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 bus systems. A MAST system allows buses to deviate from the fixed path so that customers within the service area may be picked up or dropped off at their desired locations. Such a service already exists in Los Angeles County. Line 646 operates during nighttime, transporting passengers between a business area and a nearby bus terminal; since the current demand of the service is very low, the service is entirely manageable by the bus operator, but a higher demand would certainly require the help of information technologies by means of a scheduling algorithm. We carry out a set of simulations to show the performance of the algorithm in the service area covered by the existing MTA Line 646 at different demand levels. 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

Demands on transit agencies for improved and extended services are increasing. On the other hand, 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 bus transit systems fall into two broad categories: fixed-route and demand responsive transit (DRT) systems. According to the National Transit Summaries and Trends (NTST) report for 2000, the average cost per passenger trip for fixed-route bus systems in the U.S. is $2.19 and a typical bus fare is around $0.50. On the contrary, for DRT systems, the cost per passenger trip is $16.74, while a typical DRT fare is around $2-3. Therefore, since DRT systems tend to be much more costly to deploy, they are largely limited to specialized operations such as Dial-a-Ride service mandated under the Americans with Disabilities Act (paratransit DRT). Fixed-route bus systems are instead much more cost efficient and they require less government subsidy. This is primarily due to the passenger loading capacity of the buses and the consolidation of many passenger trips onto a single vehicle (ridesharing).

However fixed-route bus transit systems, as an alternative to private automobiles, have major deficiencies. The general public considers the service to be inconvenient because of its lack of flexibility since either the locations of pick-up and/or drop-off points or the service's schedule don't match the individual rider's desires. Moreover, the total trip time is perceived as being too long and for longer trips there's often a need of transfers between vehicles.

On the other hand, commercial demand responsive transit (DRT) systems, such as taxicabs and shuttle vans, provide much of the desired flexibility, but these improvements in convenience come at a price. Taxicabs provide point-to-point pick-up and drop-off, and near real-time scheduling; however, the cost per trip is not affordable on a regular basis for most people. Shuttle vans provide flexible pick-up points. However, drop-off points are limited to popular locations and often advanced scheduling is required. These restrictions on flexibility allow the shuttle vans to guarantee sufficient ridesharing to operate at a

reduced cost per trip compared to that of taxicabs; even so, shuttle vans are still a lot less cost efficient than fixed-route bus transit systems.

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, buses are allowed to deviate from the fixed path to pick up and drop off passengers at their desired locations. The MAST system works under a dynamic environment since the majority of the requests occur while the bus is on duty (even though reservations in advance are handled by the system). The only restriction on flexibility is that the deviations must lie within a predetermined distance from the fixed base route.

Such a system somewhat 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.

Since the current demand of line 646 is low, the bus operator is able to make all the decisions concerning accepting/rejecting customer requests and routing the vehicle. Clearly, in case of heavier demand and several requests for deviations, the operator will not be able to handle this task efficiently by himself/herself. An effective MAST system needs to rely on recent developments in communication and computation technologies that allow

real-time information about pick-up/drop-off requests to be used to re-route the bus by means of a scheduling algorithm.

There has been a significant amount of research in scheduling DRT systems, but we are unaware of any work performed on scheduling systems such as the MAST service. Although these two types of systems are related, DRT systems focus strictly on point-to-point transport services, while 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 shuttle arrive at checkpoints on or before their scheduled time.

The purpose of this paper is to address this gap in the research community by developing 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, 2003)

The remainder of this paper is divided into six 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 obtained by simulation. Section 7 provides conclusions.

2 Literature review

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. Due to the combinatorial nature of the problem, most of the research efforts focus on heuristic approaches.

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). Stein (1977, 1978a, 1978b) develops a probabilistic analysis of the problem and Daganzo (1978) presents a

model to evaluate the performance of a Dial-a-Ride system. Theoretical studies for the single-vehicle case include also the work by Psaraftis (1980, 1983a), Sexton and Bodin (1985a, 1985b), Sexton and Choi (1986), Desrosiers et al. (1986) for exact algorithms and the work of Psaraftis (1983b, 1983c) for heuristic approaches. The work of Daganzo (1984) is somewhat similar to the MAST system describing a checkpoint DRT system that combines the characteristics of both fixed route and door-to-door service. In a checkpoint system, a service request is still made but the pick-up and drop-off points are not at the door but at centralized locations called checkpoints. However, the MAST system differs from a checkpoint only system since it allows also for door-to-door requests.

Heuristics to solve multi-vehicle problems have been proposed by Psaraftis (1986), Jaw et al. (1986), Bodin and Sexton (1986) and Desrosiers et al. (1988). Min (1989) considers a vehicle routing problem with simultaneous pickups and deliveries that involves the definition of a capacity constraint. Dumas et al. (1991) present a column generation scheme for optimally solving the Pickup and Delivery Problem with time windows. Local search procedures are reported in Van Der Bruggen et al. (1993) and Healy and Moll (1995). Madsen et al. (1995) present an insertion heuristic. Ioachim et al. (1995) develop a clustering algorithm. A simulated annealing procedure is introduced by Hart (1996). A parallel insertion heuristic is proposed by Toth and Vigo (1997). Savelsbergh and Sol (1998) propose three approximation algorithms derived from their branch-and-price based optimal algorithm. Borndörfer et al. (1999) propose heuristics to solve a transportation problem of handicapped persons. Tabu search techniques have been applied by Nanry and Barnes (2000), Landrieu et al. (2001) and Cordeau and Laporte (2003). Teodorovich and Radivojevic (2000) use a fuzzy logic approach. Li and Lim (2001) propose a metaheuristic. Lao and Liang (2002) present a two-phase method. A parallel regret insertion heuristic is done by Diana and Dessouky (2003). Exact procedures to solve small problems can be found in Ruland and Rodin (1997) and Lu and Dessouky (2003). Feuerstein and Stougie (2001) investigate the best possible competitive ratio for an on-line single-server dial-a-ride problem. Uchimura et al. (2002) propose a hierarchical structure for demand responsive services. A simulation model for paratransit services can be found in Fu (2002).

Recent papers focus on the design of Dial-a-Ride services on a technologically advanced basis. Dial (1995) proposes the implementation of a decentralized control

strategy for a fleet of vehicles. Horn (2002b) develops an algorithm for the scheduling and routing of a fleet of vehicles that is embedded in a modeling framework for the assessment of the performance of a general public transport system with the latter being presented in Horn (2002a).

3 System definition

The MAST system analyzed in this paper consists of a single vehicle, initially associated with a predefined schedule along a fixed route, consisting of C checkpoints, identified by $c = 1, 2, \ldots, C$; 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. We consider a loop system, having the vehicle moving along the route back and forth between 1 and C. A ride 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 rides. Since the end-terminal of a ride r corresponds to the start-terminal of the following ride $r+1$, the total number of stops at checkpoints is TC = (C-1)R+1. Hence, the initial schedule's array is represented by an ordered sequence of stops $s = 1, \ldots, TC$ and their scheduled departure time t_s (with $t_{s+1} > t_s$) are assumed to be constraints of the system which can not be violated. We treat the departure times at the checkpoints as hard constraints since the checkpoints typically represent major transfer centers and late arrivals to these stops will result in passengers missing their connections.

The service area is represented by a rectangular region defined by L*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 1).

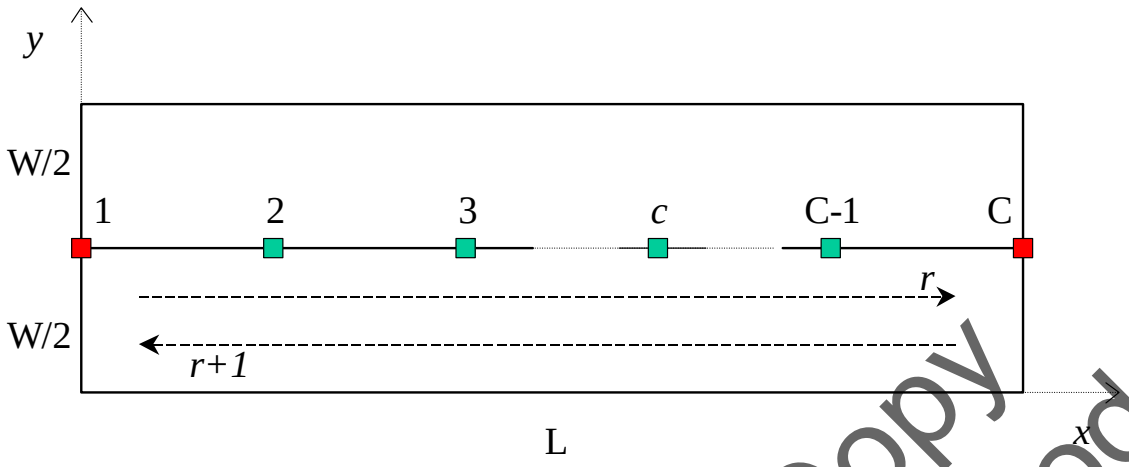

Figure 1 – MAST system

Customer types

The MAST system can respond to four different types of requests: regular pick-up (P) and regular drop-off (D) representing customers picked up/dropped off at checkpoints; non-checkpoint pick-up (NP) and non-checkpoint drop-off (ND) representing customers picked up/dropped off at any location in the service area. Hence, we can categorize the customers in four different types:

- PD: pick-up and drop-off at checkpoints
- PND: pick-up at checkpoint, drop-off at non-checkpoint
- NPD: pick-up at non-checkpoint, drop-off at checkpoint
- NPND: pick-up and drop-off at non-checkpoints

At any moment a customer may call in (or show up at checkpoints), specifying the locations of both pick-up and drop-off points, and the MAST algorithm will attempt to place the request in the schedule by means of an insertion procedure. We assume that customers are immediately ready to be picked up at the moment of their request. However, the system could easily handle reservations for future pick-ups by limiting the search for insertion in the portion of the schedule following the reservation time specified by the customer.

While checkpoints are identified by s = 1,…,TC non-checkpoint customer request (NP or ND) are identified by s = TC+1,…,TS where TS represents the current total number of stops (including checkpoints and non-checkpoints). The index $\alpha(s)$, s = 1,…,TS represents the current position of any stop s in the schedule. The problem is then to determine the indices $\alpha(s)$, s = 1,…,TS and the departure times t_s for non-checkpoint stops, s = TC+1, …,TS while not violating the given departure times t_s for checkpoint stops s = 1, …,TC.

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 slack time in the schedule. Let $st_{s,s+1}^{(0)}$ be the slack given by the schedule and is computed as follows:

$$st_{s,s+1}^{(0)} = t_{s+1} - t_s - d_{s,s+1}/bs - bt_{s+1} \qquad s = 1, \ldots, \text{NC-1} \qquad (1)$$

where bs is the vehicle speed, bt_{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 slack is reduced. Let $st_{s,s+1}$ be the available slack that can be used to route the bus off the base route. Initially,

$$st_{s,s+1} = st_{s,s+1}^{(0)} \qquad s = 1, \ldots, \text{NC-1} \qquad (2)$$

Idle policy

We assume that the bus, driving from checkpoint s to $s+1$, follows a no-idle policy until all the requests in between them have been satisfied. The unused slack time $st_{s,s+1}$ possibly still available when arriving to checkpoint $s+1$ is spent as idle time (note that while the bus is idle at $s+1$, new upcoming customer requests can still be inserted in the schedule before $s+1$ using $st_{s,s+1}$ if feasible and best at the moment, meaning that the bus leaves $s+1$ to serve the new requests and comes back to $s+1$ before t_{s+1}).

Arrival times

While t_s represents the scheduled departure time at stop s, we define t'_s as the arrival time at stop s. Because of the idle policy, we have for non-checkpoint stops (s > TC) $t_s = t'_s$ assuming that boarding and disembarking times are small and for checkpoint stops ($s \leq$ TC), $t_s \geq t'_s$ and their initial values are:

$$t'_{s+1} = t_{s+1} - st_{s,s+1} \qquad s = 1, \ldots, NC-1 \qquad (3)$$

Bus motion

We assume that the bus follows a *rectilinear motion*, allowing the vehicle to move only along the horizontal or vertical direction; this is a good approximation of the real world, since buses ride along streets which often form a grid.

Furthermore, whenever a horizontal or vertical direction can be equally chosen to reach the next scheduled stop, the bus prefers the one that keeps it closer to the central x axis of the service area. This behavior guarantees a better service to the future expected demand under the assumption of uniform distribution of non-checkpoint requests.

4 Control parameters

The challenge of operating a MAST system mainly resides in defining the logic to best operate the vehicle under a dynamic and 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 introduce the concept of "buckets" and make use of parameters that are a function of the slack time and the relative position of the new request with respect to the already scheduled stops.

4.1 Buckets

The MAST insertion algorithm does not explicitly add a constraint to limit the maximum allowable ride time of each customer as the Dial-a-Ride algorithms generally do.

Instead, it obtains a similar result working with "buckets". The underlying concept is that for PND and NPD type of customers one of the service points (either a P or a D checkpoint) is already part of the schedule; therefore, given the time of the customer request, the algorithm attempts to insert the corresponding ND and NP stops in the "vicinity" of the first occurrence of those checkpoints in the schedule's array. If not feasible, the algorithm checks for insertion in the "vicinity" of the following occurrences of the checkpoint of interest, one by one, till feasibility is found. Clearly, this postponement causes a delay on the whole trip, but the ride time will be upper bounded.

In order to define 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 bus a number of times, with different stop indices $s_k(c)$ (stop index of the k^{th} occurrence of checkpoint *c* in the schedule), depending on how many rides (R) are planned.

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

$$s_k(c) = \left\{ 1 + (C-1)(k-1) + \frac{(C-1) + (-1)^k [(C-1) - 2(c-1)]}{2} \right\} \qquad k = 1, \ldots, R \qquad (4)$$

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

$$s_k(1) = \{1 + 2(C-1)(k-1)\} \qquad k = 1, \ldots, 1+\lfloor R/2 \rfloor \qquad (5)$$

$$s_k(C) = \{C + 2(C-1)(k-1)\} \qquad k = 1, \ldots, \lceil R/2 \rceil \qquad (6)$$

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 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 (4), (5) and (6).

Table 1 – Schedule's array and buckets

ride	*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
	2C-1	1
3	2C	2
	…	…
	2(C-1)+1+(*c*-1)	*c*
	…	…
	3C-2	C
…	…	…
r	*r*(C-1)+1-(*c*-1)	*c*
	…	…
	r(C-1)+1	1
	…	…
r+1	*r*(C-1)+1+(*c*-1)	*c*
	…	…
	(*r*+1)(C-1)+1	C
…	…	…
R	TC=R(C-1)+1	1 or C

1st bucket of *c*=1

bucket of *c*=2

another bucket of *c*=2

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

$$s_k(1 \setminus C) = \{1 + (C-1)(k-1)\} \qquad k = 1, \ldots, R+1 \qquad (7)$$

We have that, for NPND type customers, a bucket represents all the stops *s* such that $\alpha[s_k(1\backslash C)] \leq \alpha(s) < \alpha[s_{k+1}(1\backslash C)]$ for any allowable *k* as described in equation (7).

4.2 Usable slack time

The slack time is a crucial resource needed to serve customers. When this resource is scarce, the system is not able anymore to properly satisfy new requests and it is forced to postpone or reject them. Therefore, a MAST service needs to be particularly careful about accepting customer requests that require a lot of slack time consumption preventing future requests from being fully satisfied. In fact, an insertion that appears to be good at the time of its placement in the schedule may very well not be so if we consider future expected customer requests. We therefore need to define a parameter that properly controls the consumption of slack time.

$st_{s,s+1}$ represents the current available unused slack time between two consecutive checkpoints *s* and *s+1*; while $st^{(0)}_{s,s+1}$ is the slack time initially available before any insertion has been performed. We define the *usable slack time* $st^{u}_{s,s+1}$ as the maximum amount of slack time that any customer request is allowed to consume for its insertion between *s* and *s+1*. It represents an upper bound on the usable amount of slack time and it prevents a single insertion from consuming too much of it. $st^{u}_{s,s+1}$ is defined as a function of the future expected demand between *s* and *s+1* and is not related to the actual unused slack time $st_{s,s+1}$ and therefore $st^{u}_{s,s+1}$ can be greater or lower than $st_{s,s+1}$ depending on the circumstances. As we will see in the insertion feasibility section, a request will be allowed to consume the minimum value among $st^{u}_{s,s+1}$ and $st_{s,s+1}$ for its insertion.

We assume that the demand rate λ (# of requests per unit time in the service area L*W) of non-checkpoint's requests (NP and ND) is uniformly distributed in the service area and constant over time. 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 *x* coordinate values of *s* and *s+1* with respect to the service area). Consequently, the expected

demand between s and $s+1$ (total # of insertion requests), $\Lambda_{s,s+1}$, is estimated as follows (see Figure 2):

$$\Lambda_{s,s+1} = \lambda \frac{|x_s - x_{s+1}|}{L} (t_{s+1}\text{-}t_s) \tag{8}$$

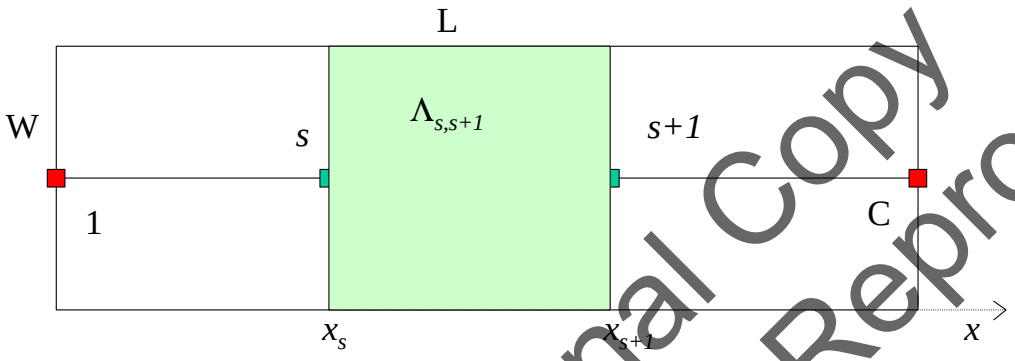

Figure 2 – Portion of service area covered by the segment between s and $s+1$

As soon as the vehicle departs from s at t_s, the expected residual demand drops linearly 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 3):

$$\Lambda_{s,s+1}^{(t_{now})} = \begin{cases} \Lambda_{s,s+1} & t_{now} < t_s \\ \Lambda_{s,s+1}\left(1 - \dfrac{t_{now} - t_s}{t_{s+1} - t_s}\right) & t_s \le t_{now} \le t_{s+1} \\ 0 & t_{now} > t_{s+1} \end{cases} \tag{9}$$

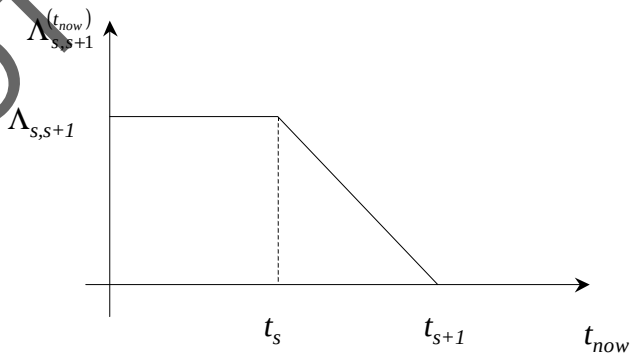

Figure 3 – Expected residual demand between s and $s+1$ as a function of t_{now}

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

$$\pi_{s,s+1} = 1 + \left(\frac{\pi_{s,s+1}^{(0)} - 1}{\Lambda_{s,s+1}} \right) \Lambda_{s,s+1}^{(t_{now})} \qquad \text{with } 0 \le \pi_{s,s+1}^{(0)} \le 1 \tag{10}$$

Since $0 \le \Lambda_{s,s+1}^{(t_{now})} \le \Lambda_{s,s+1}$, we have that $\pi_{s,s+1}^{(0)} \le \pi_{s,s+1} \le 1$ and $\pi_{s,s+1}^{(0)}$ can be set accordingly. We finally define the usable slack time, $st_{s,s+1}^{u}$, as follows:

$$st_{s,s+1}^{u} = \pi_{s,s+1} st_{s,s+1}^{(0)} \tag{11}$$

If the residual expected demand $\Lambda_{s,s+1}^{(t_{now})} \to 0$, then $\pi_{s,s+1} \to 1$ and $st_{s,s+1}^{u} \to st_{s,s+1}^{(0)}$. 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 (9), (10) and (11) we finally derive the expression for the *usable slack time*, $st_{s,s+1}^{u}$, as a function of t_{now} (see Figure 4):

$$st_{s,s+1}^{u} = \begin{cases} \pi_{s,s+1}^{(0)} st_{s,s+1}^{(0)} & t_{now} < t_s \\ \left[1 + \left(\pi_{s,s+1}^{(0)} - 1 \right) \left(1 - \dfrac{t_{now} - t_s}{t_{s+1} - t_s} \right) \right] st_{s,s+1}^{(0)} & t_s \le t_{now} \le t_{s+1} \\ st_{s,s+1}^{(0)} & t_{now} > t_{s+1} \end{cases} \tag{12}$$

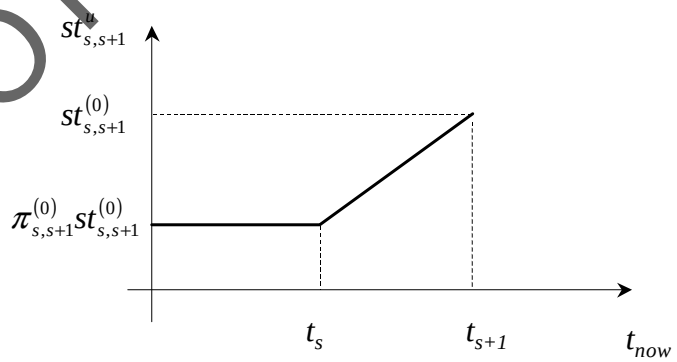

Figure 4 – *Usable slack time*

Let's now consider a non-checkpoint request q located at the edge of the service area, such that $y_q = 0$ or $y_q = \mathrm{W}$ and $x_s \le x_q \le x_{s+1}$ and let's assume that the schedule between s and $s+1$ is empty (no previously inserted stops). In order to be inserted, 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 ($st_q = \mathrm{W}/bs+bt_q$). Since the minimum amount of usable slack time from equation (12) is given by $st^u_{s,s+1} = \pi^{(0)}_{s,s+1}\, st^{(0)}_{s,s+1}$, we need to have $st^u_{s,s+1} \ge st_q$ to prevent the q request from being rejected. Hence we define:

$$\pi^{(0)\min}_{s,s+1} = (\mathrm{W}/bs+bt_q)/\, st^{(0)}_{s,s+1} \qquad (13)$$

as the minimum value of $\pi^{(0)}_{s,s+1}$ that guarantees every non-checkpoint request q to be considered for insertion between s and $s+1$ with the schedule empty, regardless of the location of q as long as $x_s \le x_q \le x_{s+1}$.

Setting $\pi^{(0)}_{s,s+1} < \pi^{(0)\min}_{s,s+1}$ would prevent the algorithm from working properly, because some customers would be rejected not because of system saturation or end of service, but because of improper parameter setting. Clearly, setting $\pi^{(0)}_{s,s+1} = 0$ would result in having $st^u_{s,s+1} = 0$ for $t_{now} < t_s$, preventing any requests before t_s from being considered for insertion. On the contrary, $\pi^{(0)}_{s,s+1} = 1$ causes $st^u_{s,s+1} = st^{(0)}_{s,s+1}$ at any time and customers requests would have no limit on the amount of slack time allowed to be consumed for their insertion.

A proper value of $\pi^{(0)}_{s,s+1}$ in between $\pi^{(0)\min}_{s,s+1}$ and 1 allows the system to control the consumption of slack time. Any request occurring before t_s can use at most the minimum value of $st^u_{s,s+1} = \pi^{(0)}_{s,s+1}\, st^{(0)}_{s,s+1}$ because there is an expected demand of future customers that should be properly served with the remaining slack time. Whereas, if a customer request occurs towards the end of the ride from s to $s+1$, it is allowed to consume a bigger portion of the slack time until a maximum of $st^{(0)}_{s,s+1}$ because the chance of having additional requests before the bus reaches the next checkpoint $s+1$ is very low.

4.3 Backtracking distance

The insertion procedure can cause the vehicle to drive back and forth with respect to the direction of a ride *r*, 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 bus drives backwards on the *x* axis while moving between two consecutive stops to either pick up or drop off a passenger at a non-checkpoint stop with respect to the direction of the current ride. More formally, as shown in Figure 5, 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 ride *r* (1→C or vice versa, parallel to the *x* axis) as follows:

$$bd_{a,b} = -\min(0, \hat{d}_r \cdot \hat{d}_{a,b}) \tag{14}$$

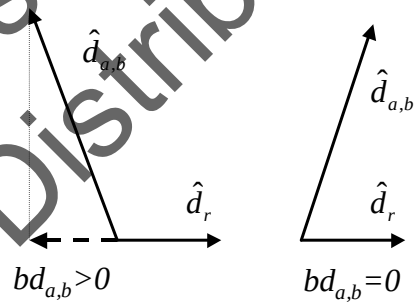

Figure 5 – Backtracking distance

The backtracking parameter (BACK > 0) is defined as the maximum allowable backtracking distance that the bus can ride between any two consecutive stops. BACK is a parameter and can be set accordingly; clearly with BACK ≥ L any backtracking is allowed.

5 Algorithm description

5.1 Feasibility

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

$$\Delta t_{a,q,b} = (d_{a,q}+d_{q,b}-d_{a,b})/bs - bt_q \quad (15)$$

Let checkpoints m and $m+1$ be the checkpoints prior and after stops a and b in the schedule. The algorithm computes also the backtracking distances $bd_{a,q}$ and $bd_{q,b}$ by equation (14). Finally, it is feasible to insert q between a and b if:

$$\begin{cases} \Delta t_{a,q,b} \le \min(st_{m,m+1}, st^u_{m,m+1}) & (16) \\ bd_{a,q} \le \text{BACK} & (17) \\ bd_{q,b} \le \text{BACK} & (18) \end{cases}$$

The algorithm doesn't need to check feasibility with respect to the bus capacity because we assume it to be infinite.

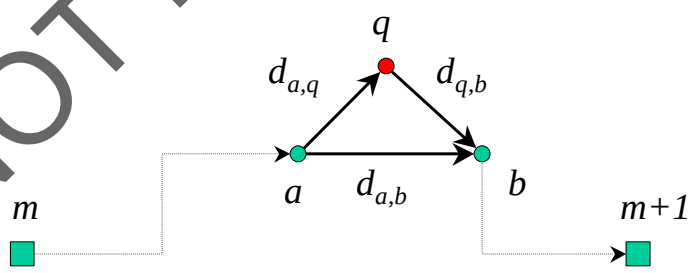

Figure 6 – Insertion feasibility of q

5.2 Cost function

When searching for the best insertion among the feasible ones, the algorithm computes a COST for each of them and selects the one with the minimum value. Let's assume that the insertion of a stop *q* between *a* and *b* is feasible and we need to compute its COST. The system's entities affected by an insertion are:

- The customer requesting the insertion, in terms of how long the ride time is.
- The passengers already onboard and waiting to be dropped off, in terms of how much longer they have to stay onboard.
- 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 expected ride time changes.
- The vehicle, in terms of how much extra miles it has to drive.

Thus, the algorithm computes the following quantities:

- ΔPT: the sum over all passengers of the extra ride time, including the ride time of the customer requesting the insertion.
- ΔPW: the sum over all passengers of the extra waiting time at the already inserted NP stops.

Finally, the cost function is defined as:

$$\text{COST} = w_1 * \Delta PT + w_2 * \Delta PW + w_3 * \Delta t_{a,q,b} \quad (19)$$

where w_1, w_2 and w_3 are the weights, which can be modified as needed to emphasize one factor over the others. $\Delta t_{a,q,b}$ corresponds to the consumption of the slack time (the resource needed by the system to serve more customers). During heavy demand periods, we should assign a higher value to this scarce resource by increasing w_3 with respect to w_1 and w_2. 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_1 and w_2 over w_3.

5.3 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 bus (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 the second bucket corresponding to the following occurrence of P, $s_{k'+1}(P)$). The 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 ($s_{k'}(D)$

with $k' = \min_k s_{k'}(D), s.t.\ t_{s_k(D)} \geq t_{now}$). In general, ($x_b$, y_b) doesn't 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 7. 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.

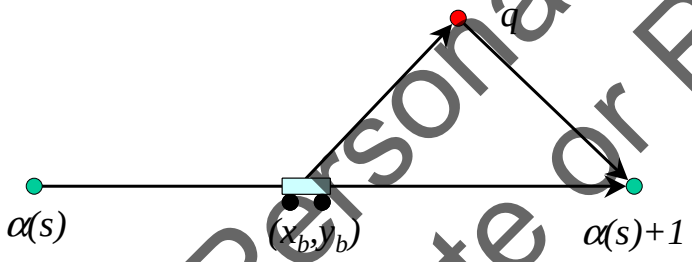

Figure 7 – Insertion from current bus position

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 rides 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 ride *r*. This is the first

occurrence in the schedule of one of the terminal checkpoints $s = 1$ or $s = C$, namely $s_{k'}(1 \setminus C) = \min_k s_k(1 \setminus C), s.t. t_{s_k(1\setminus C)} \geq t_{now}$. 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, …, *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* doesn't prevent NP insertion to be feasible in *i*), ND insertion needs to be checked only in bucket *i+1*. The procedure will continue till at least one NPND feasible insertion is found.

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. It may occur, for example with a very high demand rate, or when a customer request arrives towards the end of the service. On the other hand, the customers are assumed to never reject the insertion proposed by the algorithm and there is no negotiation between the MAST system and the customers.

5.4 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:

$$st_{m,m+1} = st_{m,m+1} - \Delta t_{a,q,b} \qquad (20)$$

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

$$t_s = t_s + \Delta t_{a,q,b} \qquad \forall s \text{ s.t. } o(s) \in [o(b), o(m+1)) \qquad (21)$$

$$t'_s = t'_s + \Delta t_{a,q,b} \qquad \forall s \text{ s.t. } o(s) \in [o(b), o(m+1)] \qquad (22)$$

Since the departure times t_s of checkpoints ($s \leq$ 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 (22).
- 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 (21) and (22). 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 (21) 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 from q, et_q, as follows:

$$et_q = t_a + d_{a,q}/bs + bt_q \tag{23}$$

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

$$t_q = t_a + d_{a,q}/bs + bt_q = et_q \tag{24}$$

It can easily be 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:

$$lt_q = et_q + st_{m,m+1} \tag{25}$$

We prove that lt_q is an upper bound for t_q by the following contradiction argument. Let's use the superscript β (with $\beta = 0, \ldots, f$) to indicate the β^{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 (21) that:

$$t_q^{(f)} - t_q^{(0)} = (t_q^{(f)} - t_q^{(f-1)}) + \ldots + (t_q^{(\beta)} - t_q^{(\beta-1)}) + \ldots + (t_q^{(1)} - t_q^{(0)}) =$$
$$= \Delta t_f + \ldots + \Delta t_\beta + \ldots + \Delta t_1 = \sum_{k=1}^{f} \Delta t_k \tag{26}$$

and from equations (24) and (25), $lt_q - t_q^{(0)} = lt_q - et_q = st_{m,m+1}$, 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 (25) 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 knowing that their actual times t_q and t'_q will be bounded by these values:

$$et_q \le t_q \le lt_q \quad (27)$$

$$et'_q \le t'_q \le lt'_q \quad (28)$$

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 $et'_D \le t'_D \le lt'_D$. Clearly NP and ND requests will also have $et_{NP} \le t_{NP} \le lt_{NP}$ and $et'_{ND} \le t'_{ND} \le lt'_{ND}$.

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 a used as an efficient scheduling tool for real MAST systems. We test its performance on a simulation model of the actual MAST service represented by MTA Line 646 in Los Angeles. In order to perform this task, we first need to define the MAST system's performance measures.

6.1 Performance measures

We define the following performance parameters for a MAST system:

- PT: average ride time per passenger
- PW: average extra waiting time (t_{NP} - et_{NP}) over NP requests only
- M: total miles driven by the bus

These three indicators are directly related to the corresponding terms of the COST function in Equation (19): ΔPT, ΔPW, $\Delta t_{a,q,b}$. Thus, we can similarly define the overall performance Z of a MAST system as:

$$Z = w_1 * PT * NC_T + w_2 * PW * NC_{NP} + w_3 * M/bs \quad (29)$$

where NC_T and NC_{NP} stand respectively for the total number of customers and the total number of NP customer requests (NPD and NPND types) served by the system. Z is in time units.

In addition, let's define:

- PI: average time interval between request/show up and earliest pick-up time (et_P or et_{NP}) per passenger
- PST: percentage of the total initial slack time ($= \sum_{s=1}^{nC-1} st_{s,s+1}^{(0)}$) consumed

Given a total demand rate θ (customers/hour), we define the *saturation level* as the maximum demand that a system configuration can satisfy without becoming unstable. This level can be estimated by looking at the PI values. Given that the demand is uniform over time, for systems well below their saturation level, the PI values should be around half the headway of the system. A slightly larger value of PI, 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 PI 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 is given by the PST; values around 90% indicate that the demand rate is more or less at saturation level. 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.

6.2 Algorithm performance

As earlier noted, a MAST service already exists in San Pedro in Los Angeles County, Line 646. San Pedro is one of Los Angeles County's busiest commercial hubs, consisting of several warehouses, factories and offices. Bus lines offer regular fixed-route service in the area during the daytime. 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, MTA Line 646 offers a MAST service during nighttime, transporting passengers between one of the business areas in San Pedro and a nearby bus terminal.

The MAST system represented by Line 646 consists of a single vehicle covering a service area with L = 10 miles and W = 1 mile, with two terminal checkpoints and one intermediate checkpoint located in the middle. The duration of each ride is 30 minutes and the headway is 1 hour. The service operates for 4.5 hours (9 rides) each night. Given that bs = 25 miles/hour, the system has very little slack time ($st^{(0)}_{s,s+1}$ = 2.5 minutes, for s = 1, …, TC-1; therefore about 6 minutes per ride), allowing very few insertions of non-checkpoint requests, 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 all the decisions concerning accepting/rejecting customer requests and routing the vehicle since the system needs to deal with only 2-3 insertion requests per ride.

MTA is interested in testing the MAST concept for higher demand levels. However, at the current slack level, the system will not be able to accommodate more demand. Therefore, in order to evaluate the performance of the insertion algorithm for the higher demand cases we perform the simulation experiments assuming a larger slack time. 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, …, TC-1)	5 miles
t_{s+1} - t_s (s = 1, …, TC-1)	25 min (t_1 = 0)
bs	25 miles/hour
bt_s (s = 1, …, TS)	18 sec
w_1 / w_2 / w_3	0.25 / 0.5 / 0.25

From equation (1) we compute the values of the initial slack times $st_{s,s+1}^{(0)}$ = 12.7 minutes (s = 1, …, 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 waiting time at stops (w_2) with more discomfort than the ride time on the bus (w_1) and that slack time consumption (w_3) and passengers' ride time (w_1) are equally weighted.

Given a total demand rate θ (customers/hour) constant over time, 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, the total number of rides R = 60.

We first perform a set of runs setting the control parameters BACK = L and $\pi_{s,s+1}^{(0)} = 1$ (for all s = 1,…,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 PI and PST values for different values of the demand θ. The results are shown in Table 4.

Table 4 – Saturation level for configurations A

Configuration	A1	**A2**	A3
θ (customers/hour)	15	**20**	25
BACK (miles)	L	**L**	L
$\pi^{(0)}_{s,s+1}$ s = 1, …, TC-1	1	**1**	1
PI (min)	56.52	**61.67**	236.74
PST (%)	81.3%	**91.3%**	98.9%
saturation level?	below	**yes**	above
PW (min)	1.07	**1.23**	1.75
PT (min)	23.86	**25.86**	30.39
M (miles)	1012.7	**1051.4**	1083.8

The findings show that the saturation level is around θ = 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 PI value is higher than half the headway (50 minutes), but it doesn't 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 θ = 20 would lead to system instability as shown by the results from A3, where the PI value is very high and keeps increasing with the simulation run time and the PST is close to 100%.

Therefore, by allowing more slack time in the schedule ($st^{(0)}_{s,s+1}$ = 12.7 minutes instead of 2.5, for s = 1, …, TC-1) and setting BACK = L and $\pi^{(0)}_{s,s+1}$ = 1 (configurations A), MTA Line 646 would be able to serve a demand θ with up to 20 customers/hour assuming the customer type distribution of Table 3.

Now, keeping the demand at the saturation level (configuration A2), we want to observe the effect of modifying the usable slack time $st^{u}_{s,s+1}$. For this purpose, maintaining BACK = L, we vary the values of $\pi^{(0)}_{s,s+1}$ (for all s = 1, …, 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 (29). 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 (13), $\pi_{s,s+1}^{(0)\min}$ is approximately equal to 0.22.

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

Configuration	B1 = A2	B2	B3	B4	**B5**	B6
θ (customers/hour)	20	20	20	20	**20**	20
BACK (miles)	L	L	L	L	**L**	L
$\pi_{s,s+1}^{(0)}$ s = 1, …, TC-1	1	0.75	0.5	0.4	**0.3**	$\pi_{s,s+1}^{(0)\min}$ =0.22
PI (min)	61.67	55.87	54.59	51.56	**52.26**	51.60
PST (%)	91.3%	87.4%	82.3%	79.2%	**76.6%**	72.0%
saturation level?	yes	below	below	below	**below**	below
PW (min)	1.23	1.15	1.25	1.32	**1.41**	1.37
PT (min)	25.86	24.68	24.13	23.09	**22.60**	22.76
M (miles)	1051.4	1021.7	989.0	968.2	**951.5**	921.7
Z	7149	6987	6853	6624	**6533**	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 PW, 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'd 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	**C6**	C7	C8
θ (customers/hour)	20	20	20	20	20	**20**	20	20
BACK (miles)	L	1.5	0.8	0.5	0.3	**0.2**	0.1	0
$\pi_{s,s+1}^{(0)}$ s = 1, …, TC-1	0.3	0.3	0.3	0.3	0.3	**0.3**	0.3	0.3
PI (min)	52.26	52.26	52.35	51.70	52.19	**52.23**	51.28	51.84
PST (%)	76.6%	76.6%	75.8%	74.2%	72.8%	**72.4%**	71.2%	70.9%
saturation level?	below	below	below	below	below	**below**	below	below
PW (min)	1.41	1.41	1.39	1.38	1.37	**1.37**	1.42	1.43
PT (min)	22.60	22.60	22.62	22.46	22.34	**22.28**	22.36	22.94
M (miles)	951.5	951.5	946.4	936.1	927.2	**924.2**	916.8	914.5
Z	6533	6533	6528	6478	6435	**6419**	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 PW and PT. 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 bus 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 PW, 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 θ.

Table 7 – New saturation level - configurations D

Configuration	D1 = C6	**D2**	D3
θ (customers/hour)	20	**25**	30
BACK (miles)	0.2	**0.2**	0.2
$\pi^{(0)}_{s,s+1}$ s = 1, …, TC-1	0.3	**0.3**	0.3
PI (min)	52.23	**55.98**	77.58
PST (%)	72.4%	**86.8%**	95.9%
saturation level?	below	**yes**	above
PW (min)	1.37	**1.72**	1.92
PT (min)	22.28	**23.93**	29.00
M (miles)	924.2	**983.4**	1020.6

As done for configurations A, we can estimate the saturation level for configurations D by looking at the stability of the PI 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.

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.

Future research on MAST systems could focus on finding lower bounds on the cost function through exact mathematical approaches, 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 bus MAST system to handle day-time 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 Line 646.

Appendix

Notation

r	[1, …, R]	rides' index
c	[1, …, C]	checkpoints' index
s	[1, …, TC, …, TS]	stops' index
$s_k(c)$	[1, …, TC]	stops' index of the k^{th} occurrence of c
$\alpha(s)$	[1, …, TS]	schedule's sequence index
L, W	length, width of service area	
R, C	total # of rides, total # of checkpoints	
TS, TC	total # of stops, total # of stops at checkpoints	
(x_b, y_b)	current position of the bus	
t_{now}	current time	
bt_s	boardings/disembarkment time at s	
bs	bus speed	
t_s, t_s'	departure/arrival times	
et_s, et'_s	earliest departure, arrival time at s	
lt_s, lt'_s	latest departure, arrival time at s	
$\hat{d}_r$	direction of r	
$d_{s,s'}$, $\hat{d}_{s,s'}$	scalar and vector distance between s and s'	
$bd_{s,s'}$	Backtracking distance between s and s'	
BACK	Backtracking parameter	
$st_{s,s+1}$	slack time between checkpoints s and $s+1$	
$st^u_{s,s+1}$	slack time usable between checkpoints s and $s+1$	
COST	cost function	
w_i	cost weights	
ΔPT	total extra ride time	
ΔPW	total extra waiting time	
$\Delta t_{s,s^*,s'}$	extra time needed to insert s^* between s and s'	
PI	average time interval between call/show-up and et_P/et_{NP}	
PST	% of total initial slack time used	
PT	average ride time	
PW	average extra waiting time over NP requests	
M	total miles driven	
Z	overall performance	
θ	total demand rate in service area	
λ	non-checkpoints' demand rate in service area	
$\Lambda_{s,s+1}$	expected demand in sector between checkpoints s and $s+1$	
$\pi_{s,s+1}$	slack time parameter	
β	variables' update index	

References

Bodin, L. and Sexton, T. (1986) "The multi-vehicle subscriber dial-a-ride problem", *TIMS Studies in the Management Sciences*, 22, 73-86.

Borndörfer, R. et al. (1999) "Telebus Berlin: vehicle scheduling in a dial-a-ride system", Lecture Notes in Economics and Mathematical Systems 471: Computer-Aided Transit Scheduling. Springer-Verlag, Berlin, 391-422.

Campbell, A. M., and Savelsbergh, M. (2003) "Efficient insertion heuristics for vehicle routing and scheduling problems," to appear *Transportation Science*

Cordeau, J.F. and Laporte, G. (2003) "A Tabu Search Heuristic for the Static Multi-vehicle Dial-a-ride Problem", *Transportation Research*, 37(B), 579-594.

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.

Dial, R.B. (1995) "Autonomous Dial-a-Ride Transit: Introductory Overview", *Transportation Research C*, 3C, 261-275.

Diana, M. and Dessouky, M. (2003) "A New Regret Insertion Heuristic for Solving Large-scale Dial-a-ride problems with Time Windows," to appear *Transportation Research*.

Dumas, Y., Desrosiers, J. and Soumis, F. (1991) "The pickup and delivery problem with time windows", *European Journal of Operational Research*, 54, 7-22.

Feuerstein, E. and Stougie, L. (2001) "On-Line Single-Server Dial-a-Ride Problems", *Theoretical Computer Science*, 268, 91-105.

Fu, L. (2002) "A simulation model for evaluating advanced dial-a-ride paratransit systems", *Transportation Research A*, 36A, 291-307.

Hart, S.M. (1996) "The Modeling and Solution of a Class of Dial-a-Ride Problems Using Simulated Annealing", *Control and Cybernetics*, 25, 131-157.

Healy, P. and Moll, R. (1995) "A new extension of local search applied to the dial-a-ride problem", *European Journal of Operational Research*, 83, 83-104.

Horn, M.E.T. (2002a) "Multi-modal and demand-responsive passenger transport systems: a modeling framework with embedded control systems", *Transportation Research A*, 36A, 167-188.

Horn, M.E.T. (2002b) "Fleet scheduling and dispatching for demand-responsive passenger services", *Transportation Research C*, 10C, 35-63.

Ioachim, I. et al. (1995) "A request clustering algorithm for door-to-door handicapped transportation", *Transportation Science*, 29, 63-78.

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.

Landrieu, A., Mati, Y. and Binder, Z. (2001) “A Tabu Search Heuristic for the Single Vehicle Pickup and Delivery Problem with Time Windows”, *Journal of Intelligent Manufacturing*, 12, 497-508.

Lao, H.C. and Liang, Z. (2002) “Pickup and Delivery with Time Windows: Algorithms and Test Case Generation”, *International Journal on Artificial Intelligence Tools (Architectures, Languages, Algorithms)*, 11(3), 455-472.

Li, H. and Lim, A. (2001) “A Metaheuristic for the Pickup and Delivery Problem with Time Windows”, Proceedings 13th IEEE ICTAI 2001, Los Alamitos, CA, 160-167.

Lu, Q. and Dessouky, M. (2003) “An exact algorithm for the multiple vehicle pickup and delivery problem”, to appear *Transportation Science*.

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.

Min, H. (1989) “The multiple vehicle routing problem with simultaneous delivery and pick-up points”, *Transportation Research A*, 23A, 377-386.

Nanry, W.P. and Barnes, J.W. (2000) “Solving the pickup and delivery problem with time windows using reactive tabu search”, *Transportation Research B*, 34B, 107-121.

Psaraftis, H.N. (1980) “A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem”, *Transportation Science*, 14, 130-154.

Psaraftis, H.N. (1983a) “An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows”, *Transportation Science*, 17, 351-357.

Psaraftis, H.N. (1983b) “k-interchange procedures for local search in a precedence-constrained routing problem”, *European Journal of Operational Research*, 13, 391-402.

Psaraftis, H.N. (1983c) “Analysis of an $O(N^2)$ heuristic for the single vehicle many-to-many Euclidean dial-a-ride problem”, *Transportation Research B*, 17B, 133-145.

Psaraftis, H.N. (1986) “Scheduling large-scale advance-request dial-a-ride systems”, *American Journal of Mathematical and Management Sciences*, 6, 327-367.

Ruland, K.S. and Rodin, E.Y. (1997) “The pickup and delivery problem: faces and branch-and-cut algorithm”, *Computers and Mathematics with Applications*, 33, 1-13.

Savelsbergh, M.W.P. and Sol, M. (1995) “The general pickup and delivery problem”, *Transportation Science*, 29, 17-29.

Savelsbergh, M.W.P. and Sol, M. (1998) “Drive: dynamic routing of independent vehicles”, *Operations Research*, 46, 474-490.

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.

Stein, D.M. (1977) “Scheduling Dial-a-Ride Transportation Systems: An Asymptotic Approach”, Harvard University, Division of Applied Science, Technical Report No. 670.

Stein, D.M. (1978a) “Scheduling dial-a-ride transportation problems”, *Transportation Science*, 12, 232-249.

Stein, D.M. (1978b) “An asymptotic probabilistic analysis of a routing problem”, *Mathematics of Operations Research*, 3, 89-101.

Teodorovich, D. and Radivojevic, G. (2000) “A fuzzy logic approach to dynamic dial-a-ride problem”, *Fuzzy Sets and Systems*, 116, 23-33.

Toth, P., and Vigo, D. (1997) “Heuristic Algorithm for the Handicapped Persons Transportation Problem”, *Transportation Science*, 31, 60-71.

Uchimura, K., Takahashi, H. and Saitoh, T. (2002) “Demand Responsive Service in Hierarchical Public Transportation System”, IEEE Transactions on Vehicular Technology, 51, 760-766.

Van Der Bruggen, L.J J., Lenstra, J.K. and Schuur, P.C. (1993) “Variable-depth search for the single-vehicle pickup and delivery problem with time windows”, *Transportation Science*, 27, 298-311.

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.