

# **An Insertion Heuristic for Scheduling Mobility Allowance Shuttle Transit (MAST) Services**

**Luca Quadrifoglio, Maged M. Dessouky\* and Kurt Palmer**

**Daniel J. Epstein Department of Industrial and Systems Engineering  
University of Southern California  
Los Angeles, California 90089-0193**

**\*Corresponding Author  
maged@usc.edu  
phone: 213-740-4891; fax: 213-740-1120**

## **Abstract**

In this paper we develop an insertion heuristic for scheduling Mobility Allowance Shuttle Transit (MAST) services, an innovative concept that merges the flexibility of Demand Responsive Transit (DRT) systems with the low cost operability of fixed-route systems. A MAST system allows vehicles to deviate from the fixed path so that customers within a service area may be picked up or dropped off at their desired locations. Such a service already exists in Los Angeles County, where MTA Line 646 is a MAST nighttime service, transporting passengers between a business area and a nearby bus terminal. Since the current demand is very low, the service is entirely manageable by the bus operator, but a higher demand would certainly require the development of a scheduling algorithm. The proposed insertion heuristic makes use of control parameters, which properly regulate the consumption of the slack time. A set of simulations performed in the service area covered by the existing MTA Line 646 at different demand levels attests the effectiveness of the algorithm by comparing its performance vs. a First-Come/First-Serve policy and optimal solutions generated by a commercial integer program solver. The results show that our approach can be used as an effective method to automate scheduling of this line and other services similar to it.

## **1 Introduction**

Today's urban transit systems are at a crossroads. On the one hand, demands on transit agencies for improved and extended services are increasing. Yet on the other, there is little public support for increases in fares or subsidies. Therefore, transit agencies are currently seeking ways to improve service flexibility in a cost efficient manner.

Most transit systems fall into two broad categories: fixed-route and Demand Responsive Transit (DRT) systems. Fixed-route systems are typically more cost efficient because of the predetermined schedule, the large loading capacity of the vehicles and the consolidation of many passenger trips onto a single vehicle (ridesharing). However, the general public considers them to be inconvenient because of their lack of flexibility, since either the locations of pick-up and/or drop-off points or the service's schedule do not match the individual rider's desires. Moreover, the total trip time is perceived as being too long and for longer trips there is often a need for transfers between vehicles. DRT systems instead provide much of the desired flexibility with a door to door type of service but they are much more costly to deploy and therefore largely limited to specialized operations such as taxicab, shuttle vans or Dial-a-Ride services mandated under the Americans with Disabilities Act (paratransit DRT). The National Transit Summaries and Trends (NTST) report for 2002 indicates that the average cost per passenger trip for DRT systems is \$20.8 with fares ranging from \$2-3. By way of contrast, the average cost per trip for fixed-route lines is \$2.4 with fares being roughly 25% of the cost (Palmer et al. 2004).

Thus, there is a need for a transit system that provides flexible service at a cost efficient price. The Mobility Allowance Shuttle Transit (MAST) system is an innovative concept that merges the flexibility of DRT systems with the low cost operability of fixed-route bus systems. A MAST service has a fixed base route that covers a specific geographic zone, with one or more mandatory checkpoints conveniently located at major connection points or high density demand zones; the innovative twist is that, given an appropriate slack time, vehicles are allowed to deviate from the fixed path to pick up and drop off passengers at their desired locations. The only restriction on flexibility is that the deviations must lie within a service area designed around the base fixed-route. Customers make a reservation in order to add their desired pick-up and/or drop-off stops in the

schedule of the service. The MAST system works under a dynamic environment since the majority of the requests occur while the vehicle is on duty.

Such a system already exists in a reduced and simplified scale. The Metropolitan Transit Authority (MTA) of Los Angeles County has recently introduced MAST as part of its feeder-line 646. Line 646 transports passengers between a large business hub in the San Pedro area of Los Angeles County and a nearby bus terminal. The area that Line 646 serves is located close to the Los Angeles harbor and is one of the county's busiest commercial hubs, consisting of several warehouses, factories and offices. However, for safety reasons, employees of local firms working on night shifts have been finding it extremely inconvenient to walk to and wait at a bus stop. Therefore, Line 646 offers a MAST *nightline* service. During daytime, this line serves as a fixed-route bus system. During nighttime, the line changes to a MAST service and allows specific deviations of half a mile from either side of the fixed-route. Customers may call in to be picked up or may ask the operator to be dropped off at their desired locations if within the service area.

The demand of line 646 is currently low enough to allow the bus operator to make all the decisions concerning accepting/rejecting requests and routing the vehicle. Clearly, in case of heavier demand in a potential daytime MAST operation and several requests for deviations, the operator would not be able to handle this task efficiently by himself/herself and would need help from the recent developments in communication and computation technologies that allow real-time information about pick-up/drop-off requests and vehicles status to be used to re-route the vehicles dynamically by means of a scheduling algorithm.

While DRT systems focus strictly on point to point transport services, the hybrid characteristic of the MAST service adds additional and significant time and space constraints to the problem mainly due to the need of having the vehicles arrive at the checkpoints on or before their scheduled departure time. This is because the checkpoints typically represent major transfer centers and serve simultaneously as pick-up and drop-off points, like regular fixed-route stops. Delays at the checkpoints would result in undesirable deviations from a predetermined fixed schedule and passengers missing their connections in case of late arrivals.

Although MAST systems can be considered as a special case of the Pickup and Delivery Problem (PDP) with time windows and there has been a significant amount of

research on DRT systems like the PDP, systems such as the MAST service have not yet been extensively studied by researchers. The purpose of this paper is to describe these types of services and develop an insertion heuristic algorithm suitable for a MAST system. An insertion heuristic approach is used because they are computationally fast and they can easily handle complicating constraints in a dynamic environment (Campbell and Savelsbergh, 2004). The vehicle route and schedule are updated in real time after each request and customers are immediately notified whether their request has been accepted and are provided with time windows for their pick-up and/or drop-off stops.

The remainder of this paper is divided into the following sections. After reviewing the literature in Section 2, we define the MAST system in Section 3 and the control parameters needed to enhance the algorithm performance in Section 4. Section 5 illustrates the algorithm. In Section 6 we describe the experimental results and the performance evaluation. Section 7 provides conclusions.

## **2 Literature review**

Hybrid types of transportation systems have been only recently approached by researchers. Zhao and Dessouky (2005) studied the optimal service capacity through a stochastic approach. Malucelli et al. (1999) also approached the problem including it in a general overview of flexible transportation systems. Crainic et al. (2001) described the MAST concept and incorporated it in a more general network setting providing also a mathematical formulation.

The hybrid type of service that we are studying consists of the same vehicle performing the fixed and variable portions of the trip. There has been some work in studying hybrid systems in which different vehicles perform the fixed and variable portions. In the latter case, local service is provided by on-demand vehicles and line-haul service is provided by a fixed-route line. Passengers switch vehicles at a transfer station. Aldaihani et al. (2004) develop a continuous approximation model for designing such a service. There has been some work in developing operational scheduling and routing policies for this latter type of hybrid system. Liaw et al. (1996) develop a scheduling heuristic based on a system in Ann Arbor, Michigan. Hickman and Blume (2000) develop

an insertion heuristic and test it on a data set from Houston, Texas. Aldaihani and Dessouky (2003) develop a tabu search heuristic and test it on a data set from Antelope Valley in California. They show that shifting some of the demand to a hybrid service route (18.6% of the requests) reduces the on-demand vehicle distance by 16.6% without significantly increasing the trip times.

As previously mentioned, there is a significant body of work in the literature on scheduling and routing DRT systems. Desaulniers et al. (2000) and Savelsbergh and Sol (1995) provide a detailed review of the Pickup and Delivery problem and its related problems. Most of this work is intended for Dial-a-Ride systems for the delivery of the elderly and handicapped. Pioneering research on the Dial-a-Ride problem dates back to the seventies. Wilson et al. (1971) formulate the problem as a dynamic search procedure, inserting newly arriving passenger's origin and destination into the prospective route of one of the buses. Continuing work is presented by Wilson and Hendrickson (1980). Daganzo (1978) presents a model to evaluate the performance of a Dial-a-Ride system. Theoretical studies and exact algorithms for the scheduling problem include the work by Psaraftis (1980, 1983a), Sexton and Bodin (1985a, 1985b), Sexton and Choi (1986), Desrosiers et al. (1986) and Lu and Dessouky (2004). Heuristic approaches include the work by Psaraftis (1986), Jaw et al. (1986), Bodin and Sexton (1986), Desrosiers et al. (1988) and Madsen et al. (1995). Parallel insertion heuristics are proposed by Toth and Vigo (1997), Dessouky et al. (2003), Diana and Dessouky (2004) and Lu and Dessouky (2006).

### **3 Description of MAST services and systems definition**

A Mobility Allowance Shuttle Transit (MAST) system is represented by a fleet of vehicles serving a set of customers' requests. Vehicles follow a fixed-route line (back and forth between two terminal checkpoints or around a loop, see Figure 1) composed by an ordered set of stops (*checkpoints*) associated with prescheduled departure times.

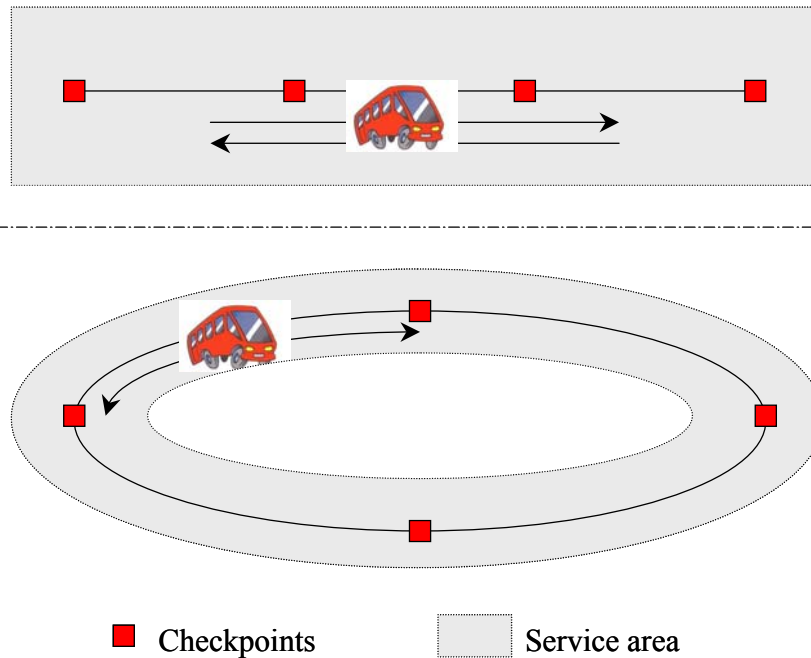


Figure 1 – Possible configurations of MAST systems

The demand is defined by a set of requests. Each request is defined by pick-up/drop-off service stops and a ready time for pick-up. The MAST service can respond to four different types of requests: pick-up (P) and drop-off (D) at the checkpoints; non-checkpoint pick-up (NP) and drop-off (ND), representing customers picked up/dropped off at any location within a *service area* designed around the base fixed-route. A certain amount of *slack time* between any consecutive pair of checkpoints is needed in order to allow deviations to serve NP or ND requests.

There are consequently four different possible types of customers' requests:

- PD (“regular”): pick-up and drop-off at the checkpoints
- PND (“hybrid”): pick-up at the checkpoint, drop-off not at the checkpoint
- NPD (“hybrid”): pick-up not at the checkpoint, drop-off at the checkpoint
- NPND (“random”): pick-up and drop-off not at the checkpoints

PD requests rely only on already scheduled checkpoints and they use the service like a regular fixed-route line; therefore, they just show up at their pick-up checkpoint,

without any need of a booking or scheduling procedure. The other types of requests need to make reservations instead (by phone, internet or at terminals located at the checkpoints) to schedule one or both their non-checkpoint service stops. The service can work dynamically, so that customers may book their rides (or show up at the checkpoints) at any moment before or during the service.

### ***MTA Line 646***

The MTA Line 646 in San Pedro, Los Angeles County, offers a MAST *nightline* service. Line 646 consists of a single vehicle covering a quasi-rectangular service area (approximately 10×1 miles), with two terminal checkpoints and one intermediate checkpoint located in the middle. The duration of each trip between terminal checkpoints is 30 minutes and the headway is 1 hour. The service operates for 4.5 hours (9 trips) each night. The system has very little slack time, but this is justified by the very low actual demand (4-5 customers/hour, most of them being of type PND and NPD). These “light” conditions allow the bus operator to easily make in real time all the decisions concerning accepting/rejecting customer requests and routing the vehicle since the system needs to deal with only 2-3 non-checkpoint requests per trip.

## **3.1 Model**

The MAST system model analyzed in this paper is essentially based on the existing MTA Line 646 in San Pedro (Los Angeles County) and consists of a single vehicle, initially associated with a predefined schedule along a fixed-route, consisting of  $C$  checkpoints, identified by  $c = 1, 2, \dots, C$ ; two of them are terminals located at the extremities of the route ( $c = 1$  and  $c = C$ ) and the remaining  $C-2$  intermediate checkpoints are distributed along the route. The vehicle is moving back and forth between 1 and  $C$ . A trip  $r$  is defined as a portion of the schedule beginning at one of the terminals and ending at the other one after visiting all the intermediate checkpoints; the vehicle’s schedule consists of  $R$  trips. Since the end-terminal of a trip  $r$  corresponds to the start-terminal of the following trip  $r+1$ , the total number of stops at the checkpoints is  $TC = (C-1)R+1$ . Hence, the initial schedule’s array is represented by an ordered sequence of stops  $s = 1, \dots, TC$ .

The service area is represented by a rectangular region defined by  $L \times W$ , where  $L$  (on the  $x$  axis) is the distance between terminals 1 and  $C$  and  $W/2$  (on the  $y$  axis) is the maximum allowable deviation from the main route in either side (see Figure 2).

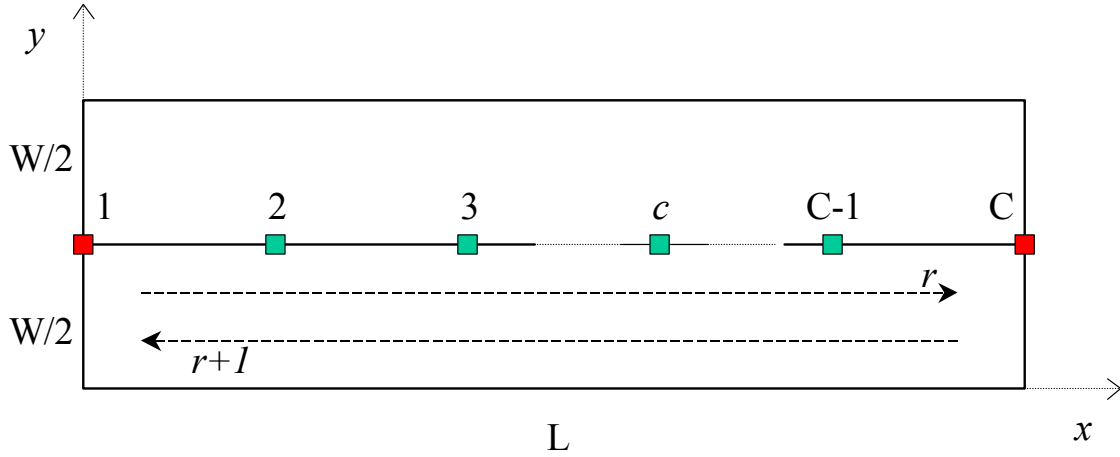


Figure 2 – MAST system model

### ***Demand***

We assume that the total demand rate  $\theta$  (including any type of request) is constant over time and that the non-checkpoint's stops (NP and ND) are uniformly distributed in the service area. At any moment a customer may call in (or show up at the checkpoints), specifying the locations of pick-up and/or drop-off points. We assume that customers are immediately ready to be picked up at the moment of their request, even though the system could easily handle reservations for future pick-ups.

We identify the checkpoints with  $s = 1, \dots, TC$ , and the non-checkpoint customers' requests (NP or ND) with  $s = TC+1, \dots, TS$ , where  $TS$  represents the current total number of stops. Each stop  $s$  has an associated departure and arrival times respectively identified by  $t_s$  and  $t'_s$ . As mentioned, the scheduled departure times  $t_s$  of the checkpoints ( $\forall s \leq TC$ ) are fixed and assumed to be constraints of the system, which can not be violated, while the  $t_s$  of the non-checkpoint stops ( $\forall s > TC$ ) and all the  $t'_s$  are variables of the system.

### ***Slack time***

In order to allow deviations from the main route to serve NP and ND requests between two consecutive checkpoints, identified by  $s$  and  $s+1$ , there needs to be a certain amount of initial slack time  $st_{s,s+1}^{(0)}$  available in the schedule, given by

$$st_{s,s+1}^{(0)} = t_{s+1} - t_s - d_{s,s+1}/v - h_{s+1} \quad \forall s = 1, \dots, TC-1 \quad (1)$$

where  $v$  is the vehicle speed,  $h_{s+1}$  is the time allowed at stop  $s+1$  for passengers' boardings and disembarkments and  $d_{s,s+1}$  is the distance between  $s$  and  $s+1$ . As more pick-ups and drop-offs occur off the base route, the current slack time  $st_{s,s+1}$  available is progressively reduced. Initially,

$$st_{s,s+1} = st_{s,s+1}^{(0)} \quad \forall s = 1, \dots, TC-1 \quad (2)$$

### ***Vehicle assumptions***

We assume that the vehicle follows a rectilinear metric, allowing the vehicle to move only along the horizontal or vertical directions only, since vehicles ride along roads which often form a grid especially in urban areas. In addition, we assume the vehicle capacity to be infinite, since most DRT systems are more constrained by time rather than capacity. These are both reasonable assumptions as outlined by an analysis of a large DRT system in Los Angeles County performed by Dessouky et al. (2005).

## **3.2 Problem definition**

Given the number of customers NC served by the MAST systems, the overall objective function  $Z$  (in time units) of the problem is defined as

$$Z = w_1 \times M/v + w_2 \times RT \times NC + w_3 \times WT \times NC \quad (5)$$

where  $w_1$ ,  $w_2$  and  $w_3$  are weights and

- M represents the total miles driven by the vehicle
- RT is the average ride time per customer
- WT is the average waiting time per customer from the ready time to the pick-up time

This definition of the  $Z$  allows optimizing in terms of both the vehicle variable cost (first term) and the service level (the last two terms); modifying the weights accordingly we can emphasize one factor over the others as needed.

Let  $\alpha(s)$  represents the current position of stop  $s$  in the schedule,  $\forall s$ . The problem defined by a MAST system is to determine the indices  $\alpha(s)$  and the departure/arrival times  $t_s, \forall s > TC$ , and  $t'_s, \forall s$ , in order to minimize  $Z$ , while not violating the checkpoints' fixed departure times  $t_s, \forall s \leq TC$ , and the customers' precedence constraints. The problem is solved dynamically by means of an insertion heuristic algorithm described in the following sections.

## 4 Control parameters

The challenge of operating a MAST system mainly resides in defining the logic to best operate the vehicle under a dynamic multi-criteria environment. In particular we need to set the insertion feasibility rules for any given customer at any point in time because inserting a new request in the vehicle's schedule even if feasible at that time, might not be best overall. For this purpose we make use of parameters that are a function of the slack time (*usable slack time*) and the relative position of the new request with respect to the already scheduled stops (*backtracking distance*).

### 4.1 Usable slack time

The slack time is a crucial resource needed to serve customers requesting deviations of the vehicle from its current route. When this resource is scarce, the system is not able anymore to satisfy new incoming requests. Therefore, a MAST service needs to be

particularly careful about accepting customer requests that can consume a lot of the slack time whereby preventing future requests from being satisfied. Thus, we define a parameter that properly controls the consumption of slack time.

We define the *usable slack time*  $st''_{s,s+1}$  as the maximum amount of slack time that any request is allowed to consume for its placement between checkpoints  $s$  and  $s+1$ . It represents an upper bound on the amount of slack time utilizable by a single request, preventing a single request from consuming too much of it.  $st''_{s,s+1}$  is defined as a function of the future expected demand between  $s$  and  $s+1$  and is not directly related to the actual unused slack time  $st_{s,s+1}$ ; in fact,  $st''_{s,s+1}$  can be greater or lower than  $st_{s,s+1}$  depending on the circumstances. As we will see in Section 5.1, a request will be allowed to consume the minimum value among  $st''_{s,s+1}$  and  $st_{s,s+1}$  in order to be feasible and be scheduled between  $s$  and  $s+1$ . Otherwise, it will be considered for placement in succeeding portions of the schedule; thus, they are not rejected, but postponed.

Let  $\lambda$  be the demand rate of non-checkpoint's stops (NP and ND), uniformly distributed in the service area and constant over time (note that  $\lambda$  is closely related but not equal to  $\theta$  defined earlier). The time interval between two checkpoints  $s$  and  $s+1$  is defined by  $t_{s+1}-t_s$ , while the ratio between the area covered by the segment of the route from  $s$  and  $s+1$  and the total service area is given by  $\frac{|x_s - x_{s+1}|}{L}$  (where  $x_s$  and  $x_{s+1}$  are the longitudinal coordinates of  $s$  and  $s+1$ ). Consequently, the expected total number  $\Lambda_{s,s+1}$  of requests between  $s$  and  $s+1$  during the time interval from  $t_s$  to  $t_{s+1}$  is estimated as (see Figure 3):

$$\Lambda_{s,s+1} = \lambda \frac{|x_s - x_{s+1}|}{L} (t_{s+1} - t_s) \quad (6)$$

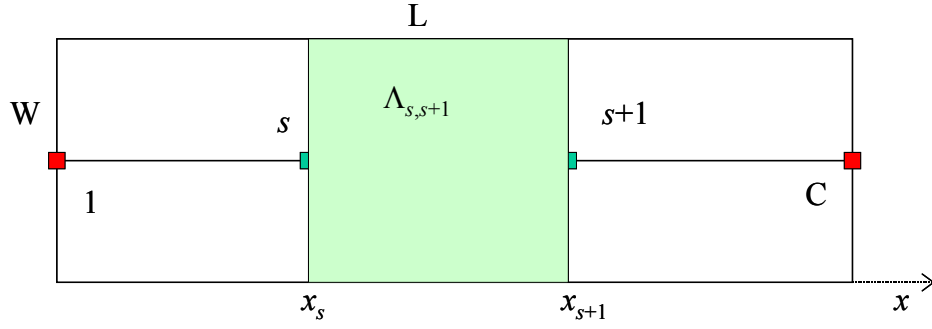


Figure 3 – Portion of service area covered by the segment between  $s$  and  $s+1$

As soon as the vehicle departs from  $s$  at  $t_s$ , the expected residual demand drops linearly from  $\Lambda_{s,s+1}$  until reaching the zero value at  $t_{s+1}$ . Hence, the expected residual demand as a function of the current clock time  $t_{now}$ ,  $\Lambda_{s,s+1}^{(t_{now})}$ , may be expressed as (see Figure 4):

$$\Lambda_{s,s+1}^{(t_{now})} = \begin{cases} \Lambda_{s,s+1} & t_{now} < t_s \\ \Lambda_{s,s+1} \left( 1 - \frac{t_{now} - t_s}{t_{s+1} - t_s} \right) & t_s \leq t_{now} \leq t_{s+1} \\ 0 & t_{now} > t_{s+1} \end{cases} \quad (7)$$

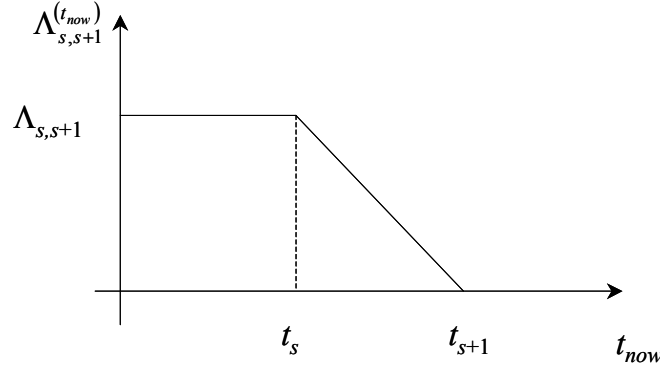


Figure 4 – Expected residual demand between  $s$  and  $s+1$  as a function of  $t_{now}$

We define the parameter  $\pi_{s,s+1}$  as a function of the expected demand as follows:

$$\pi_{s,s+1} = 1 + \left( \frac{\pi_{s,s+1}^{(0)} - 1}{\Lambda_{s,s+1}} \right) \Lambda_{s,s+1}^{(t_{now})} \quad \text{with } 0 \leq \pi_{s,s+1}^{(0)} \leq 1 \quad (8)$$

Since  $0 \leq \Lambda_{s,s+1}^{(t_{now})} \leq \Lambda_{s,s+1}$ , we have that  $\pi_{s,s+1}^{(0)} \leq \pi_{s,s+1} \leq 1$  and  $\pi_{s,s+1}^{(0)}$  can be set accordingly.

We finally define the usable slack time  $st_{s,s+1}^u$  as

$$st_{s,s+1}^u = \pi_{s,s+1} st_{s,s+1}^{(0)} \quad (9)$$

If the residual expected demand  $\Lambda_{s,s+1}^{(t_{now})} \rightarrow 0$ , then  $\pi_{s,s+1} \rightarrow 1$  and  $st_{s,s+1}^u \rightarrow st_{s,s+1}^{(0)}$ . Whereas, when  $\Lambda_{s,s+1}^{(t_{now})}$  attains its maximum ( $\Lambda_{s,s+1}$ ),  $\pi_{s,s+1}$  reaches its minimum value,  $\pi_{s,s+1}^{(0)}$ , and so does  $st_{s,s+1}^u = \pi_{s,s+1}^{(0)} st_{s,s+1}^{(0)}$ .

Combining Equations (7), (8) and (9) we finally derive the expression for the *usable slack time*  $st_{s,s+1}^u$  as a function of  $t_{now}$  (see Figure 5):

$$st_{s,s+1}^u = \begin{cases} \pi_{s,s+1}^{(0)} st_{s,s+1}^{(0)} & t_{now} < t_s \\ \left[ 1 + \left( \pi_{s,s+1}^{(0)} - 1 \right) \left( 1 - \frac{t_{now} - t_s}{t_{s+1} - t_s} \right) \right] st_{s,s+1}^{(0)} & t_s \leq t_{now} \leq t_{s+1} \\ st_{s,s+1}^{(0)} & t_{now} > t_{s+1} \end{cases} \quad (10)$$

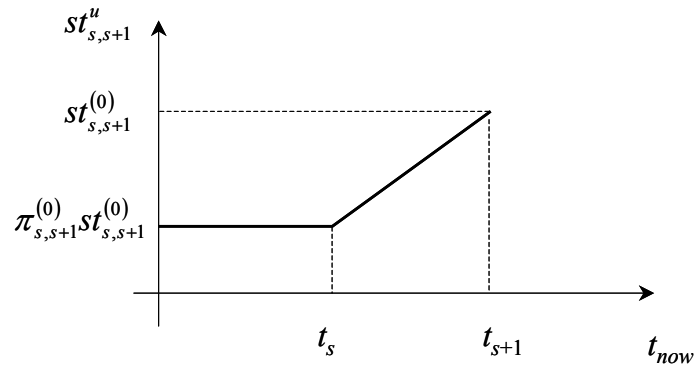


Figure 5 – Usable slack time

Let's now consider a non-checkpoint request  $q$  located at the edge of the service area, such that  $y_q = 0$  or  $y_q = W$  and  $x_s \leq x_q \leq x_{s+1}$  and let's assume that the schedule between  $s$  and  $s+1$  is empty (no previously placed stops). In order to be placed here, the  $q$  request would require an amount of slack time  $st_q$  given by the time needed by the vehicle to deviate from the  $x$  axis, serve the  $q$  request and come back to the  $x$  axis ( $st_q = W/v+h_q$ ). Since the minimum amount of usable slack time from Equation (10) is given by  $st_{s,s+1}^u = \pi_{s,s+1}^{(0)} st_{s,s+1}^{(0)}$ , we need to have  $st_{s,s+1}^u \geq st_q$  to prevent the  $q$  request from being rejected in this situation. Hence we define:

$$\pi_{s,s+1}^{(0)\min} = (W/v+h_q)/st_{s,s+1}^{(0)} \quad (11)$$

as the minimum value of  $\pi_{s,s+1}^{(0)}$  that guarantees every non-checkpoint request  $q$  to be feasibly placed between  $s$  and  $s+1$  with the schedule empty, regardless of the location of  $q$  as long as  $x_s \leq x_q \leq x_{s+1}$ .

Setting  $\pi_{s,s+1}^{(0)} < \pi_{s,s+1}^{(0)\min}$  would prevent the algorithm from working properly, because some customers would be rejected because of an improper parameter setting. Clearly, setting  $\pi_{s,s+1}^{(0)} = 0$  would result in having  $st_{s,s+1}^u = 0$  for  $t_{now} < t_s$ , preventing any requests before  $t_s$  from being satisfied. On the contrary,  $\pi_{s,s+1}^{(0)} = 1$  causes  $st_{s,s+1}^u = st_{s,s+1}^{(0)}$  at any time and customers requests would have no limit on the amount of slack time they could consume.

A proper value of  $\pi_{s,s+1}^{(0)}$  in between  $\pi_{s,s+1}^{(0)\min}$  and 1 allows the system to control the consumption of slack time. Looking at Figure 5 any request occurring before  $t_s$  can use at most the minimum value of  $st_{s,s+1}^u = \pi_{s,s+1}^{(0)} st_{s,s+1}^{(0)}$  allowing the future expected customers to be properly served with the remaining slack time. Whereas, if a customer request occurs towards the end of the trip from  $s$  to  $s+1$ , it is allowed to consume a bigger portion of the slack time until a maximum of  $st_{s,s+1}^{(0)}$  because the chance of having additional requests before the vehicle reaches the next checkpoint  $s+1$  is very low.

We want to point out that the choice of shaping the usable slack time function as shown in Figure 5 is arbitrarily derived from an empirical approach and could have been developed in different ways, for example by replacing  $\Lambda_{s,s+1}$  with the total expected demand occurring in a whole cycle (the duration of two trips and not only the time interval between  $t_s$  and  $t_{s+1}$ ), with the result of having the location of the corner of the function (currently at  $t_s$ ) moved earlier. However, the concept of limiting the consumption of the slack time by setting a parameter such as  $\pi_{s,s+1}^{(0)}$  would have been analogous.

## 4.2 Backtracking distance

The vehicle could drive back and forth with respect to the direction of a trip  $r$  while serving customers in the service area, not only consuming the extra slack time, but also having a negative impact on the customers already onboard, which may perceive this behavior as an additional delay. Therefore, we limit the amount of backtracking in the schedule. The backtracking distance indicates how much the vehicle drives backwards on the  $x$  axis while moving between two consecutive stops to either pick up or drop off a customer at a non-checkpoint stop with respect to the direction of the current trip. More formally, as shown in Figure 6, given any two consecutive stops identified by  $a$  and  $b$ , such that  $\alpha(a)+1 = \alpha(b)$ , and the vector  $\hat{d}_{a,b}$  representing the distance from  $a$  to  $b$ , the backtracking distance  $bd_{a,b}$  is defined as the negative component of the projection of  $\hat{d}_{a,b}$  along the unit vector  $\hat{d}_r$ , representing the direction of the current trip  $r$  ( $1 \rightarrow C$  or vice versa, parallel to the  $x$  axis) as follows:

$$bd_{a,b} = -\min(0, \hat{d}_r \circ \hat{d}_{a,b}) \quad (12)$$

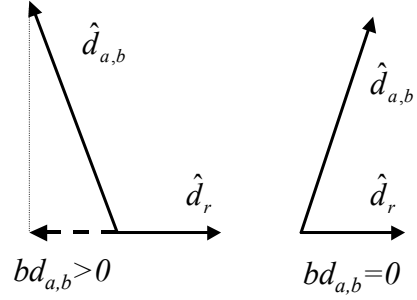


Figure 6 – Backtracking distance

We define the control parameter  $BACK > 0$  that is the maximum allowable backtracking distance that the vehicle can ride between any two consecutive stops.  $BACK$  can be set accordingly; clearly with  $BACK \geq L$  any backtracking is allowed.

## 5 Algorithm description

The proposed dynamic insertion heuristic is described in this section. We define the feasibility of an insertion in Section 5.1. We describe how to evaluate the cost of each feasible insertion in Section 5.2. We introduce the definition of “bucket” in Section 5.3, needed to describe the insertion procedure outlined in Section 5.4. We finally illustrate in Section 5.5 the update procedure after a feasible insertion candidate has been chosen by the algorithm to be placed in the schedule.

### 5.1 Feasibility

While evaluating a customer request, the algorithm needs to determine the feasibility of the insertion of a new stop  $s = q$  between any two consecutive stops  $a$  and  $b$  already scheduled. The extra time needed for the insertion is computed as follows:

$$\Delta t_{a,q,b} = (d_{a,q} + d_{q,b} - d_{a,b}) / v - h_q \quad (13)$$

Let  $m$  and  $m+1$  be the checkpoints before and after stops  $a$  and  $b$  in the schedule. The algorithm computes  $st_{m,m+1}^u$  by Equation (10) and the backtracking distances  $bd_{a,q}$  and  $bd_{q,b}$  by Equation (12). Finally, it is feasible to insert  $q$  between  $a$  and  $b$  if (see Figure 7):

$$\begin{cases} \Delta t_{a,q,b} \leq \min(st_{m,m+1}, st_{m,m+1}^u) & (14) \\ bd_{a,q} \leq \text{BACK} & (15) \\ bd_{q,b} \leq \text{BACK} & (16) \end{cases}$$

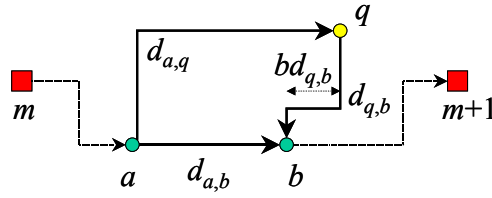


Figure 7 – Insertion feasibility of  $q$

## 5.2 Cost function

The cost function considered by the dynamic algorithm is similar to Equation (5), which represents the overall objective function of a MAST system, suitable for a static environment. The system's entities affected by an insertion in a dynamic environment are:

- i. The vehicle, in terms of how many extra miles it has to drive.
- ii. The customer requesting the insertion, in terms of how long the waiting time until the pick-up is and how long the ride time is.
- iii. The passengers already onboard and waiting to be dropped off, in terms of how much longer they have to stay onboard.
- iv. The previously inserted customers in the schedule waiting to be picked up at the NP stops, in terms of how much longer their pick-up time is delayed and also in terms of how much their ride time changes.

For each feasible insertion of a stop  $q$  the algorithm computes the following quantities:

- $\Delta RT$ : the sum over all passengers of the extra ride time, including the ride time of the customer requesting the insertion.
- $\Delta WT_E$ : the sum over all passengers of the extra waiting time at the already inserted NP stops.

Finally, the cost function is defined as:

$$\text{COST} = w_1 \times \Delta t_{a,q,b} + w_2 \times \Delta RT + w_3 \times \Delta WT_E \quad (17)$$

where  $\Delta t_{a,q,b}$ , from Equation (13), represents the consumption of the slack time. During heavy demand periods, it should be assigned a higher value to this scarce resource by increasing  $w_1$  with respect to  $w_2$  and  $w_3$ . In contrast, during low demand periods, the opposite is true and the COST function should emphasize more the service quality for the customers rising  $w_2$  and  $w_3$  over  $w_1$ .

In a dynamic insertion procedure we can distinguish among two different types of waiting time. The first type is the waiting time spent by the customer initially from the ready time to the pick-up time at the moment of their request (mentioned in point ii above), that customers may spend at their home, office or a comfortable location of their NP pick-up location (or at the checkpoints in case of P pick-up) and that we define as  $WT_1$  (as average per customer). The second type is the extra waiting time defined as  $WT_E$ , as average per customer ( $\Delta WT_E$  as sum, defined above) that customers have to spend at their NP stops, due to other insertions in the schedule (mentioned in point iv above).

The objective function  $Z$  defined in Equation (5) measures the total  $WT = WT_1 + WT_E$ , because it is defined for a static environment. The COST function instead measures the incremental cost and is directly related to the objective function  $Z$ , but it does not account for  $WT_1$  that is indirectly minimized by working with “buckets” that we define in the following section and describe their use in Section 5.4.

### 5.3 Buckets

Let's consider the schedule's array as shown in Table 1, illustrating the checkpoints only with their corresponding stop index  $s$ . Each checkpoint  $c$  is scheduled to be visited by the vehicle a number of times, with different stop indices  $s_k(c)$  (stop index of the  $k^{\text{th}}$  occurrence of checkpoint  $c$  in the schedule), depending on how many trips  $R$  are planned.

Table 1 – Schedule's array and buckets

trip	$s$	Checkpoints $c$
1	1	1
	2	2
	3	3
	...	...
	$c$	$c$
	...	...
	$C-1$	$C-1$
	$C$	$C$
2	$C+1$	$C-1$
	...	...
	$2(C-1)+1-(c-1)$	$c$
	...	...
	$2C-2$	2
3	$2C-1$	1
	$2C$	2
	...	...
	$2(C-1)+1+(c-1)$	$c$
$\dots$	...	...
	...	...
	...	...
$r$	$r(C-1)+1-(c-1)$	$c$
	...	...
$r+1$	$r(C-1)+1$	1
	...	...
	$r(C-1)+1+(c-1)$	$c$
$\dots$	...	...
	...	...
$R$	$TC=R(C-1)+1$	1 or $C$

1<sup>st</sup> bucket of  $c=1$

bucket of  $c=2$

another bucket of  $c=2$

For each intermediate checkpoint  $c = 2, \dots, C-1$  the indices  $s_k(c)$ , which identify them in the schedule, are computed by the following sequence:

$$s_r(c) = 1 + (C-1)(r-1) + \frac{(C-1) + (-1)^r [(C-1) - 2(c-1)]}{2} \quad \forall k = 1, \dots, R \quad (18)$$

For the terminal checkpoints 1 and C, since their frequency of occurrence is halved, the sequences are the following:

$$s_r(1) = 1 + 2(C-1)(k-1) \quad \forall k = 1, \dots, 1 + \lfloor R/2 \rfloor \quad (19)$$

$$s_r(C) = C + 2(C-1)(k-1) \quad \forall k = 1, \dots, \lceil R/2 \rceil \quad (20)$$

Definition: For every checkpoint  $c$ , we define a **bucket of  $c$** , in general, as a portion of the schedule delimited by two successive occurrences of  $c$ , namely all the stops  $s$  in the current schedule's array such that  $\alpha[s_k(c)] \leq \alpha(s) < \alpha[s_{k+1}(c)]$  for any allowable  $k$ , as described in Equations (18), (19) and (20).

The buckets' definition for NPND type customers needs to be revised since they do not rely on the checkpoints for pick-ups and drop-offs; so we identify the buckets with the trips. More formally, let's characterize the sequence representing the occurrences of any terminal checkpoint ( $c = 1$  or  $C$ ):

$$s_k(1 \text{ or } C) = 1 + (C-1)(k-1) \quad \forall k = 1, \dots, R+1 \quad (21)$$

We have that, for NPND type customers, a bucket represents all the stops  $s$  such that  $\alpha[s_k(1 \text{ or } C)] \leq \alpha(s) < \alpha[s_{k+1}(1 \text{ or } C)]$  for any allowable  $k$  as described in Equation (21).

## 5.4 Insertion procedure

### *PD type*

PD type requests do not need any insertion procedure since both pick-up and drop-off points are checkpoints and they are already part of the schedule. However, once the PD type customers are onboard, they are important in evaluating the COST of any other insertion.

### *PND type*

PND type customers need to have their ND stop  $q$  inserted in the schedule. The algorithm checks for insertion's feasibility in the buckets of the P checkpoint. Since the ND stop can not be scheduled before P, the first bucket to be examined is the one starting with the first occurrence of P following the current position of the vehicle, that is the bucket delimited by  $s_k(P)$  and  $s_{k'+1}(P)$ , with  $k' = \min_k s_k(P)$ , s.t.  $t_{s_k(P)} \geq t_{now}$ . Among the feasible insertions between all pairs of consecutive stops  $a, b$  in the first bucket, the algorithm selects the one with the minimum COST and then stops. The customer is therefore scheduled to be picked up at  $s_k(P)$  and dropped off at the ND inserted stop  $q$ . If no feasible insertions are found in the first bucket, the algorithm repeats the procedure in the second bucket, assuming that the customer will be picked up at the beginning of it corresponding to the following occurrence of P, that is  $s_{k'+1}(P)$ . This process is repeated bucket by bucket until at least one feasible insertion is found.

### *NPD type*

NPD type customers need to have their NP stop  $q$  inserted in the schedule. Similarly, the algorithm checks for insertion's feasibility in the buckets of the D checkpoint. The first bucket to be examined is the one delimited by the current position of the bus  $(x_b, y_b)$  and the first occurrence of D following the current position of the bus, that is  $s_k(D)$ , with  $k' = \min_k s_k(D)$ , s.t.  $t_{s_k(D)} \geq t_{now}$ . In general,  $(x_b, y_b)$  does not correspond to a stop. Therefore, the first pair of points, between which the algorithm checks for feasibility, is represented by  $(x_b, y_b)$  and the first stop to be visited afterwards, as shown in Figure 8. Among the feasible insertions in the first bucket, the algorithm selects the one with the

minimum COST and then stops. The customer is therefore scheduled to be picked up at the inserted NP stop  $q$  and dropped off at  $s_k(D)$ . If no feasible insertions are found in the first bucket, the algorithm repeats the procedure in the second bucket, forcing the customer to be dropped off at the end of the second bucket, corresponding to the following occurrence of  $D$ ,  $s_{k+1}(D)$ . This process is repeated bucket by bucket until at least one feasible insertion is found.

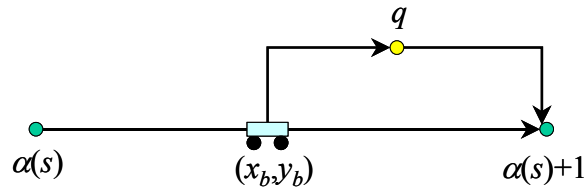


Figure 8 – Insertion from current vehicle position

### ***NPND type***

A NPND type customer requires the insertion of two new stops  $q$  and  $q'$ ; therefore the insertion procedure will be performed by a  $O(TS^2)$  procedure, meaning that for each feasible insertion of the NP stop  $q$ , the algorithm checks feasibility for the ND stop  $q'$ . A NPND feasibility is granted when both NP and ND insertions are simultaneously feasible. The search for NPND feasibility is performed with the additional constraint of having  $q$  scheduled before  $q'$ .

Recall that buckets correspond to the trips for a NPND type customer. The search for NPND feasibility is performed in at most two consecutive buckets meaning that when checking for NP insertion feasibility in bucket  $i$  and  $i+1$ , the algorithm looks for ND insertion feasibility only in bucket  $i$  and  $i+1$ .

The algorithm starts checking the NPND feasibility in the first bucket delimited by the current position of the bus  $(x_b, y_b)$  and the end of the current trip  $r$ . This is the first occurrence in the schedule of one of the terminal checkpoints  $s = 1$  or  $s = C$ , namely  $s_k(1 \text{ or } C) = \min_k s_k(1 \text{ or } C)$ , s.t.  $t_{s_k(1 \text{ or } C)} \geq t_{now}$ . Among all feasible NPND insertions in the first bucket, the algorithm selects the one with the minimum COST. If no NPND feasibility is found, the algorithm will then check pairs of two consecutive buckets at a time, increasing the “checking-range” by one bucket at each step (buckets 1/2, then buckets

$2/3, \dots, i/i+1$ , etc.). While checking buckets  $i/i+1$ , we already know that NPND insertion is infeasible in bucket  $i$  (because it has been already established before in the procedure while checking buckets  $i-1/i$ ). Therefore, while NP insertion feasibility needs to be considered in both buckets (since NPND insertion infeasibility in bucket  $i$  does not prevent NP insertion to be feasible in  $i$ ), ND insertion needs to be checked only in bucket  $i+1$ . The procedure will continue until at least one NPND feasible insertion is found.

### ***Working with buckets***

The insertion procedure performed by bucket assures that the ride time of each customer is upper bounded. In addition, it attempts to minimize the waiting time of each request until their pick-up (measured by  $WT_1$ ) by placing them in the earliest bucket where feasibility is found, avoiding postponements of the whole ride as much as possible.

### ***Rejection policy***

The general assumption while performing the insertion procedure is a no-rejection policy from both the MAST service and the customers. Thus, the algorithm attempts to insert the customer requests checking, if necessary, the whole existing schedule bucket by bucket, and rejection may occur only if there is no feasibility at all due, for example, to an over saturated system or when a customer request arrives towards the end of the service. However, in our simulation (Section 6), for MAST systems below or up to saturation level, rejected customers are only a very small percentage (due to end of the service) and they are removed from consideration. In addition, customers are assumed to never reject the insertion proposed by the algorithm and there is no negotiation between the MAST system and the customers.

## **5.5 Update procedure**

Once a minimum COST feasible insertion is selected, a new stop  $q$  (either a NP or a ND request) has been successfully scheduled between two points  $a$  and  $b$  in a portion of the schedule delimited by checkpoints  $m$  and  $m+1$ , and the variables of the system need to be updated.

The slack time will be updated as follows:

$$st_{m,m+1} = st_{m,m+1} - \Delta t_{a,q,b} \quad (22)$$

The departure and arrival times will also be updated (delayed) as follows:

$$t_s = t_s + \Delta t_{a,q,b} \quad \forall s \text{ s.t. } \alpha(s) \in [\alpha(b), \alpha(m+1)) \quad (23)$$

$$t'_s = t'_s + \Delta t_{a,q,b} \quad \forall s \text{ s.t. } \alpha(s) \in [\alpha(b), \alpha(m+1)] \quad (24)$$

Since the departure times  $t_s$  of checkpoints ( $\forall s \leq TC$ ) are constraints of the system and act as “time-barriers”, all the stops that are not in the portion of the schedule where the insertion takes place (between  $m$  and  $m+1$ ) are not affected. We can therefore identify six different cases:

- Customers having both pick-up and drop-off stops scheduled before  $q$  are not affected by the insertion.
- Customers having their pick-up stop before  $q$  and their drop-off stop in between  $q$  and  $m+1$  will have their ride time increased because their drop-off stop will be delayed as given by Equation (24).
- Customers having their pick-up stop before  $q$  and their drop-off stop after  $m+1$  will not be affected by the insertion because the departure time  $t_{m+1}$  will remain unchanged.
- Customers having both their pick-up and drop-off stops in between  $q$  and  $m+1$  will have both of them delayed by the same amount as given by Equations (23) and (24). Therefore, their waiting time at the pick-up stop will be increased but their ride time will remain unchanged.
- Customers having their pick-up stop in between  $q$  and  $m+1$  and their drop-off stop after  $m+1$  will have their waiting time at the pick-up stop increased as given by Equation (23) and their ride time decreased by the same amount because their drop-off stop will not be affected.

- Customers having both their pick-up and drop-off stops after  $m+1$  will not be affected.

### ***Time windows***

The algorithm provides customers at the time of the request with time windows for their pick-up and drop-off locations. To do so, it computes the earliest departure time  $et_q$  from  $q$  as follows:

$$et_q = t_a + d_{a,q}/v + h_q \quad (25)$$

where  $t_a$  represents the current departure time from stop  $a$ . Also the departure time of  $q$  is initialized likewise:

$$t_q = t_a + d_{a,q}/v + h_q = et_q \quad (26)$$

It can be easily shown that  $et_q$  is a lower bound for any further updates of  $t_q$ .

The algorithm then computes the latest departure time from  $q$ ,  $lt_q$ , as follows:

$$lt_q = et_q + st_{m,m+1} \quad (27)$$

We prove that  $lt_q$  is an upper bound for  $t_q$  by the following contradiction argument. Let's use the superscript  $\beta$  (with  $\beta = 0, \dots, f$ ) to indicate the  $\beta^{\text{th}}$  update of a variable and suppose that  $t_q^{(f)} > lt_q$ , we have  $t_q^{(f)} - t_q^{(0)} > lt_q - t_q^{(0)}$ . We also know by Equation (23) that:

$$\begin{aligned} t_q^{(f)} - t_q^{(0)} &= (t_q^{(f)} - t_q^{(f-1)}) + \dots + (t_q^{(\beta)} - t_q^{(\beta-1)}) + \dots + (t_q^{(1)} - t_q^{(0)}) = \\ &= \Delta t_f + \dots + \Delta t_\beta + \dots + \Delta t_1 = \sum_{k=1}^f \Delta t_k \end{aligned} \quad (28)$$

and from Equations (26) and (27),  $lt_q - t_q^{(0)} = lt_q - et_q = st_{m,m+1}$ , but this would imply

$\sum_{k=1}^f \Delta t_k > st_{m,m+1}$ , meaning that the sum of the extra time needed for insertions after the

insertion of  $q$  had exceeded the total slack time available after the insertion of  $q$  and this is a contradiction since the feasibility check would have prevented this from happening. Therefore Equation (27) says that future possible insertions between  $m$  and  $q$  will delay  $t_q$  to a maximum total amount of time bounded by the currently available slack time.

In a similar fashion, the earliest and latest arrival times,  $et'_q$  and  $lt'_q$ , are computed. As a result, the customer, once accepted, is provided with  $et_q$ ,  $lt_q$ ,  $et'_q$  and  $lt'_q$ , being aware that their actual times  $t_q$  and  $t'_q$  will be bounded by these values:

$$et_q \leq t_q \leq lt_q \tag{29}$$

$$et'_q \leq t'_q \leq lt'_q \tag{30}$$

While a P request has  $et_P = t_P = lt_P$  because the departure time from a checkpoint is a constant in a MAST system, a D request will have  $et'_D \leq t'_D \leq lt'_D$ . Clearly NP and ND requests will also have  $et_{NP} \leq t_{NP} \leq lt_{NP}$  and  $et'_{ND} \leq t'_{ND} \leq lt'_{ND}$ .

## 6 Experimental results

In this section we discuss the results obtained by simulation analysis. The target is to show that the insertion heuristic developed in this paper can be used as an efficient scheduling tool for real MAST systems. We compare the performance of the algorithm to a simple First-Come/First-Serve policy and we progressively improve the insertion heuristic effectiveness by modifying the values of the control parameters. Then the algorithm is compared to optimality assuming a static scenario.

### 6.1 System saturation and performance measures

The saturation level is the maximum demand  $\theta$  that a system configuration can satisfy without becoming unstable. We have that:

- WT<sub>i</sub>: average time interval between request/show up and earliest pick-up time ( $et_P$  or  $et_{NP}$ ) per customer, as mentioned in Section 5.2.

- PST: percentage of the total initial slack time ( $= \sum_{s=1}^{TC-1} st_{s,s+1}^{(0)}$ ) consumed

The saturation level can be estimated by looking at the  $WT_1$  values. Given that the demand is uniform over time, for systems well below their saturation level, the  $WT_1$  values should be around half the headway of the system. A slightly larger value of  $WT_1$ , but constant over the simulation time, shows that the system is near the saturation level, but still below it. Even if a few customers have to wait longer to be picked up due to temporary congestions created by the randomness of the demand, the system on average is stable. If instead the  $WT_1$  value increases over the simulation time, then the system is unstable and the demand rate is above the saturation level. An indication of how much the demand rate is below the saturation level may be given by the PST; values around 90% indicate that the demand rate is more or less at saturation level, for an insertion algorithm. In addition, since the slack time consumption is directly proportional to the miles driven, the PST and M values are related to each other. Therefore, bigger values of M also indicate a higher level of saturation.

We slightly redefine the objective function Z introduced in Section 3.2 to conform it to a dynamic environment by modifying the waiting time term in accordance to the COST function of Equation (17). Therefore we have

$$Z = w_1 \times M/v + w_2 \times RT \times NC + w_3 \times WT_E \times NC \quad (31)$$

where  $WT_E$  is the average extra waiting time ( $t - et$ ) per customer served by the system.

## 6.2 Algorithm performance

As noted in Section 3, the MAST system modeled in this paper is based on the existing MTA Line 646, which has very little slack time ( $st_{s,s+1}^{(0)} = 2.5$  minutes,  $\forall s = 1, \dots, TC-1$ , with a  $v = 25$  miles/hour; therefore about 5 minutes per trip), and very low demand (4-5 customers/hour, mostly PND and NPD types). Since MTA is interested in testing the MAST concept for higher demand levels, we assume a larger slack time for our simulation

experiments in order to allow the system to accommodate more insertion requests and evaluate the performance of the insertion algorithm. The rest of the data are consistent with MTA Line 646. A summary of the parameters values that are used in the experiments are shown in Table 2.

Table 2 – System parameters

L	10 miles
W	1 mile
C	3
$d_{s,s+1}$ ( $s = 1, \dots, TC-1$ )	5 miles
$t_{s+1} - t_s$ ( $s = 1, \dots, TC-1$ )	25 min ( $t_1 = 0$ )
$v$	25 miles/hour
$h_s$ ( $s = 1, \dots, TS$ )	18 sec
$w_1 / w_2 / w_3$	0.25 / 0.25 / 0.5

From Equation (1) we compute the values of the initial slack times  $st_{s,s+1}^{(0)} = 12.7$  minutes ( $s = 1, \dots, TC-1$ ) that are about 50% of the time intervals between two consecutive checkpoints' departure times ( $t_{s+1} - t_s = 25$  minutes).

In setting the COST function's weights, we assume that customers perceive the extra waiting time at stops ( $w_3$ ) with more discomfort than the ride time on the vehicle ( $w_2$ ) and that slack time consumption ( $w_1$ ) and passengers' ride time ( $w_2$ ) are equally weighted.

Given a constant demand rate  $\theta$ , we also assume that the customer types are distributed as shown in Table 3:

Table 3 – Customer type distribution

Type	PD	PND	NPD	NPND
%	10%	40%	40%	10%

The above distribution assumes that most of the customers need to be transported from a checkpoint to a desired location (home/office) and vice versa (PND and NPD types) as actually is the case for Line 646. We further assume that the checkpoint requests (P and D) are uniformly distributed among the C checkpoints and that non-checkpoint requests (NP and ND) are uniformly distributed in the service area. The simulation is run for 50

hours. We verified that this length of simulation time was sufficiently long to have all the performance parameters converge to their steady-state values for stable systems. According to the parameter values shown in Table 2, we have  $R = 60$ .

We first perform a set of runs setting the control parameters  $BACK = L$  and  $\pi_{s,s+1}^{(0)} = 1$  ( $\forall s = 1, \dots, TC-1$ ) allowing any backtracking and any slack time consumption if available, thus giving the most freedom to the algorithm when checking for insertion feasibility. At these parameter settings (configurations A) we seek the saturation level of the system, by examining the  $WT_I$  and PST values for different values of  $\theta$ .

We compare the results to the saturation level of a straightforward First-Come/First-Serve (FCFS) policy, where each request is placed one by one in the schedule in the earliest current feasible spot and then never moved nor postponed, behaving like hard constraints of the system. The results are shown in Table 4.

Table 4 – Saturation level for the FCFS policy and configurations A

Configuration	FCFS	A1	A2	A3
$\theta$ (customers/hour)	<b>2</b>	15	<b>20</b>	25
BACK (miles)	/	L	<b>L</b>	L
$\pi_{s,s+1}^{(0)}$ $s = 1, \dots, TC-1$	/	1	<b>1</b>	1
$WT_I$ (min)	<b>68.16</b>	56.52	<b>61.67</b>	236.74
PST (%)	<b>25.7%</b>	81.3%	<b>91.3%</b>	98.9%
saturation level?	<b>yes</b>	below	<b>yes</b>	above
$WT_E$ (min)	<b>0.00</b>	1.07	<b>1.23</b>	1.75
RT (min)	<b>24.44</b>	23.86	<b>25.86</b>	30.39
M (miles)	<b>742.6</b>	1012.7	<b>1051.4</b>	1083.8

The system scheduled by the FCFS policy reaches saturation (looking at the  $WT_I$  value, already over half the headway) with only  $\theta = 2$  customers/hour, even with little slack time used (a PST value of about 25% only). Although the FCFS policy is very simple to implement, it is very inefficient. The spatial and hard time constraints imposed by the myopic assignments of spots in the schedule do now allow this policy to make a wise use of the slack time and force the system to reach saturation very easily.

The saturation level of the configurations A is instead around  $\theta = 20$  customers/hour (configuration A2). While A1 is a stable system relatively far from saturation

(PST = 81.3%), A2 is right at the boundary because the  $WT_1$  value is higher than half the headway (50 minutes), but it does not increase over time. Hence, the system is stable, but since the slack time consumption is very high (PST = 91.3%), it is near the demand limit. Anything above  $\theta = 20$  would lead to system instability as shown by the results from A3, where the  $WT_1$  value is very high and keeps increasing with the simulation run time and the PST is close to 100%.

Therefore, Table 4 shows that the system benefits drastically by using the insertion heuristic algorithm instead of a FCFS policy. In addition, MTA Line 646 would be able to serve a demand  $\theta$  with up to 20 customers/hour by allowing more slack time in the schedule ( $st_{s,s+1}^{(0)} = 12.7$  minutes,  $\forall s = 1, \dots, TC-1$ ) and setting  $BACK = L$  and  $\pi_{s,s+1}^{(0)} = 1$  (configurations A), assuming a customer type distribution given by Table 3.

Now, keeping the demand at the saturation level (configuration A2), we want to observe the effect of modifying the usable slack time  $st_{s,s+1}^u$ . For this purpose, maintaining  $BACK = L$ , we vary the values of  $\pi_{s,s+1}^{(0)}$  ( $\forall s = 1, \dots, TC-1$ ) in the range from 1 to  $\pi_{s,s+1}^{(0)\min}$  (configurations B) to observe the effect of this control parameter. We compare the performances of each case by means on the object function,  $Z$ , as defined in Equation (31). The simulation run time is again 50 hours. Each configuration is tested with exactly the same demand using CNR (Common Random Numbers). The results are summarized in Table 5. From Equation (11),  $\pi_{s,s+1}^{(0)\min}$  is approximately equal to 0.22.

Table 5 – Effect of  $\pi_{s,s+1}^{(0)}$  - configurations B

Configuration	B1 = A2	B2	B3	B4	<b>B5</b>	B6
$\theta$ (customers/hour)	20	20	20	20	<b>20</b>	20
BACK (miles)	L	L	L	L	<b>L</b>	L
$\pi_{s,s+1}^{(0)}$ $s = 1, \dots, TC-1$	1	0.75	0.5	0.4	<b>0.3</b>	$\pi_{s,s+1}^{(0)\min}=0.22$
WT <sub>I</sub> (min)	61.67	55.87	54.59	51.56	<b>52.26</b>	51.60
PST (%)	91.3%	87.4%	82.3%	79.2%	<b>76.6%</b>	72.0%
saturation level?	yes	below	below	below	<b>below</b>	below
WT <sub>E</sub> (min)	1.23	1.15	1.25	1.32	<b>1.41</b>	1.37
RT (min)	25.86	24.68	24.13	23.09	<b>22.60</b>	22.76
M (miles)	1051.4	1021.7	989.0	968.2	<b>951.5</b>	921.7
Z	7149	6987	6853	6624	<b>6533</b>	6551

The figures reveal the positive effect of decreasing  $\pi_{s,s+1}^{(0)}$  from 1 to almost  $\pi_{s,s+1}^{(0)\min}$ . All the performance parameters significantly improve their values, with the exception of WT<sub>E</sub>, showing initially a progress, but then a progressive worsening. Also the Z values gradually drop and reach their minimum value with configuration B5 at  $\pi_{s,s+1}^{(0)} \cong 0.3$ , slightly greater than  $\pi_{s,s+1}^{(0)\min}$ . Due to the increased efficiency of the algorithm, all the configurations drop well below their saturation levels. Note that configuration B6 has lower PST and M values, indicating a better performance in terms of the slack time consumption, but the overall performance Z shows a worsening of the service quality with respect to B5. These results show the benefit of controlling the consumption of slack time and saving some of it for future insertions.

Now, starting from configuration B5, we would like to observe the effect of limiting the backtracking distance. We perform another set of simulations (configurations C), keeping  $\theta = 20$  and  $\pi_{s,s+1}^{(0)} = 0.3$  and varying the BACK parameter from L to 0. The results are shown in Table 6.

Table 6 – Effect of BACK - configurations C

Configuration	C1 = B5	C2	C3	C4	C5	<b>C6</b>	C7	C8
$\theta$ (customers/hour)	20	20	20	20	20	<b>20</b>	20	20
BACK (miles)	L	1.5	0.8	0.5	0.3	<b>0.2</b>	0.1	0
$\pi_{s,s+1}^{(0)}$ $s = 1, \dots, TC-1$	0.3	0.3	0.3	0.3	0.3	<b>0.3</b>	0.3	0.3
WT <sub>I</sub> (min)	52.26	52.26	52.35	51.70	52.19	<b>52.23</b>	51.28	51.84
PST (%)	76.6%	76.6%	75.8%	74.2%	72.8%	<b>72.4%</b>	71.2%	70.9%
saturation level?	below	below	below	below	below	<b>below</b>	below	below
WT <sub>E</sub> (min)	1.41	1.41	1.39	1.38	1.37	<b>1.37</b>	1.42	1.43
RT (min)	22.60	22.60	22.62	22.46	22.34	<b>22.28</b>	22.36	22.94
M (miles)	951.5	951.5	946.4	936.1	927.2	<b>924.2</b>	916.8	914.5
Z	6533	6533	6528	6478	6435	<b>6419</b>	6451	6596

There are no changes in the performance by lowering the value of the BACK parameter from L (configuration C1) down to about 1.5 miles (C2). This means that in the simulation there are no cases of an insertion with a backtracking distance bigger than 1.5 miles. Therefore, setting BACK to a value larger than 1.5 has no effect on the schedule. On the contrary, improvements in all the performance measures can be progressively seen in cases C3, C4, C5 and C6 (BACK = 0.8, 0.5, 0.3 and 0.2) while C7 and C8 (BACK = 0.1 and 0) show better values for PST and M, but the overall performance Z slightly worsens due to the increasing values of WT<sub>E</sub> and RT. All the cases are well below their saturation level and the best configuration according to Z is found by setting BACK = 0.2 miles, corresponding to case C6. These experiments illustrate the positive effect of limiting to a certain degree the amount of backtracking that the vehicle is allowed to do.

Case C6 represents a better configuration than A2 with respect to the overall performance Z and almost all the other parameters (with the exception of WT<sub>E</sub>, slightly increased). In particular, the improved efficiency of the algorithm causes the M and PST values to drop and the system is now well below saturation. We therefore look for the new saturation level for these more efficient parameter settings by performing another set of runs (configurations D, see Table 7) starting from configuration C6 and progressively increasing  $\theta$ .

Table 7 – New saturation level - configurations D

Configuration	D1 = C6	<b>D2</b>	D3
$\theta$ (customers/hour)	20	<b>25</b>	30
BACK (miles)	0.2	<b>0.2</b>	0.2
$\pi_{s,s+1}^{(0)}$ $s = 1, \dots, TC-1$	0.3	<b>0.3</b>	0.3
WT <sub>I</sub> (min)	52.23	<b>55.98</b>	77.58
PST (%)	72.4%	<b>86.8%</b>	95.9%
saturation level?	below	<b>yes</b>	above
WT <sub>E</sub> (min)	1.37	<b>1.72</b>	1.92
RT (min)	22.28	<b>23.93</b>	29.00
M (miles)	924.2	<b>983.4</b>	1020.6

As done for configurations A, we can estimate the saturation level for configurations D by looking at the stability of the WT<sub>I</sub> value over the simulation time. The figures show that  $\theta = 25$  customer/hour (D2) approximately represent the limit for the system. Anything above this value would cause instability.

Therefore, the adjustments made on the control parameters allow the insertion heuristics to handle a demand rate 25% larger than the initial configuration A2 (and drastically larger than the demand that a FCFS policy would be able to handle)

### 6.3 Comparison vs. optimality

We now provide an evaluation of the insertion heuristic algorithm by comparing its performance against optimality computed by CPLEX 9.1, a commercial integer program solver. In order to perform this task we consider static cases, with all the demand known in advance, and therefore we can use  $Z$  defined by Equation (5) to measure the performance. However we need to slightly revise the COST function defined in Equation (17), modify the waiting time term to match the  $Z$  for the static case. Thus,

$$\text{COST} = w_1 \times \Delta t_{a,q,b} + w_2 \times \Delta RT + w_3 \times \Delta WT \quad (32)$$

where  $\Delta WT$  represents the sum over all passengers of the total waiting time, similarly to WT. Also the weights are redefined, because the waiting time measured here is of different

nature. In this case we assume that customers would rather wait for their pick-up instead of spending time onboard the vehicle setting  $w_1 = w_2 = 0.4$  and  $w_3 = 0.2$ .

We run two sets of experiments: in set 1 we assume the system parameters of Table 2 (except the weights). In set 2 we use the same data except for the difference between the scheduled departure times of consecutive checkpoints ( $t_{s+1} - t_s, \forall s = 1, \dots, TC-1$ ) being 17.5 minutes instead of 25 minutes. As a result the slack time for set 2 is approximately 25% instead of 50%.

In each set we consider two different cases. In cases 1b and 2b we assume larger number of trips R compared to cases 1a and 2a. We assume a different number of requests of each type, as shown in the following Table 8, trying to maintain the ratio between the different types of requests as close as possible to the real data of MTA line 646 in Los Angeles, which have a distribution described in Table 3. The NP and ND locations are sampled from a spatial uniform distribution over the whole service area ( $W \times L$ ); while the ready times are sampled from a uniform distribution starting from half an hour before the beginning of the service to the end of it.

Table 8 – System parameters specific to each case

Parameters \ Cases	1a	1b
	2a	2b
R	4	6
TC	9	13
PD	2	1
PND	6	8
NPD	6	7
NPND	2	1
TS	25	30

The size of these cases in terms of R and the demand (and therefore TS) is considerably smaller compared to the experiments performed in Section 6.2, in order to find an optimal solution. All the runs are performed utilizing CPLEX 9.0 with default settings on a 3.2 GHz CPU with 2GB RAM and we allowed a maximum CPU time of 10 hours for each case. In Table 9 for each case we provide the Z value obtained by the insertion heuristic and by CPLEX. For the heuristic results we show the Z obtained with no

control and with the best setting of the control parameters found for each case (if any). The CPLEX results show the optimal value (opt), when reached, the upper (ub) and lower (lb) bounds and the corresponding gap.

Table 9 – Heuristic vs. optimality

	Heuristic				CPLEX			
	no control ( $\pi_{s,s+1}^{(0)} = 1$ ; BACK = L)	best control						
case	Z	Z	$\pi_{s,s+1}^{(0)}$	BACK (miles)	opt	ub	lb	gap
1a	323.1	314.1	0.3	0.2	?	312.8	293.0	6.3%
1b	344.1	332.8	0.9	5	?	332.8	278.7	16.3%
2a	242.4	already optimal			242.4	/	/	0.0%
2b	294.1	no improvement			?	294.1	274.7	6.6%

The figures show that in cases 1a and 1b the heuristic with no control reaches the Z values of 323.2 and 344.1 respectively that are higher than the upper optimality bounds found by CPLEX (312.8 and 332.8 correspondingly); a proper setting of the control parameters allows to improve the solutions substantially, down to 314.1 and 332.8 respectively. In case 2a the heuristic reaches the optimal value of 242.4 even with the default settings of the control parameters ( $\pi_{s,s+1}^{(0)} = 1$  and BACK = L). In case 2b the heuristic reaches a Z value of 294.1 (corresponding to the upper optimality bound) with the default values of the control parameters and we could not improve the result by modifying them.

## 7 Conclusions

In this paper we presented an insertion heuristic for scheduling Mobility Allowance Shuttle Transit (MAST) services. The algorithm allows customers to place a request, and once accepted, provides them with time windows for both pick-up and drop-off points. Due to the dynamic nature of the environment, the algorithm makes effective use of a set of control parameters to reduce the consumption of slack time and enhance the algorithm performance. The results of simulations performed on a system representing the existing

MTA Line 646 of Los Angeles show the efficacy of the algorithm and its control parameters and demonstrate that the algorithm can be used as an effective method to automate scheduling of this line and other similar services. In addition, a comparison vs. optimality values computed by CPLEX in a static scenario shows that the results obtained by the heuristic are not far from optimum.

Future research on MAST systems could focus on improving the solution by introducing local search techniques, studying the system under different demand distributions and stochastic environments, finding the optimal slack time for a given demand distribution, developing heuristics for the multiple vehicle MAST system to handle daytime heavy demand environments and comparing MAST systems to conventional transportation services like fixed-route bus or DRT.

## **8 Acknowledgements**

The research reported in this paper was partially supported by the National Science Foundation under grant NSF/USDOT-0231665. We would also like to thank Operation Shuttle, Inc. for providing us with data on MTA Line 646.

## References

- Aldaihani M.M. and Dessouky, M. (2003), "Hybrid scheduling methods for paratransit operations", *Computers & Industrial Engineering*, 45, 75-96.
- Aldaihani, M.M., Quadrifoglio, L., Dessouky, M. and Hall, R.W. (2004) "Network design for a grid hybrid transit service", *Transportation Research*, 38A, 511-530.
- Bodin, L. and Sexton, T. (1986) "The multi-vehicle subscriber dial-a-ride problem", *TIMS Studies in the Management Sciences*, 22, 73-86.
- Campbell, A. M., and Savelsbergh, M. (2004) "Efficient insertion heuristics for vehicle routing and scheduling problems," *Transportation Science*, 38, 369-378.
- Crainic, T.G., Malucelli, F. and Nonato, M. (2001) "Flexible many-to-few + few-to-many = an almost personalized transit system", TRISTAN IV, São Miguel Azores Islands, 435-440.
- Daganzo, C.F. (1978) "An approximate analytic model of many-to-many demand responsive transportation systems", *Transportation Research*, 12, 325-333.
- Daganzo, C.F. (1984) "Checkpoint dial-a-ride systems", *Transportation Research*, 18B, 315-327.
- Desaulniers, G. et al. (2000) "The VRP with pickup and delivery", *Cahiers du GERARD G-2000-25*, Ecole des Hautes Etudes Commerciales, Montréal.
- Desrosiers, J., Dumas, Y. and Soumis, F. (1986) "A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows", *American Journal of Mathematical and Management Sciences*, 6, 301-325.
- Desrosiers, J., Dumas, Y. and Soumis, F. (1988) "The multiple dial-a-ride problem", *Lecture Notes in Economics and Mathematical Systems* 308: Computer-Aided Transit Scheduling. Springer, Berlin.
- Dessouky, M., Ordóñez, F. and Quadrifoglio, L. (2005) "Productivity and cost-effectiveness of demand responsive transit systems", PATH/Caltrans Technical Report. Berekely, CA.
- Dessouky, M., Rahimi, M., and Weidner, M. (2003) "Jointly Optimizing Cost, Service, and Environmental Performance in Demand-Responsive Transit Scheduling," *Transportation Research Part D: Transport and Environment*, 8, 433-465.
- Diana, M. and Dessouky, M. (2004) "A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows", *Transportation Research*, 38B, 539-557.
- Hickman, M. and Blume, K. (2000) "A method for scheduling integrated transit service", 8<sup>th</sup> International Conference on Computer-Aided Scheduling of Public Transport (CASPT), Berlin, Germany.
- Jaw, J.J. et al. (1986) "A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows", *Transportation Research*, 20B(3), 243-257.
- Liaw, C.F., White, C.C. and Bander, J.L., (1996) "A decision support system for the bimodal dial-a-ride problem", *IEEE Transactions on Systems, Man, and Cybernetics*, 26(5), 552-565.
- Lu, Q. and Dessouky, M. (2004) "An exact algorithm for the multiple vehicle pickup and delivery problem," *Transportation Science*, 38, 503-514.

- Lu, Q. and Dessouky M. (2006) "New insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows", to appear *European Journal of Operational Research*.
- Madsen, O.B.G., Raven, H.F. and Rygaard, J.M. (1995) "A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives", *Annals of Operations Research*, 60, 193-208.
- Malucelli, F., Nonato, M. and Pallottino, S. (1999) "Demand adaptive systems: some proposals on flexible transit", *Operations Research in Industry*, T. Cirania, E. Johnson, and R. Tadei (eds), 157-182.
- Palmer, K, Dessouky, M. and Abdelmaguid, T. (2004) "Impact of management practices and advanced technology on demand responsive transit systems", *Transportation Research, Part A: Policy and Practice*, 38, 495-509.
- 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. (1986) "Scheduling large-scale advance-request dial-a-ride systems", *American Journal of Mathematical and Management Sciences*, 6, 327-367.
- Savelsbergh, M.W.P. and Sol, M. (1995) "The general pickup and delivery problem", *Transportation Science*, 29, 17-29.
- Sexton, T.R. and Bodin, L.D. (1985a) "Optimizing single vehicle many-to-many operations with desired delivery times: 1. Scheduling", *Transportation Science*, 19, 378-410.
- Sexton, T.R. and Bodin, L.D. (1985b) "Optimizing single vehicle many-to-many operations with desired delivery times: 2. Routing", *Transportation Science*, 19, 411-435.
- Sexton, T.R. and Choi, Y. (1986) "Pickup and delivery of partial loads with soft time windows", *American Journal of Mathematical and Management Sciences*, 6, 369-398.
- Toth, P., and Vigo, D. (1997) "Heuristic algorithm for the handicapped persons transportation problem", *Transportation Science*, 31, 60-71.
- Wilson, N.H.M. et al. (1971) "Scheduling algorithms for a dial-a-ride System", M.I.T, Urban Systems Laboratory, Technical Report USL TR-70-13.
- Wilson, N.H.M., and Hendrickson, C. (1980) "Performance models of flexibly routed transportation services", *Transportation Research*, 14B, 67-78.
- Zhao, J. and Dessouky, M. (2005) "Optimal service capacity for a single bus mobility allowance shuttle transit (MAST) system", submitted for publication.