real Request Data

The numerical results we now report involve testing the DSS with request data associated with the subscriber dial-a-

TABLE I
PERFORMANCE TESTS USING SIMULATED REQUEST DATA

		Solution of the on-line DSS			Solution of the off-line DSS		
Problem Number	# of requests generated	# of requests serviced	# of pieces of work used	total travel distance of para. vehicle	# of requests serviced	# of pieces of work used	total travel distance of para. vehicle
1	50	44	20	4210700	46	19	3821010
2	50	45	19	4627500	46	18	4166860
3	60	50	18	4315800	50	16	3824850
4	60	53	20	4918000	57	19	4351470
5	70	48	19	4402400	52	18	3880780
6	70	63	20	5119900	64	18	4911270
7	80	61	20	5121800	65	19	4855040
8	80	62	20	5144600	66	20	5040050
9	90	69	20	5540500	71	19	5284300
10	90	65	20	5282900	67	20	5034230
11	100	77	20	5739700	83	20	5685370
12	100	71	20	5607500	75	20	4501560
13	120	80	20	5595400	89	20	5526140
14	120	81	20	5864500	85	20	5753000

TABLE II
PERFORMANCE TESTS USING REAL REQUEST DATA

		Manual Solution			Solution of the on-line DSS			Solution of the off-line DSS		
Problem Number	total # of requests	# of requests serviced	# of pieces of work used	total travel distance of para.veh.	# of requests serviced	# of pieces of work used	total travel distance of para.veh.	# of requests serviced	# of pieces of work used	total travel distance of para.veh.
1	64	59	13	1657317	64	11	1803218	64	9	1622705
2	67	63	12	1714821	67	11	1712478	67	10	1628199
3	68	61	12	1694706	68	11	1737252	68	11	1641011
4	72	60	13	1721332	68	11	1835387	72	12	1735816
5	78	73	14	2032574	74	14	2122435	77	14	1893420
6	79	72	14	2153452	78	13	2325519	79	13	2110234
7	82	73	13	1898882	77	12	1913973	79	11	1800304
8	84	73	14	2020733	82	13	2398390	84	13	2108966
9	85	79	14	2136488	85	13	2253830	85	13	1959069
10	85	78	13	1960589	82	13	2099405	83	12	1832508

ride system operated by AATA. Each of the following ten test problems corresponds to a single day's data from this dial-a-ride system. The (reduced) average speed of paratransit vehicles is assumed to be (12) 16 MPH. The actual travel time from any point in the Ann Arbor area to any other point is calculated based on the Euclidean distance between these two points. There are 14 pieces of work available in this dial-a-ride system, each of which has a specified work period and a capacity of 4 wheelchairs and 10 passengers. Other information used in these tests are as follows:

Maximum deviation from the desired delivery time: 15 min;
Maximum excess riding time: 15 min + (0.5 $\times$ direct riding time);
Parameters: $MWT = 15$ min, $BS = 2.5$ min, $ATW = 1$ min, $ATP = 0.5$ min, and $\theta = 0.5$.

Table II presents manual solutions generated by the scheduling operators in AATA, as well as the solutions provided by our DSS for these ten test problems. We remark that the manual solutions are generated based on the assumption that no fixed route bus service is involved. Computation time of the off-line DSS is about 10 min for each test problem. The average percentage of the requests that can be serviced is 90.46% in the manual solutions, 97.6% in the solutions of the on-line DSS and 99.27% in the solutions of the off-line DSS. In relation to the manual solutions, the average reduction in the number of pieces of work used is as follows: 7.66% after the on-line DSS, and 10.8% after the off-line DSS. For 8 out of the 10 test problems (except problems 4 and 8), the solution of the off-line DSS not only has an increase in the number of requests serviced but also has reductions in both the number of pieces of work used and the total travel distance of paratransit vehicles, compared to the manual solution. That is, with the DSS described in this chapter, we are able to service more requests with less pieces of work and less travel distance of paratransit vehicles. Hence, using this DSS, we can both increase the service capacity and reduce the operation cost of a dial-a-ride system.

VI. CONCLUSIONS

We have presented a description and analysis of a DSS for the automatic construction of paratransit vehicle routes and schedules for a BDARP. This DSS consists of two subsystems: the on-line DSS and the off-line DSS. The on-line DSS, which is used whenever a new request appears adopts a greedy-type heuristic to identify an initial feasible solution with respect to the current input to the problem. The off-line DSS, which is used when no more requests will be considered, adopts an iterative improvement procedure based on simulated annealing and heuristic search techniques to improve the initial solution given by the on-line DSS. This DSS was tested on a set of data from the AATA, Ann Arbor, Michigan. Results of these tests have shown that with this DSS we are able to service more requests with less pieces of work and less travel distance of paratransit vehicles. Final implementation of this DSS is currently underway, and system evaluation is due to begin shortly.

In this paper, we have chosen simulated annealing as the technique for performing the iterative improvement procedure in the off-line DSS. The reason for this choice is due to the fact that simulated annealing is easy to implement in our problem setting. Basically, any other local search technique, e.g., genetic algorithm or tabu search, can be used to perform the iterative improvement procedure in the off-line DSS. Hence, one possible direction for future research is to determine the most efficient iterative improvement technique for the off-line DSS.

Another important direction for future research would be to develop a "dynamic" version of the off-line DSS, so that some requests could be added to the system on a real-time basis. In other words, while most customers would still be scheduled on an advance-request basis, the system would allow some customers to request immediate service, which would be provided through reconfiguration of existing paratransit vehicle routes and schedules with the assistance from such a dynamic version of the off-line DSS.

Finally, another direction for future research is to generalize the bimodal dial-a-ride problem to the multimodal dial-a-ride problem by including another transportation mode, e.g., cabs, into the dial-a-ride problem.

References

[1] E. Baker, "Time oriented vehicle routing and the traveling salesman problem," School of Business, Univ. of Miami, Aug. 1980.

[2] L. D. Bodin, B. L. Golden, A. Assad, and M. O. Ball, "Routing and scheduling of vehicles and crews: The state of the art," *Comp. and Ops. Res.*, vol. 10, pp. 63–211, 1983.

[3] L. D. Bodin and T. Sexton, "The multi-vehicle subscriber dial-a-ride problem," *The Delivery of Urban Services, TIMS Studies in The Management Sciences*, vol. 22, pp. 73–86, 1986.

[4] J. Desrosiers, Y. Dumas, and F. Soumis, "A dynamin programming solution of the large-scale single-vehicle dial-a-ride problem with time windows," *Amer. J. Math. and Mgmt. Sci.*, vol. 6, pp. 301–325, 1986.

[5] ——, "The multiple vehicle dial-a-ride problem," in *Lecture Notes on Economics and Mathematical System 308: Computer-Aided Transit Scheduling*, J. R. Daduna and A. Wren, Eds. Berlin: Springer-Verlag, 1988, pp. 15–27.

[6] J. Jaw, A. Odoni, H. Psaraftis, and N. Wilson, "A heuristic algorithm for the multi-vehicle many-to-many advance-request dial-a-ride problem with time windows," *Transportation Res.*, vol. 20B, pp. 243–257, 1986.

[7] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[8] P. J. M. V. Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications.* Dordrecht: Kluwer, 1987.

[9] C. Liaw and C. C. White, III, "A heuristic search approach for solving a minimum path problem requiring arc cost determination," *IEEE Trans. Syst., Man, Cybern.*, this issue, pp. 545-551.

[10] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Sys. Tech. J.*, vol. 36, pp. 1389–1401, 1957.

[11] H. Psaraftis, "An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows," *Trans. Sci.*, vol. 17, pp. 351–357, 1983.

[12] S. Roy, J. M. Rousseau, L. Chapleau, G. Lapalme, and J. A. Ferland, "Routing and scheduling for the transportation of disabled person—The algorithm," Transport Canada, Transport Development Centre, Montreal, TP 5596E, 1984.

[13] T. Sexton and B. L. Bodin, "Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling," *Trans. Sci.*, vol. 19, pp. 378–410, 1985.

[14] ——, "Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing," *Trans. Sci.*, vol. 19, pp. 411–435, 1985.

Ching-Fang Liaw, photograph and biography not available at the time of publication.

Chelsea C. White, III (S'73–M'74–SM'87–F'88) for a photograph and biography, see p. 551, this issue.

James Bander (M'91) received the B.A. degree in mathematics, and the M.S. degree in systems engineering from the University of Virginia, Charlottesville. He is now a Ph.D. student in the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor. His research interests include knowledge representation, routing and scheduling, and transportation information systems.