Abstract

CHAN, SOOK-YEE EDNA. Metaheuristics for solving the Dial-a-Ride problem. (Under the direction of John W. Baugh Jr.)

Many transit agencies face the problem of generating routes and schedules to meet customer requests consisting of either pickup or dropoff requests using an available fleet of vehicles. The Dial-a-Ride Problem (DARP) is a mathematical model that closely approximates the problem faced by these agencies. The problem is a generalization of the well-known *Pickup and Delivery Vehicle Routing Problem* or *Vehicle Routing Problem with Time Window*. However, due to the high level of service required by this type of transportation service, additional operational constraints must be considered. While the DARP can be solved exactly by various techniques, exact approaches for the solution to real-world problems (typically consisting of hundreds of requests) are not practical. The time required is often excessive as the problem is $\mathcal{NP}$-hard. In this thesis, we develop heuristics that find high quality solutions in a reasonable amount of computer time for the many-to-many, advanced reservation, multi-vehicle, single-depot, static DARP. The objectives considered include the minimization of total travel time and excess ride time, and the problem is subjected to maximum ride time, route duration, vehicle capacity,

and wait time constraints. The cluster-first route-second approach is adopted. Clustering is performed using either Tabu Search (TS) or Scatter Search (SS) while routing is performed via insertion. The class of insertion heuristics has been extensively applied to the DARP. Earlier algorithms focused on feasible insertions but recently, heuristics that allow infeasible insertions to be considered during searches have been introduced. In this research, two insertion heuristics are considered: IRAU, which assigns requests only when they are feasible, and IRDU, which assigns all requests even if they result in infeasibilities. Comparison studies show that the benefit of using a particular algorithm depends on the statistical properties of the data sets used. Overall, the algorithms generated better solutions than a previously published real-world (322-request) problem and found the optimal solutions for constructed (32-request and 80-request) problems with known optimal solutions.

METAHEURISTICS FOR SOLVING THE DIAL-A-RIDE PROBLEM

by

SOOK-YEE EDNA CHAN

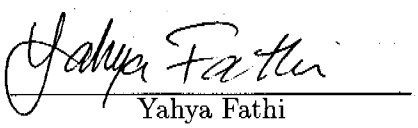
A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

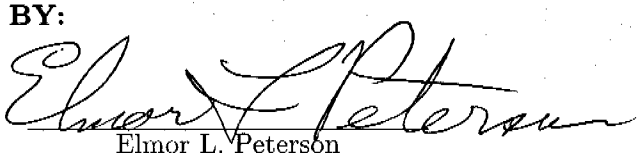
OPERATIONS RESEARCH

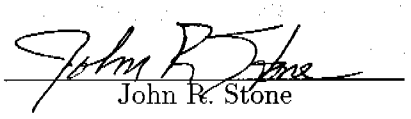
Raleigh

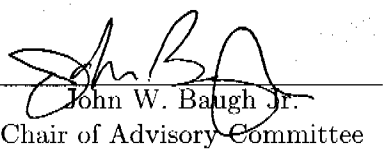
2004

APPROVED BY:


Yahya Fathi


Elmor L. Peterson


John R. Stone


John W. Baugh Jr.
Chair of Advisory Committee

Biography

Edna Chan was born in the Republic of Singapore. She graduated from Geylang Methodist Secondary School and started her tertiary education in Ngee Ann Polytechnic, graduating with a Diploma in Electronic Engineering. After some industrial experience, she continued her education in the University of Nebraska-Lincoln. In 1996 she graduated with a Bachelor of Science in Mathematics with a minor in Computer Science.

She then continued her studies in Mathematics and graduated in 1999 with a Master of Science in Mathematics and a minor in Industrial Engineering. In the fall of 2000 she began her doctoral studies at North Carolina State University. She earned a Master of Operations Research degree in 2003.

Acknowledgments

I would like to thank my advisor, Dr. John Baugh, for his invaluable guidance, patience, and encouragement throughout the course of this research. I will always remember his advice on the importance of having forests and not just leaves in future writing. I would also like to express my gratitude to Dr. Yahya Fathi, Dr. John Stone, and Dr. Elmor Peterson for giving their time and talent to serve on my committee.

I also wish to thank the Department of Mathematics and the Department of Civil Engineering at the North Carolina State University for providing financial support throughout my studies.

To my family – Thank you for your love, words of encouragement, and prayers when I needed them most. Finally, thank you Adriel for being with me every step (through the trials and victories) of the way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A Dial-a-Ride is a form of demand-responsive public transit that runs with a fleet of vehicles operating on flexible routes. When customers call a Dial-a-Ride service requesting to be transported from specified locations to other locations, the agency schedules and dispatches vehicles to service them. This form of service can be described as a shared-ride public transportation system where routes and schedules are planned with the possibility of multiple passengers (with different requests) traveling together during a typical day of operation. The problems involving the generation of such routes and schedules to meet customer demand are referred to as *Dial-a-Ride Problems* (DARPs). Practical applications of DARPs can be found in the form of airbuses, taxis, ambulances, and even certain courier services where the time necessary for service is relatively small compared to the driving time. However, the more popularly known application of DARPs is primarily associated with the service of the elderly and the disabled.

There are about 50 million Americans with disabilities today[1]. For these citizens, participation in society is limited by their mobility. Limited access to transportation entails limited participation in the many aspects of society, including work, commerce, and recreation. In recognition of this fact, the "Americans with Disabilities Act" (ADA) of 1990, a civil rights bill, was passed with the objective of eliminating the physical and attitudinal barriers that have kept persons with disabilities from participating fully in American society. It requires all transit agencies with fixed-route bus systems to also provide supplemental paratransit service or a Dial-a-Ride service. In addition to this societal need, there is also an emerging trend toward the development of ambulatory health care services as noted by Cordeau et al. [7], on top of an aging Western population. As a result, new Dial-a-Ride services are being set up or in cases where such systems are already in place, they are being overhauled in response to these needs. However, this kind of transportation service is very costly and labor intensive.

[1]Based on the 2000 census obtained from the Census Bureau at `http://www.census.gov`

DARPs are of interest to professionals including researchers and transit managers. Although commercial packages exist, our experience with transit agencies leads us to believe that existing commercial packages often produce inefficient schedules. In fact, we were informed that schedulers can easily find better alternate solutions than those generated by these packages[2]. Hence, there is a genuine need to be able to develop effective methods to solve DARPs.

The DARP involves the design of complete routes and schedules for a fleet of vehicles that service customers who specify pickup and dropoff requests for particular origins and destinations. The aim is to generate a set of minimum cost routes capable of accommodating as many requests as possible under a set of constraints. From the modeling point of view, the DARP generalizes a number of vehicle routing problems such as the *Pickup and Delivery Vehicle Routing Problem (PDVRP)* and the *Vehicle Routing Problem with Time Windows (VRPTW)*. For overviews of these problems, the reader is referred to survey papers by Desrosiers et al. [11] and Desaulniers et al. [9]. Toth et al. [48] call the problem the Handicapped persons Transportation Problem (HTP).

Due to the multiple competing objectives and constraints considered in the problem, DARPs are generally hard to solve. In fact, the multi-vehicle DARP has been shown to be $\mathcal{NP}$-hard [15]. In the last decade, metaheuristics have been applied to solve many combinatorial problems successfully. This makes it a candidate approach in solving the DARP. Furthermore, even though it is desirable to use exact methods to solve DARPs, time limitations and the large number of requests in real-world Dial-a-Ride systems render current exact algorithms ineffective. Hence, we have chosen metaheuristics as our approach to solving the DARP.

1.1 Dynamics of DARPs

To better understand the underlying dynamics of DARPs, we need to examine the concerns of its three major stakeholders: customers, drivers, and the transit agency.

A customer would prefer taxi-like service, which provides direct connections with no unnecessary detours from origin to destination. Basically, we can say that a customer wants short ride times and prompt service. For example, a customer will not want to travel for an hour when a direct route takes just ten minutes. Similarly, a customer will not appreciate being told to expect the arrival of his ride anytime between noon and two in the afternoon if instead a request can be made for his ride to arrive between noon and ten minutes after noon. It is obvious that every customer would prefer reliable and comfortable rides at affordable rates.

From a driver's point of view, certain routes may be preferred over others. This preference

[2]From personal communication with Linda Little, Director of Trans-AID at the Winston Salem Transit Authority, NC, in June 2003

may be influenced by the number of customers to be serviced and the traffic conditions encountered, such as congestion, number of traffic stops, and speed limits. Equity of workload may also be desired since an imbalance could result in the perception of unfairness. In addition, a driver may prefer having one long break in a day instead of several short breaks (or vice-versa). Every driver has an individual utility. Thus, it is difficult to generate schedules to please everyone.

With the available resources, such as vehicles and drivers, a transit agency aims to meet all customer demands at a low operating cost. However, an agency is faced with competing issues. On the one hand, it would like to provide prompt and reliable service to customers at affordable rates. On the other hand, it has to contend with operating costs, scarce resources, and occasionally driver unions. The uniqueness of the Dial-a-Ride system may sometimes provide an agency with opportunities to receive external funding. In order to ensure continued support, the agency will have to demonstrate that funds are utilized well and, at the same time, customers are satisfied. Accountability to the community and funding sources, in particular, may be the driving force that determines the policies adopted by an agency.

To account for all these various concerns, the DARP should be modeled with competing objectives: customer-attributed and agency-attributed. Furthermore, the model is usually a reflection of policies adopted by an agency. For example, Alfa [1] models a small-scale demand-responsive operation in Winnipeg. In his model, he minimizes the number of vehicles used as well as the cumulative deviation of actual arrival times from request times. Baugh et al. [2] model a mid-sized demand-responsive operation in Winston Salem. In their model, they minimize the number of vehicles used, the cumulative travel time, and the cumulative time window violation. A time window denotes the time interval during which a customer can expect the arrival of his ride. Depending on policies considered, the modeling scenario may differ.

1.2 General DARPs

It is not always clear what is meant by "The Dial-a-Ride Problem" since the meaning and statement of the problem are not consistent in the literature. For example, in Cordeau et al. [6] and Fu [14], a DARP is assumed to include time windows, or an interval of time where a customer can expect the arrival of his ride. To emphasize the inclusion of time windows in their problem, Madsen et al. [33] and Psarafatis [36, 37] state that they solve the Dial-a-Ride problem with time-windows (DARPTW). On the other hand, Healy et al. [25] refer to a DARP with no notion of time windows. As most studies of DARPs stem from analyzing real-world Dial-a-Ride transit services [1, 2, 33, 48] and since resource availability and policy issues considered by transit agencies are often, if not always, different, there are variations in the model objectives and constraints, further complicating the classification of DARPs. These

in turn result in the development of algorithms that focus on different solution types, which makes direct comparison of algorithms difficult or even impossible.

Generally, DARPs can be described in terms of the *nature of the Dial-a-Ride system* and the *policy issue considerations*. Reflecting the variation in real-world issues, the nature of a Dial-a-Ride system can be categorized as follows:

- the pattern of origins to destinations (one-to-many, many-to-many, many-to-one)
- the type of reservations (advanced or real-time)
- the number of depots (single or multiple)
- the type of requests (desired pickup request, desired dropoff request or both)
- the assumption of travel time (static or dynamic)

Policy issues considered in the formulation of a DARP may include:

- System requirements:
  - the classification of time windows (at pickup, at dropoff, at both pickup and dropoff or depending on request type)
  - the maximum allowable travel time (fixed or varying)
  - the scheduling horizon (fixed or varying)
  - the allowable waiting time (no waiting, at pickup stop, at dropoff stop or at both stops)
- Agency's objectives:
  - minimizing customer ride time
  - minimizing customer wait time
  - minimizing the deviation from a customer's request time
  - minimizing vehicle travel time
  - minimizing number of vehicles used
  - maximizing total number of trips per vehicle

It should be noted that the sole objective of an agency is to minimize system cost which is not easily quantified. The objectives listed above are surrogates for cost.

1.3 The Problem

Although different model formulations and algorithms have been developed to solve DARPs, some of them are more practical than others. In our research, the model formulated is based on what we believe to be realistic assumptions that reflect the policies of real-world Dial-a-Ride services. This section provides a detailed description of the DARP to be solved in this research.

Many transit agencies that provide Dial-a-Ride services have an advanced reservation system for customers to place their request the day before. Our research focuses on the generation of routes and schedules for these requests. With reference to the classifications of the nature of the Dial-a-Ride system described in Section 1.2, the problem that we are interested in is the many-to-many, advanced reservation, multi-vehicle, single-depot, static DARP. The following paragraphs describe the system requirements in detail.

The DARP model considered in our research can be described as follows. A fleet containing m vehicles serves trip requests. For each vehicle k, two types of seating accommodations may be specified – the number of ordinary seats, denoted by CO_k, and the number of wheelchair places, denoted by CW_k. In addition, a maximal route duration H cannot be exceeded and a uniform average speed may be specified for all vehicles.

A customer can call in to make two types of reservations – a desired pickup request (DPR) or a desired dropoff request (DDR). A DPR is requested when the time of departure from the origin is more important than the arrival time at the destination. In the case of a DDR, the arrival time at the destination is crucial. Hence when a DPR is requested, the customer must provide a desired time for pickup. An example of a DPR would be when a customer wants a ride to his home from a meeting he attended in a dangerous neighborhood. The prompt arrival of his ride is of utmost importance for his safety while the time he gets home is not important. When a DDR is requested, the customer has to specify a time for dropoff. An example of such a request is when a customer wants to travel from her home to a scheduled appointment at the dialysis center since she will not want to forfeit her appointment by arriving late. It is obvious that her arrival at the center is more important than the departure from her home.

Since it is possible for a customer to make more than one request, we will refer to a reservation as a request instead of identifying the reservation by a particular customer. The request includes attributes that are customer dependent. We assume that n requests, known in advance, are given to us. We index each request by i and its corresponding origin and destination stops as $+i$ and $-i$ respectively. Hence, the sequence

$$< 0, +1, +2, -2, -1, 0 >$$

represents the schedule where the vehicle leaves the depot (indexed 0), picks up *request 1*

followed by *request 2* then, drops off *request 2* then *request 1* before returning to the depot. In order to define the specific problem, the following information is uniquely determined for each request i:

- od-pair (or origin-destination pair): the pickup (origin) stop and the dropoff (destination) stop. We denote the origin stop as $+i$ and the destination stop as $-i$
- request type:
  - for DPR, the request time rt_i represents the time when the customer wants to leave the origin stop
  - for DDR, the request time rt_i represents the time when the customer wants to arrive at the destination stop
- duration of time window: the interval of time tw_i, negotiated between customer(s) and dispatcher, that the customer(s) should expect the arrival of the vehicle
- capacity requirement: seating accommodation requirement may be specified where co_i and cw_i represent the number of ordinary seats and wheelchair places required by request i. This formulation allows the modeling of customers who need wheelchair accommodation, as well as customers who need to travel with an accompanying person due to special needs
- loading time: the time $load_i$ required for the customer(s) to get on or get off the vehicle. In our model, we assume that the loading time is the same at the pickup and dropoff stops. That is, $load_{+i} = load_{-i} = load_i$
- maximum ride time: m_i is the time the customer arrives at his destination minus the time the vehicle departs from his origin stop. In other words, this is the total time the customer is in the vehicle starting from his origin stop until he arrives at his destination stop. In the literature, this time is usually expressed as a proportion of the time needed to travel directly from the origin to the destination. We denote $m_i = a + b \times t(+i, -i)$ where pre-specified parameters $a, b \geq 0$ and $t(i, j)$ represents the time taken to travel from stop i to stop j

While it is possible that time windows can be negotiated between a customer and the scheduler, in practice, fixed time windows are normally adopted. In the following paragraphs we describe a procedure to define time windows with reference to the request time. This procedure allows the modeling of scenarios when no negotiated time windows exist.

In the case of a DDR, a customer desires to arrive at a destination at some specified request time. When the customer calls in to make such a request, the dispatcher will have to determine

a probable interval of time for the customer to reach the destination, or $[e_{-i}, l_{-i}]$ where e_{-i} and l_{-i} represent the earliest and latest service time at stop $-i$, respectively. This interval can be negotiated between the dispatcher and the customer or, in the case of our model, it can be created as follows:

$$\begin{aligned} e_{-i} &= rt_i + dOff \\ l_{-i} &= e_{-i} + tw_i \end{aligned}$$

where *dOff* is a real number such that when $dOff < 0$, the time window starts before the request time rt_i. Otherwise, it starts either at or after the request time. The inclusion of this variable allows flexibility in the way time windows are specified.

In the case of a DPR, a customer specifies the time rt_i that he wants to be picked-up from the origin stop. When the customer calls in to make such a request, the dispatcher will have to provide the customer with a time window for the arrival of the vehicle at the origin. As with a DDR, this interval can be predetermined or it can be created as follows:

$$\begin{aligned} e_{+i} &= rt_i + pOff \\ l_{+i} &= e_{+i} + tw_i \end{aligned}$$

where *pOff* is a real number where the effect of *pOff* on the time window is similar to that of *dOff*.

We now discuss the constraints placed on wait time at stops. No passenger will endure waiting in a vehicle for too long. Therefore, it is reasonable to set a maximal allowable wait time at each stop. If we let w_j be the wait time at stop j, then we require that $w_j \leq W$, where W is the maximum wait time allowable, whenever passengers are on board. However, if no passengers are on board the driver can wait at a stop without restriction, especially in the case of an early arrival at an origin stop since the customer may not be ready. On the contrary, if the vehicle arrives at a destination stop too early while passengers are aboard, no waiting is allowed. Basically, when the vehicle arrives at a destination stop, the customer is allowed to alight immediately. However, if the vehicle arrives at an origin stop earlier than the specified time window of the customer to be picked up, it can wait (for not more than W) for the customer to board the vehicle. This is a departure from models considered in the literature because previous models either allow waiting or they do not allow waiting, regardless of the type of stops and the number of passengers in the vehicle. Practically, allowing the vehicle to wait at a destination stop is equivalent to keeping a customer in the vehicle even though the vehicle has arrived at the destination stop. Consequently, other customers in the vehicle are forced to wait with the unhappy customer.

Now that we have described the nature and system requirements of our DARP, we will summarize the **constraints** that must be satisfied by our vehicle routes:

(i) **round trip**: every route starts and ends at the depot

(ii) **coupling**: for every request i, the od-pair $(+i, -i)$ must belong to the same route

(iii) **precedence**: the origin stop $+i$ must be visited before the destination stop $-i$

(iv) **time windows**: time windows pertaining to both DPR and DDR must not be violated

(v) **vehicle capacity**: the maximum concurrent passengers requiring ordinary seats and wheelchair places in vehicle k at any instant must not exceed co_k (number of ordinary seats) and cw_k (wheelchair places) respectively

(vi) **maximum ride time**: the total time that request i is in a vehicle must not exceed m_i

(vii) **route duration**: the total duration of each route must not exceed H

(viii) **no waiting at destination stops**: no waiting at destination stops is allowed

(ix) **maximum waiting time**: the total waiting time at origin stops must not exceed W when the vehicle is carrying passengers

In this research, feasibility is met only when all constraints, that is, constraints (i) – (ix), are satisfied. The ***cost*** function must reflect the objective of the transit system. Because of the stakeholders involved (in our case, customer and transit agency), DARPs are generally modeled with competing objectives. In our model, we aim to minimize:

1. the total travel time TT (that is, the total time required to travel from stop to stop for all vehicles)

2. the total customer excess ride time ERT where the excess ride time of a customer is computed as the difference between the arrival time at his destination stop and the departure time at his origin stop minus the direct travel time from his origin to the destination stop

Mathematically, the cost function can be represented as:

$$cost = \eta TT + \psi ERT \tag{1.1}$$

where $\eta,\ \psi > 0$ indicate the relative weights of the different objectives.

While agencies are interested in minimizing system costs, or cumulative travel time in our case (since this affects fuel expenditure, and the lifespan of vehicles), they are also concerned

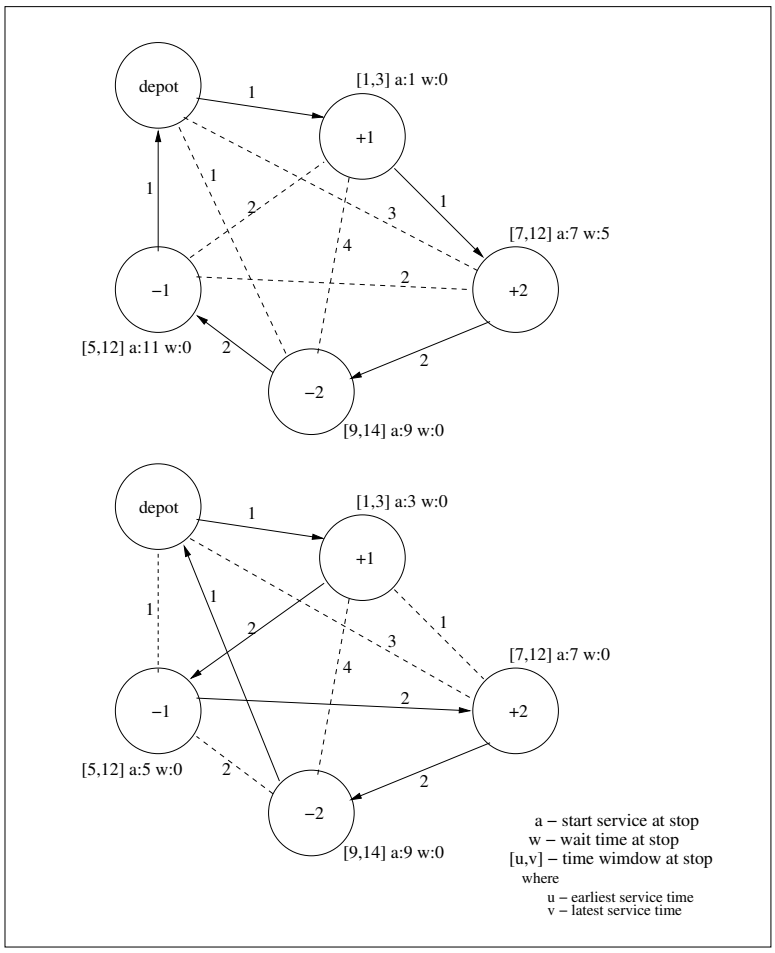


Figure 1.1: The tour $< 0, +1, +2, -1, -2, 0 >$ (top) results in a shorter total travel time (TT=7) than the tour $< 0, +1, -1, +2, -2, 0 >$ (bottom) which has a total travel time (TT=8).

about the satisfaction of their customers, or minimizing cumulative excess ride time in our model. This fundamental tradeoff between cost and customer satisfaction can be illustrated by the two tours shown in Figure 1.1. In the top network, the tour is represented as $T_1 =< 0, +1, +2, -2, -1, 0 >$. In this case, the vehicle leaves the depot (or stop 0) and arrives at stop $+1$ after one unit of time without waiting and then proceeds to stop $+2$. Because the vehicle arrives at this stop before its earliest service time (at seven units of time) it waits for five units of time. Next, the vehicle travels to stop -2 and arrives at nine units of time. Finally at 11 units of time, the vehicle arrives at stop -1 before returning to the depot. It can be seen that TT = 7, however this tour requires customer 1 to wait (for 5 units of time) before customer 2 is serviced. An alternate tour $T_2 =< 0, +1, -1, +2, -2, 0 >$ represented by the bottom network in the same figure has TT = 8, but no customers are required to wait.

Table 1.1: Example Client Table

ID	Name	Address	Wheelchair
C216	"Franklin, Amy"	L002	no
C217	"McCain, Maggie"	L013	yes
C218	"Moore, Thelma"	L025	no
C219	"Ellison, Roy"	L123	no
C219	"Ellison, Beth"	L122	yes

In addition, it can be noted that the cumulative excess ride time for tour T_1 is six, i.e., ERT=6, which is longer than that of tour T_2, i.e., ERT=0. Despite the longer total travel time, tour T_2 gives customers the feeling that they are serviced better. Hence, because of our dual objective, we require both weights to be strictly greater than zero.

1.4 DARP Data Structure

In our research, a DARP instance is represented by five tables: client, location, vehicle, request, and model. Each table consists of fields that provide information about the specific set. Two types of request tables are recognized: the Type A table does not require the specification of time window and assumes a fixed time window setting for all requests while the type B table allows a pre-specified time window. The fields for each data set are shown below:

- **Client** - name, address, wheelchair requirement
- **Location** - address, geo-coordinates(latitude, longitude) or xy-coordinates
- **Vehicle** - capacity, wheelchair availability
- **Request** - client, origin, destination, type, request time, loading time, load and (for type B), an additional specification for the time window (TW)
- **Model** - maximum ride time (fixed:M or vary:a, b), time window (fixed:tw, $pOff$, $dOff$ or vary), speed, route:H

For illustrations of client, location, vehicle, and request tables, refer to Tables 1.1 – 1.5 respectively. An example model table is shown below:

$$MRT \;\; FIXED \;\; 60 \;\; TW \;\; FIXED \;\; 10 \;\; 0 \;\; 0 \;\; SPEED \;\; 30 \;\; ROUTE \;\; 480$$

The above example can be described as follows: the maximum ride time for all requests is fixed at 60 minutes. A fixed time window of 10 minutes is specified for all requests and each time

Table 1.2: Example Location Table

ID	Address	x-coordinate	y-coordinate
L403	"120 Miller Street"	0.5386713767692366	0.0030497183215678625
L247	"618 N Spring Street"	-1.353336878166211	-1.6172067678836877
L102	"1060 N Trade Street"	-1.9749960225005156	0.824697371443468
L390	"452 N Cherry Street"	-1.4909464783810047	0.9943576252514399
L390	"122 Riverside Street"	1.2439464538500237	0.8043573552514369

Table 1.3: Example Type A Request Table

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load
R314	C242	L131	L221	2003/01/07	9:30	P	2	1
R201	C126	L110	L749	2003/01/07	8:20	D	1	1
R232	C658	L367	L237	2003/01/07	10:30	D	1	1
R879	C940	L231	L980	2003/01/07	10:45	P	2	1
R129	C900	L221	L880	2003/01/07	10:15	D	2	1

Table 1.4: Example Type B Request Table

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R314	C242	L131	L221	2003/01/07	9:30	P	2	1	9:20	9:40
R201	C126	L110	L749	2003/01/07	8:20	D	1	1	9:00	9:30
R232	C658	L367	L237	2003/01/07	10:30	D	1	1	10:45	10:50
R442	C561	L327	L137	2003/01/07	10:30	P	1	1	10:00	11:00
R879	C940	L231	L980	2003/01/07	10:45	D	2	1	10:40	11:05

Table 1.5: Example Vehicle Table

ID	Capacity	Wheelchairs
V01	14	0
V07	12	1
V10	12	1
V13	16	0
V03	23	1

window is determined by setting $pOff = dOff = 0$ as described in Section 1.3. In addition, the average speed for each vehicle is set at 30 miles/hour and finally, the maximum route duration allowable is 480 minutes.

In our research, a solution to a DARP instance is represented as a complete tour of each vehicle, where the tour is composed of a sequence of stops visited by the vehicle and a specific time that the vehicle should arrive at each stop. This solution is represented by a stop table that has the following fields:

- **Stop** - vehicle, request, type, location, arrival time, departure time, next stop

For an example of a stop table refer to Table 1.6, which shows the tour taken by vehicle 2 (V02), the sequence of the tour, and the time vehicle 2 arrives and departs from each stop.

Table 1.6: Example Stop Table

ID	Vehicle	Request	type	Location	Arrive	Depart	Next Stop
S001	V02	depot	start	L001	0:00	7:55	S002
S002	V02	R097	P	L110	8:05	8:05	S003
S003	V02	R097	D	L749	8:20	8:20	S004
S004	V02	R121	P	L100	8:38	8:38	S005
S005	V02	R121	D	L134	8:50	8:50	S006

1.5 Scope

This research deals with the many-to-many, advanced reservation, multi-vehicle, single-depot, static DARP. The objectives are to minimize total excess ride time and total travel time. These are subject to round trip, precedence, coupling, maximum ride time, vehicle capacity, time window, wait time and route duration constraints.

One of the goals of this research is to develop heuristics to solve the DARP. Insertion heuristics for the multi-objective DARPs have been studied by Jaw et al. [28], Toth et al. [48] and Roy et al. [40, 41]. In their algorithms only feasible solutions were considered. Infeasible solutions were considered in tabu search approaches developed by Cordeau et al. [6] and Nanry et al. [32] for single-objective DARPs. A significant contribution of this research is the exploration of both feasible and infeasible insertions for a multi-objective DARP. The exploration of both types of insertions facilitates comparison of the benefits gained from considering one type of insertion over the other with regard to the nature of the DARP. A Tabu Search and a Scatter Search metaheuristic are developed.

To the best of our knowledge, no benchmark problems exist for the DARP model considered in our research. Therefore, another contribution is made to the literature by the generation of

benchmark problems. In these problems, a specific policy was considered for the purpose of evaluating the heuristics that were developed.

Finally, we produce a Graphical User Interface for the visual representation of a DARP instance and its solution. Besides providing a way to display the tables of the DARP instance, this tool can also be used to provide an aerial view of stops and the specific routes taken by each vehicle in the solution.

Chapter 2

Literature Review

DARPs have been a focal point of research since the inception of the demand-responsive paratransit service concept. The first few DARP studies conducted by Wilson et al. [52, 50, 51] focused on developing real-time algorithms for several pioneering paratransit systems. Other models to facilitate economic planning and efficient operation of these systems were proposed by Daganzo [8] and Stein [47]. The initial idea was to provide real-time service. However, due to high operating cost, most Dial-a-Ride systems were turned into reservation-based operations after the late 1970s. Research on DARPs also shifted to efficient heuristics for solving the static version of the problem where most requests were known in advance. The following sections describe algorithms developed for both the single-vehicle and multi-vehicle static DARP.

2.1 Single-Vehicle DARPs

Exact methods using dynamic programming for the single-vehicle DARP are presented by Psaraftis [36, 37]. The objective function addressed is the minimization of a weighted combination of total time taken to service all customers and customer dissatisfaction. As is often the case with dynamic programming, the algorithm can be applied only to small problems since the computational complexity of the procedure is $O(n^2 3^n)$ where n denotes the number of requests. The largest instance solved using this approach is nine customers. Although single-vehicle DARPs do not exist in practice, algorithms developed to solve such problems can be used as subroutines in multi-vehicle DARPs provided the number of customers in each route remains small.

Sexton et al. [45, 46] introduce an approach based on Benders' decomposition. The objective function addressed by these researchers is one that minimizes customer inconvenience, which is expressed as a linear combination of excess ride time (that is, the time difference between the actual ride time and the direct ride time of a user) and the deviation from the user's desired

drop-off time from the actual drop-off time. DARP instances reported contain request sizes between 7 and 20.

Van Der Bruggen et al. [49] propose a local search method based on variable-depth search. The method consists of two phases. In the first phase, a feasible route is constructed and in the second phase, the route is improved. In both phases, a variable-depth search based on seven arc-exchange moves is used. The objective is to minimize total route duration. Tests were conducted on DARP instances with request sizes ranging from 5 to 50.

2.2 Multi-vehicle DARPs

Among the heuristics that have been successfully developed for the multi-vehicle DARP is a class of insertion heuristics in which requests are *inserted* into existing routes. This class of heuristics has been investigated quite extensively in the literature.

One of the first insertion heuristics developed for DARP was proposed by Jaw et al. [28] called ADARTW (Advanced Dial-A-Ride with Time Window). In ADARTW, requests are first sorted based on their earliest service time at their origin stops. Each request is then inserted sequentially into schedule blocks in existing vehicle routes where a schedule block is represented by a sequence of stops starting and ending at the depot. A vehicle can contain more than one schedule block. The model considered by these authors imposes one time window at the origin stop for DPR and one time window at the destination stop for DDR. A maximum ride time restriction is imposed for each user and is expressed as a linear function with respect to the direct ride time of the user. In addition, no waiting at a stop is allowed whenever passengers are aboard the vehicle. Their objective is minimization of total customer disutility (in particular, the total deviation from request time and ride time). DARP instances tested involved a randomly generated instance containing 250 requests and a real-world instance with 2617 requests, the largest instance reported to date in the literature. Alfa [1] applies ADARTW to a small-scale demand-responsive operation in Winnipeg after adding a module to account for vehicle capacity constraints. Kikuchi et al. [29] develop a parallel insertion heuristic, a variation of ADARTW. In their algorithm, the notion of schedule blocks is removed.

Toth et al. [48] propose a two-phase algorithm that first inserts requests via a parallel insertion heuristic (TV) followed by inter- and intra-route exchanges. Further improvements can be made by applying a Tabu Thresholding algorithm (TT) to the solution obtained. Their model imposes a limit, proportional to the direct ride time, on the maximum ride time of each user. One time window is defined for each type of request. In the case of DPR, this time window is specified at the origin stop and in the case of DDR, the time window is defined at the destination stop. In addition, the fleet of vehicles considered consists of minibuses and

special cars where each vehicle may contain various seating capacity constraints. On occasions, when necessary, taxis can also be mobilized. However, because they are less suitable for the transportation of disabled persons, a penalty is imposed on their use. The objective is to minimize the total cost of service. DARP instances of 276 and 312 requests were tested and the results showed significant improvements over previous hand-made solutions. Besides insertion heuristics, another commonly used technique developed to solve DARPs is the cluster-first route-second approach. In this method, requests are first grouped into clusters and economical routes are then constructed for each cluster. Bodin et al. [3] propose an algorithm using this approach. Clusters are first formed by grouping users based on their space-time information. A route is then constructed for each cluster via the single-vehicle algorithm proposed in Sexton et al. [45, 46] and by swapping requests between clusters. The algorithm was tested on data extracted from a Baltimore database with 85 requests each.

Dumas et al. [12, 13] propose a mini-cluster first, route second approach. The approach is similar to cluster-first route-second except that at the first stage, rather than defining clusters corresponding to all given requests, mini-clusters are formed by grouping nearby customers who can be transported by the same vehicle over a route segment. At the second stage, feasible routes are then constructed by stringing these route segments via a column generation algorithm. Numerical results are presented for six test problems obtained from the Canadian cities of Sherbrooke, Toronto and Montreal. DARP instances of up to 200 requests and 85 mini-clusters are easily solved. However these authors observed that with larger instances, spatial and temporal decomposition techniques will have to be employed. Ioachim et al. [27] show that it is beneficial to use mathematical optimization techniques to globally define the mini-clusters.

While significant progress has been made on multi-vehicle DARPs, very few studies investigate the use of metaheuristics. The past decade saw the rise of several metaheuristics that were adopted in other areas to solve many difficult combinatorial problems successfully. Among those developed for DARPs, Baugh et al. [2] propose a cluster-first route-second approach. In the model formulation, two time windows are specified for every request; one for the origin stop and one for the destination stop. Clustering is performed via a simulated annealing implementation that incorporates the notion of a tabu list, and routing is performed via a space-time nearest neighbor algorithm. The multi-objective function addressed by these authors is one that minimizes total distance traveled, customer disutility, and fleet size. Customer disutility is quantified by the amount of time window violation. A randomly generated DARP instance of 30 requests and a real-world instance from North Carolina with around 300 requests were examined.

Nanry et al. [32] develop a reactive Tabu Search implementation. In their formulation, they consider time window violation, overload violation and total distance traveled. Three

neighborhood structures are considered. The first removes a request from one vehicle and inserts it into another. The second swaps two requests from different vehicles and the last structure re-positions the sequence in which a request is visited. The objective is to minimize total time traveled. Randomly generated test instances of size 25, 50, and 100 were examined.

More recently, Cordeau et al. [6] develop a Tabu Search with vertex reinsertion heuristic. The model considered by these authors specifies one time window at the origin stop for DPRs and one time window at the destination stop for DDRs. The maximum ride time for each user can either be expressed as the largest deviation from its direct ride time or can be fixed for all users. In addition, there is a route length constraint associated with each vehicle. The objective is to minimize total route length. The algorithm was tested on randomly generated instances with request sizes ranging from 24 to 144 and on six real-world data sets of size 200 and 295 provided by a Danish transporter.

Another area of research is the multi-vehicle dynamic DARP. Madsen et al. [33] develop an insertion heuristic called REBUS and solve a real-life problem that involves servicing the elderly and disabled in Copenhagen. Gendreau et al. [16] develop an adaptive memory-based neighborhood search heuristic where new solutions are explored through a neighborhood structure based on ejection chain. Another paper that is relevant to DARPs is the study by Fu [14]. In Fu's formulation, travel times are time-varying and stochastic. For more information on DARP models, algorithms, and applications, refer to the survey papers by Cordeau et al. [7] and Desaulniers et al. [9].

Chapter 3

Exact Algorithm - MILP

The DARP has been proven to be $\mathcal{NP}$-hard. While it may be ideal to develop an exact algorithm that can solve the DARP optimally, due to the large-sized DARP instances considered in real-world Dial-a-Ride applications, it is generally not practical in terms of computational time to develop exact methods to solve the multi-vehicle DARP. Baugh et al. [2] consider several approaches via integer programming to solve the multi-vehicle DARP using a commercial optimization package. These authors were successful in solving small-sized DARP instances of 10 requests. Larger DARP instances were solved using a customized branch-and-bound approach but the authors reported difficulties in identifying feasible integer solutions when the number of requests considered grew beyond 15.

Assuming an improvement in commercial optimization packages and since a different model formulation is considered in our research, we developed a mixed-integer linear programming (MILP) formulation for our DARP and solved it using a commercial package. Table 3.1 summarizes the notations used in our MILP and the mathematical MILP formulation is shown below.

We considered DARPs of various sizes and the largest DARP problem size that could be solved was about 15 customers in 3 vehicles. Since real-world DARP problem sizes are normally in the hundreds, it can be concluded that the current MILP formulation is inadequate in solving the multi-vehicle DARP.

Table 3.1: Notations for MILP

Parameters	
N	set of requests and $\|N\| = n$
$S = S^+ \cup S^-$	$S^+ = \{+1, +2, \ldots, +n\}$ is the set of origin stops
	$S^- = \{-1, -2, \ldots, -n\}$ is the set of destination stops
0	depot stop
$\hat{S}$	$S \cup 0$
V	set of vehicles and $\|V\| = m$
$t(i, j)$	travel time between stops i and j
e_i	earliest service time at stop i
l_i	latest service time at stop i
$load_i$	loading time for request i
cw_i	number of wheelchair places required by request i
co_i	number of ordinary seats required by request i
m_i	maximum ride time of request i
W	maximum wait time at an origin stop
CO_k	maximum number of ordinary seats in vehicle k
CW_k	maximum number of wheelchair places in vehicle k
H	maximum route duration for each vehicle
Additional variables for MILP	
y_i	arrival time at stop i
x_{ij}^k	1 if travel from stop i to stop j is on tour k
	0 otherwise
w_i	wait time at stop i
nw_i	number of occupied wheelchair places at stop i
no_i	number of occupied ordinary seats at stop i
T	a number larger than the latest drop-off time

Minimize:

$$z = \eta * \left(\sum_{i \in \hat{S}} \sum_{j \in \hat{S}} \sum_{k \in V} t(i,j) x_{i,j}^k \right) \quad + \quad \psi * \left(\sum_{i \in N} y_{-i} - y_{+i} - t(+i, -i) \right)$$

Subject to:

Tours leaving the depot

$$\sum_{j \in S^+} x_{0,j}^k = 1 \quad k \in V$$

Tours returning to the depot

$$\sum_{i \in S^-} x_{i,0}^k = 1 \quad k \in V$$

Every origin should be visited

$$\sum_{k \in V} \sum_{j \in S} x_{i,j}^k = 1 \quad i \in S^+$$

A stop is in only one tour

$$\sum_{j \in S} x_{j,i}^k - \sum_{j \in S} x_{i,j}^k = 0 \quad i \in \hat{S}, \;\; k \in V$$

Origin and destination of a customer are in the same trip

$$\sum_{j \in S} x_{+i,j}^k - \sum_{j \in \hat{S}} x_{(-i),j}^k = 0 \quad i \in N, \;\; k \in V$$

Arrival time at a stop must be after the departure time at the previous stop

$$y_j - w_j \geq y_i + load_i + t(i,j) \sum_{k \in V} x_{ij}^k - \left(1 - \sum_{k \in V} x_{ij}^k\right) T \quad i,j \in \hat{S}$$

$$y_j - w_j \leq y_i + load_i + t(i,j) \sum_{k \in V} x_{ij}^k + \left(1 - \sum_{k \in V} x_{ij}^k\right) T \quad i,j \in \hat{S}$$

Wait time constraint at an origin stop

$$z_j \leq 1 \quad j \in S^+$$

$$z_j \leq c_j \quad j \in S^+$$

$$w_j \leq W z_j + T(1 - z_j) \quad j \in S^+$$

$$w_j \geq 0 \quad j \in S^+$$

No wait time at any origin stop that is visited immediately after the depot

$$w_j \leq \left(1 - \sum_{k \in V} x_{0,j}^k\right) T \quad j \in S^+$$

Wait time constraint at a destination stop

$$w_j = 0 \quad j \in S^-$$

DPR time window constraint at a stop

$$e_i \leq y_i \leq l_i \quad i \in S^+$$

DDR time window constraint at a stop

$$e_i \leq y_i \leq l_i \quad i \in S^-$$

Maximum-ride time constraint

$$y_{-i} - y_{+i} - load_i \leq m_i \quad i \in N$$

Route duration constraint

$$y_i + load_i + t(i,0) - y_0 - H \leq M(1 - x_{i,0}^k) \quad i \in S^{-i} \;\; k \in V$$

Service time at a destination must occur after the service time at its origin for every od-pair

$$y_{-i} - y_{+i} - load_i \geq t(+i,-i) \quad i \in S^+$$

Capacity of the vehicle at the start of the trip

$$nw_0 = 0$$

$$no_0 = 0$$

Capacity of the vehicle after visiting an origin stop

$$nw_j \geq nw_i + cw_j \sum_{k \in V} x_{i,j}^k - \left(1 - \sum_{k \in V} x_{i,j}^k\right) T \quad i \in \hat{S} \;\; j \in S^+$$

$$nw_j \leq nw_i + cw_j \sum_{k \in V} x_{i,j}^k + \left(1 - \sum_{k \in V} x_{i,j}^k\right) T \quad i \in \hat{S} \;\; j \in S^+$$

$$no_j \geq no_i + co_j \sum_{k \in V} x_{i,j}^k - \left(1 - \sum_{k \in V} x_{i,j}^k\right) T \quad i \in \hat{S} \;\; j \in S^+$$

$$no_j \leq no_i + co_j \sum_{k \in V} x_{i,j}^k + \left(1 - \sum_{k \in V} x_{i,j}^k\right) T \quad i \in \hat{S} \;\; j \in S^+$$

Capacity of the vehicle after visiting a destination stop

$$nw_j \leq nw_i - cw_j \sum_{k \in V} x_{i,j}^k + \left(1 - \sum_{k \in V} x_{i,j}^k\right) T \quad i \in S \;\; j \in S^-$$

$$nw_j \geq nw_i - cw_j \sum_{k \in V} x_{i,j}^k - \left(1 - \sum_{k \in V} x_{i,j}^k\right) T \quad i \in S \;\; j \in S^-$$

$$no_j \leq no_i - cw_j \sum_{k \in V} x_{i,j}^k + \left(1 - \sum_{k \in V} x_{i,j}^k\right) T \quad i \in S \;\; j \in S^-$$

$$no_j \geq no_i - co_j \sum_{k \in V} x_{i,j}^k - \left(1 - \sum_{k \in V} x_{i,j}^k\right) T \quad i \in S \;\; j \in S^-$$

Number of occupied ordinary seats cannot exceed the maximum ordinary seats available

$$no_i \leq CO_k \quad i \in S \;\; k \in V$$

Number of occupied wheelchair places cannot exceed the maximum wheelchair places available

$$nw_i \leq CW_k \quad i \in S \;\; k \in V$$

In addition,

$$x_{i,j}^k = 0 \quad \text{whenever} \quad \begin{cases} i = -j, \;\; j \in N \;\; \text{and} \;\; \forall k \in V \\ i = 0, \;\; j \in S^- \text{and} \;\; \forall k \in V \\ j = 0, \;\; i \in S^+ \text{and} \;\; \forall k \in V \end{cases}$$

$$z_i \in 0, 1 \quad i \in N$$

Chapter 4

Solution Methodologies

In the previous chapter, we presented a mixed-integer linear programming formulation for solving the DARP. However, the formulation was found to be inadequate. In particular, the largest DARP problem size that we were able to solve in a reasonable amount of time was a 15-request DARP instance. While it may be ideal to solve the DARP optimally, any problem that takes too long to solve is impractical since schedules are generated on a daily basis in the DARP model that we consider. Instead, we focus on finding approaches that can solve the DARP in a reasonable amount of time. In this chapter, we discuss the approaches we take toward solving the DARP via metaheuristics.

Our approach uses a cluster-first route-second strategy. In a cluster-first route-second strategy, requests are first organized into clusters. Economical routes are then developed to schedule all requests in each cluster. Several researchers have adopted the cluster-first route-second strategy successfully to the Vehicle Routing Problem with Time Window (VRPTW). The DARP is a generalization of the VRPTW. For examples of applications using the approach, refer to Gillet et al. [17] and Chapleau et al. [4]. The modular structure of this strategy allows plug-ins of various possible clustering and routing techniques. Hence, it allows easy experimentation and exploration of different techniques. Also, this strategy allows "presetting" groups of customers who desire to *travel together*. By assigning these customers to the same vehicle (or same cluster) this will ensure that they are routed in the same tour. This is a normal occurrence in real-world Dial-a-Ride systems. In our approach, we consider two clustering techniques; Tabu Search and Scatter Search. Routing is performed using two insertion heuristics we developed.

In developing our routing techniques, we consider two approaches. One approach is to disregard solution feasibility during the search for a solution. However, there is no guarantee that the available fleet size is adequate to service all requests without violating the desired constraints. In other words, infeasibility is inevitable with the given resources, that is, fleet

size or driver availability. In which case, it is possible that the final (best) solution may be infeasible. One way to avoid infeasibility is to introduce alternate transportation, e.g., taxis, into the original available fleet. Another approach is to ensure feasibility throughout the search. If a request cannot be assigned feasibly, the request will be left unassigned. Depending on the policies of the implementing agency, unassigned requests can either be assigned to alternate transportation or be "turned away."

In our research, both approaches are explored. Because of the cluster-first route-second strategy adopted, we were able to explore both approaches by means of developing suitable routing techniques. The routing techniques developed are:

- insertion router - allow unassigned (IRAU)
- insertion router - disallow unassigned (IRDU)

Refer to Section 5 for the discussions of both techniques. The next few chapters are organized as follows: In Chapter 5, the routing techniques are presented. Following that, we present the Tabu Search approach in Chapter 6 and report results obtained. Next, in Chapter 7, we discuss the Scatter Search approach and present our results. We then compare the algorithms developed in both approaches over test problems and real-world DARP problems in Chapter 8.

Chapter 5

Routing Techniques

In the cluster-first route-second approach, requests are first grouped into clusters. Tours are then generated that route requests in each cluster. In the following sections, we describe two routing heuristics developed for the DARP described in Section 1.3. Both routing techniques route via insertion. The two routing techniques can be characterized as follows:

- Insertion Router - Allow Unassigned (or IRAU)
  - tour generated is *always* feasible, that is, all constraints discussed in Section 1.3 are met
  - requests that cannot be routed will be left unassigned since routing the request may result in an infeasible tour
- Insertion Router - Disallow Unassigned (or IRDU)
  - tour generated may not be feasible, that is, some of the constraints may be violated
  - no requests are left unassigned

The specification of a time window at each stop can facilitate insertion by quantifying the time interval in which a stop can be inserted. This time interval is used as a guide to determine a finite range in time in which the stop can be considered for insertion hence limiting the number of insertion positions in an existing schedule. Because only one time window is defined for each request, that is, $[e_{+i}, l_{+i}]$ for DPR and $[e_{-i}, l_{-i}]$ for DDR, a fictitious time window is created so that each request will have *two* time windows. The fictitious time window for each request type can be computed as follows:

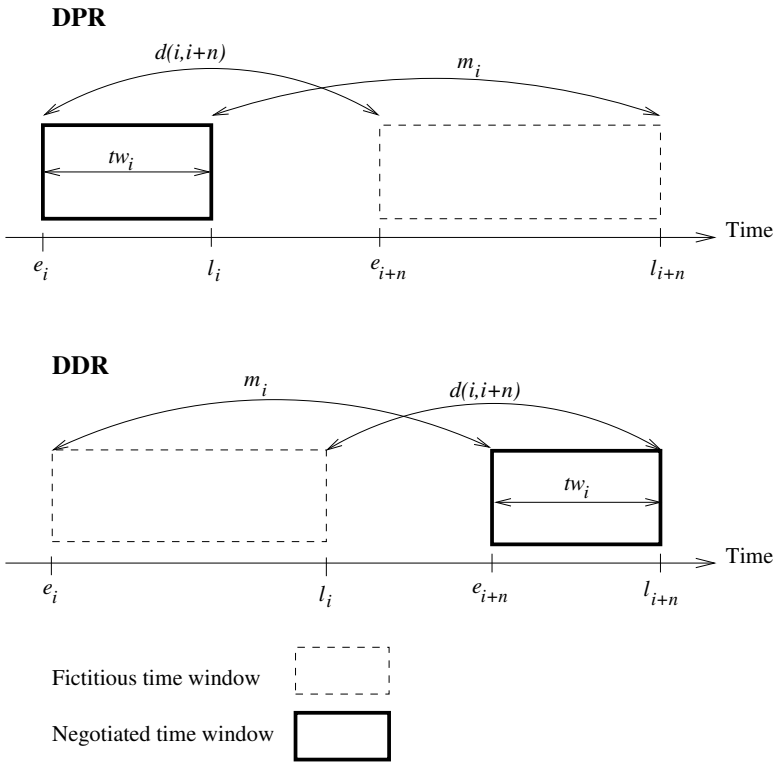


Figure 5.1: A schematic representation of DPR and DDR negotiated and fictitious time windows.

For DPR:

$$\begin{aligned} e_{-i} &= e_{+i} + load_i + t(+i,-i) \\ l_{-i} &= l_{+i} + load_i + m_i \end{aligned}$$

For DDR:

$$\begin{aligned} e_{+i} &= e_{-i} - m_i - load_i \\ l_{+i} &= l_{-i} - t(+i,-i) - load_i \end{aligned}$$

Refer to Figure 5.1 for a schematic representation of DPR and DDR time window parameters. Observe that with two time window specifications, regardless of request type, a set of necessary but not sufficient conditions for feasibility is

$$e_i \leq a_i \leq l_i$$

where a_i represents the scheduled service start time for stop i. Although the above equation ensures that the time window is satisfied, it is not sufficient since it is possible that the maximum ride time constraint is violated. Hence, an additional check must be conducted to ensure that the maximum ride time constraint is not violated.

To simplify notation, we adopt the shorthand summarized in Table 5.1 in the discussion of the routing techniques described in the remaining sections. The insertion or removal of a request from a tour is thus equivalent to the insertion or removal of two stops representing the origin and destination stops of the request.

Table 5.1: Notations for the routing techniques.

e_i	earliest service time at stop s_i
l_i	latest service time at stop s_i
$load_i$	loading time at stop s_i
W	maximum wait time allowed at each stop
H	maximum route duration of each vehicle
$t(i,j)$	the time taken to travel from stop s_i to stop s_j and $t(i,j)$ =(distance between stops s_i and s_j)/*speed*
D_i	departure time at stop s_i
A_i	arrival time at stop s_i where $A_i = D_{i-1} + t(i-1,i)$
B_i	planned arrival time at stop s_i where $B_i \geq A_i$
W_i	wait time at stop s_i

5.1 Insertion Router - Allow Unassigned

In this section, we propose a router that is capable of scheduling requests via insertion and enforces feasibility for each qualified insertion. We will refer to our routing technique as **Insertion Router - Allow Unassigned** (IRAU). The insertion of a request, or two stops; one for the origin and the other for the destination, into a tour is performed so as to minimize total cost while keeping the ordering of stops already in the tour unchanged. Such an insertion is only allowed if the insertion of the two stops does not lead to a violation of all constraints considered, as discussed in Section 1.3, for the new stops and all existing stops in the tour. If the insertion of any stop leads to a violation, the corresponding request will be left unassigned. In addition to insertion, since it is required by our clustering technique, this router is also able to remove a specific request while minimizing infeasibility since it is possible that the removal of the request may result in the violation of constraints. An error reflecting the infeasibility of removal of a specific request will be returned. This error is computed as follows:

$$\begin{aligned} error = {} & \textit{maximum wait time violation} + \textit{maximum ride time violation} \\ & + \textit{route duration violation} + \textit{no. of excess ordinary seats required} \\ & + \textit{no. of excess wheelchair places required} \end{aligned}$$

Hence, in the **evaluation** of an insertion or a removal of a request, the following information is returned:

- For insertion:
  - the cost, as indicated in Equation 1.1, and
  - an indicator if request is assigned or left unassigned.
- For removal:
  - the cost, as indicated in Equation 1.1, and
  - error associated with the removal. If request is removed feasibly, the error is zero.

In the following sections, we discuss how a request is inserted and also how it is removed from a tour when IRAU is selected as the candidate routing technique.

5.1.1 Inserting a request

The insertion of a request into an existing tour in the IRAU routing technique can be summarized as follows:

1. Find all possible ways in which the request can be inserted into an existing tour.
2. If the request cannot be inserted feasibly into the tour, add the request to the unassigned list of this tour. Else, insert its origin and destination stop into the positions in this tour that result in the least cost.
3. Return tour and unassigned list.

Before discussing the details involved in inserting a request, it is important to first understand how a schedule can be shifted forward and backward for all stops preceding and succeeding some stop i in order to accommodate the insertion of a new stop since the insertion of a request requires the insertion of an origin and a destination stop. Clearly, time windows impose a limit on the amount of shift that we can make to all stops already in the tour in order to insert a new stop at a particular position in the tour. Assume an ordered sequence of stops, say tour $T =< s_0, s_1, \ldots, s_i, \ldots, s_{q-1}, s_q >$, is given, where stops s_0 and s_q both represent the depot. Assuming $load_i = 0$, Savelsbergh [43] defines the forward time slack of a stop s_i as:

$$F_i = \min_{i \leq j \leq q} \left\{ l_j - \left(B_i + \sum_{i \leq p < j} t(p, p+1) \right) \right\} \tag{5.1}$$

The forward time slack indicates how long the service time of all stops after stop i (inclusive) can be delayed, without violating the time window requirement of these stops. This amount

of delay is restricted by the smallest time shift allowed for all stops following stop i(inclusive). Cordeau et al. [6] uses the fact that

$$B_j = B_i + \sum_{i \leq p < j} t(p, p+1) + \sum_{i < p \leq j} W_p$$

and substitutes the above equation into Equation 5.1 to get

$$F_i = \min_{i \leq j \leq q} \left\{ \sum_{i < p \leq j} W_p + (l_j - B_j) \right\} \tag{5.2}$$

Equation 5.2 emphasizes the fact that the allowable amount of delay at stop s_i is the cumulative waiting time from stop s_{i+1} to stop s_j, plus the difference between the latest service time and the earliest service time at each stop. Observe that this equation also generalizes to the case of non-zero loading time. This gives us a way to compute the allowable amount of shift that can be made for all stops succeeding the position in which the new stop will be inserted. We adopt the convention of setting time from the first stop to the last stop. Hence, when considering the insertion of stop α into position i of an existing tour T, we can define four parameters that will determine the amount of forward and backward allowable shifts. They are:

$$P_i^+ = \min_{j \leq i}(l_j - B_j), \tag{5.3}$$

$$P_i^- = \min_{j \leq i}(B_j - e_j), \tag{5.4}$$

$$S_i^+ = \min_{j \geq i} \left\{ \sum_{i < p \leq j} W_p + (l_j - B_j) \right\}, \quad \text{and} \tag{5.5}$$

$$S_i^- = \min_{j \geq i}(B_j - e_j). \tag{5.6}$$

P_i^+ (S_i^+) denotes the amount of time all stops preceding (succeeding) stop i (inclusive) can be delayed or shifted backwards in order to insert the new stop. Similarly, P_i^- (S_i^-)

represents the amount of time all stops preceding (succeeding) stop i (inclusive) can be advanced or shifted forward in order to insert the new stop. The new tour becomes $T' =< s_0, \ldots, s_{i-1}, \alpha, s_i, \ldots, s_{q+1} >$.

Suppose that to accommodate the insertion of a stop we need to delay the service time of all stops preceding or succeeding stop s_i (inclusive) by an amount of $SHIFT > 0$. There are three ways to shift stops in a schedule.

1. If all stops preceding stop s_i(inclusive) have to be pushed forward by $SHIFT > 0$, this can be done by a simple advancement in service time for these stops, that is, $B_j \leftarrow B_j - SHIFT$ for $j \leq i$. Observe that with this time shift, the wait time of each stop is not affected since these stops are shifted by the same amount of time. The subroutine that performs this shift is shown in Algorithm 5.1.1. **aMove** $(j, i, \boldsymbol{SHIFT})$ advances the service time of all stops from stop j (inclusive) to stop i (inclusive) by *SHIFT*.

   Algorithm 5.1.1 aMove $(j, i, \boldsymbol{SHIFT})$

   $B_j \leftarrow B_j - \text{SHIFT}$
   $D_j = B_j + \text{load}_j$
   Set $k = j + 1$
   Repeat until $k = i$
   $B_k \leftarrow B_k - \text{SHIFT}$
   $W_k = B_k - A_k$
   $D_k = B_k + \text{load}_k$
   $k \leftarrow k + 1$

2. If an absolute shift is desired for all stops following stop s_i, that is, we need to delay the service time of *all* stops from position i onwards; this shifting can be done by setting $B_j \leftarrow B_j + SHIFT$ for $j \geq i$. Algorithm 5.1.2 shows the subroutine that performs this shift. **bMove** $(i, j, \boldsymbol{SHIFT})$ delays the service time of all stops from stop i (inclusive) to stop j (inclusive) by *SHIFT*.

   Algorithm 5.1.2 bMove $(j, i, \boldsymbol{SHIFT})$

   $B_i \leftarrow B_i + \text{SHIFT}$
   $D_i = B_i + \text{load}_i$
   Set $k = i + 1$
   Repeat until $k = j$
   $B_k \leftarrow B_k + \text{SHIFT}$
   $W_k = B_k - A_k$
   $D_k = B_k + \text{load}_k$

$k \leftarrow k + 1$

3. If a delay of *SHIFT* is only required for stop s_i, it is clear that sequentially setting $B_i \leftarrow B_i + SHIFT$ and $B_j \leftarrow \max\{B_j, A_j\}$ for $j > i$ suffices. However, in the case of destination stops, we require $B_i \leftarrow B_i + SHIFT$ and $B_j \leftarrow \max\{e_j, A_j\}$ for $j > i$ since no wait time is allowed based on constraint (viii) discussed in Section 1.3. The subroutine that performs this operation on all stops following stop s_i is shown in Algorithm 5.1.3. **cMove** $(i, j, \boldsymbol{SHIFT})$ delays the service time of stop s_i by *SHIFT* and delays subsequent stops only if the delay in service time of stop s_i affects it.

Algorithm 5.1.3 cMove $(i, j, \boldsymbol{SHIFT})$

$B_i \leftarrow B_i + \text{SHIFT}$
$D_i = B_i + \text{load}_i$
Set $k = i + 1$
Repeat until $k = j$,
  If s_k *is an origin stop*
    $B_k \leftarrow \max\{A_k, B_k\}$
  else
    $B_k \leftarrow \max\{A_k, e_k\}$
  $W_k = B_k - A_k$
  $D_k = B_k + \text{load}_k$
  $k \leftarrow k + 1$

Inserting request i into an existing tour is equivalent to inserting two stops: stop $+i$ (origin stop) and stop $-i$ (destination stop). Stop $+i$ will be inserted first into the tour to be followed by stop $-i$ which will be inserted later in the tour after the position in which stop $+i$ was inserted. There are mainly two ways to view the insertion of these stops; (A) the destination stop $-i$ and origin stop $+i$ are at least one stop away from each other in the new tour, i.e, after visiting stop $+i$, other stops are visited first before the vehicle services the destination stop $-i$ or (B) the destination stop $-i$ follows immediately after the origin stop $+i$ in the new tour. The two cases are summarized below.

CASE A

When the origin and destination stops are inserted into the tour at least one stop away from each other (with the destination stop occurring at a later position in the tour), we can consider

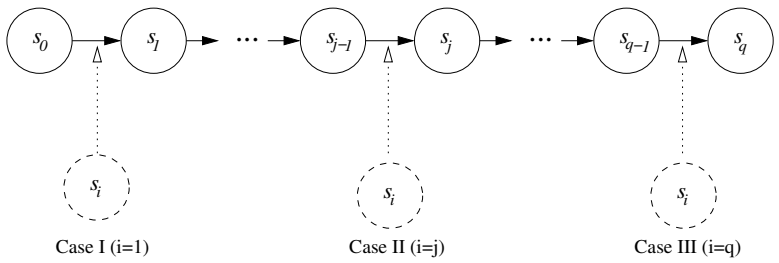


Figure 5.2: Three possible positions to insert stop s_i into an existing tour given by $< s_0, \ldots, s_j, \ldots, s_q >$.

the insertion of each stop separately. In developing our algorithm, we minimize the number of insertions considered so as to increase the computational efficiency of the algorithm. It is clear that if the insertion of an origin stop at some position is infeasible, then we will not need to consider the insertion of its destination stop. There are altogether three ways of inserting a stop into an existing tour. A stop can be inserted

Case AI - immediately after the (start) depot

Case AII - between other origin or destination stops

Case AIII - immediately before the (terminal) depot

Refer to Figure 5.2 for a schematic representation of the three possible positions to insert a stop.

CASE B

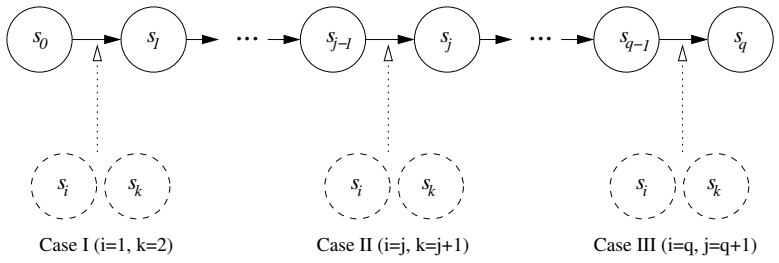


Figure 5.3: Three possible positions to insert stops s_i and s_k into an existing tour given by $< s_0, \ldots, s_j, \ldots, s_q >$.

In this case, we consider the scenario where the destination stop of a request is visited immediately after its origin stop in the new tour. Observe that if we consider the insertion of these two stops separately, as in case A, it is possible that we may reject the insertion of its

origin stop prematurely. Consider the case where we want to insert an origin stop, for example stop s_a, immediately before the origin stop, for example stop s_b, of some other request in tour T. Due to the time window requirement of stop s_a, the insertion of stop s_a may result in $W_b > W$ since the vehicle will be servicing a passenger when it arrives at stop s_b. However, if both its origin and destination are considered at the same time, the new tour may be feasible. As in case A, there are three possible ways to insert an origin stop followed immediately by a destination stop into an existing tour. They are:

Case BI - immediately after the (start) depot

Case BII - between other origin or destination stops

Case BIII - immediately before the (terminal) depot

Refer to Figure 5.3 for a schematic representation of the three possible positions to insert two stops together, origin stop followed immediately by its corresponding destination stop. The insertion of two stops can be performed by a two-step procedure using the mini-cases described in case A with the exception that the schedule will not be rejected except when the schedule is infeasible after the insertion of a destination stop. The following summarizes the two-step procedure used in all three cases:

Case BI

- $i = 1$, $k = i + 1$
  - Insert stop s_i: Case AI
  - Insert stop s_k: Case AII

Case BII

- $i = j$, $k = i + 1$
  - Insert stop s_i: Case AII
  - Insert stop s_k: Case AII

Case BIII

- $i = q$, $k = i + 1$
  - Insert stop s_i: Case AII
  - Insert stop s_k: Case AIII

Table 5.2: Notations for Figures 5.4, 5.5, and 5.6.

s_i	stop to be inserted Case AI) $s_0 \quad s_i \quad s_{i+1} \quad \ldots\ldots \quad s_{q+1}$ Case AII) $s_0 \quad \ldots\ldots \quad s_{i-1} \quad s_i \quad s_{i+1} \quad \ldots\ldots \quad s_{q+1}$ Case AIII) $s_0 \quad \ldots\ldots \quad s_{i-1} \quad s_i \quad s_{q+1}$
e_i	earliest service time at stop s_i
l_i	latest service time at stop s_i
$load_i$	loading time at stop s_i
W	maximum wait time allowed at each stop
H	maximum route duration of each vehicle
$t(i,j)$	the time taken to travel from stop s_i to stop s_j and $t(i,j)$ =(distance between stops s_i and s_j)/*speed*
D_i	departure time at stop s_i
A_i	arrival time at stop s_i where $A_i = D_{i-1} + t(i-1,i)$
B_i	planned arrival time at stop s_i where $B_i \geq A_i$
W_i	wait time at stop s_i
shift_j	the amount of time by which the schedule is to be shifted, for $j = 1, 2, \ldots$
viol	the amount of time window violation
dur	the required duration for a detour
P_i^+ (S_i^+)	the amount of time all stops preceding (succeeding) s_i (inclusive) can be delayed
P_i^- (S_i^-)	the amount of time all stops preceding (succeeding) s_i (inclusive) can be advanced

Figures 5.4, 5.5, and 5.6 show the procedures for Cases AI, AII, and AIII respectively in the form of flowcharts. The notations used in these flowcharts are summarized in Table 5.2.

The procedure involved in the insertion of a stop in all cases can be generalized as a sequence of the following steps:

1. Determine if a detour can be made to accommodate the insertion of this stop. This may require the shifting of stops preceding and/or succeeding the position of insertion. If no detour can be made, such an insertion is infeasible.

2. Check for violation of the time window of this newly inserted stop. If the time window is violated, shift stops appropriately to minimize this violation. Continue until either the time window constraint is met or no further shifting can be done without violating any constraints.

3. Evaluate the cost of the insertion of this stop. This evaluation includes the computation of the objective function and the error as described in one of the previous paragraphs.

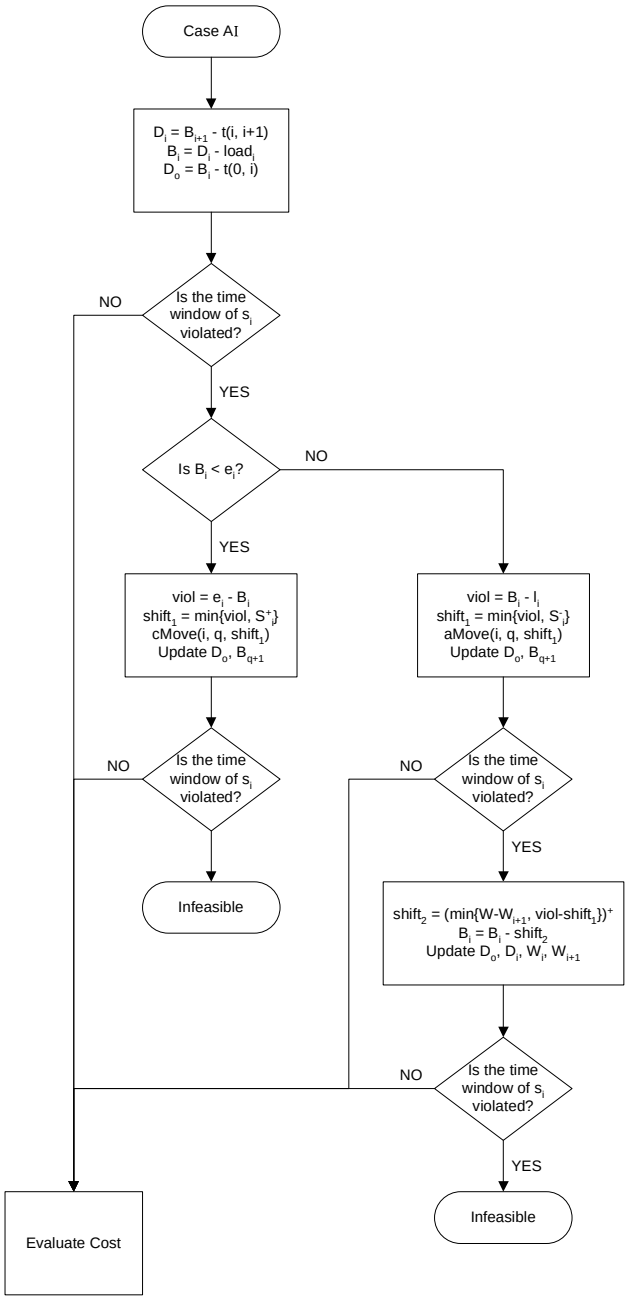


Figure 5.4: IRAU - Case AI

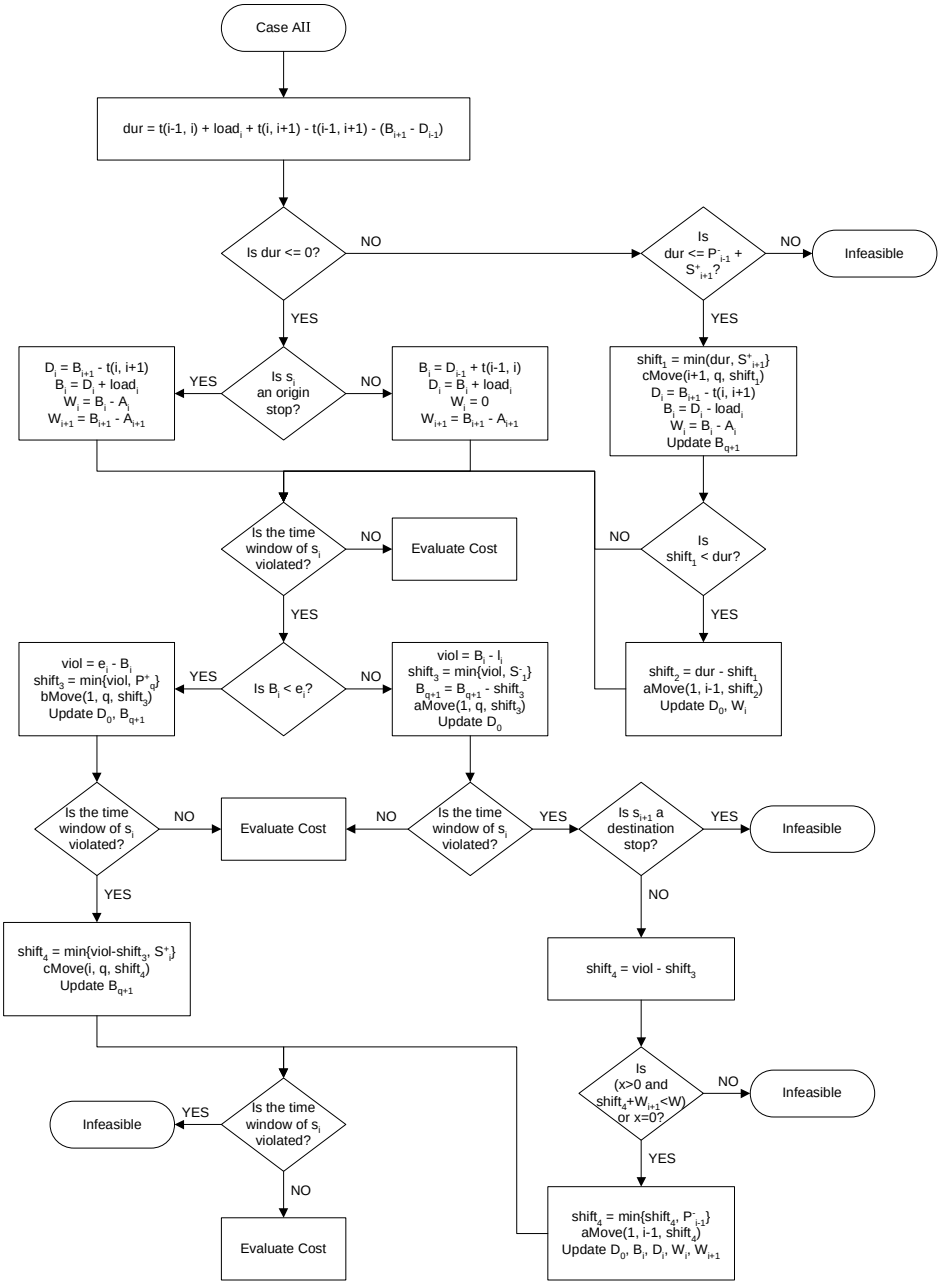


Figure 5.5: IRAU - Case AII

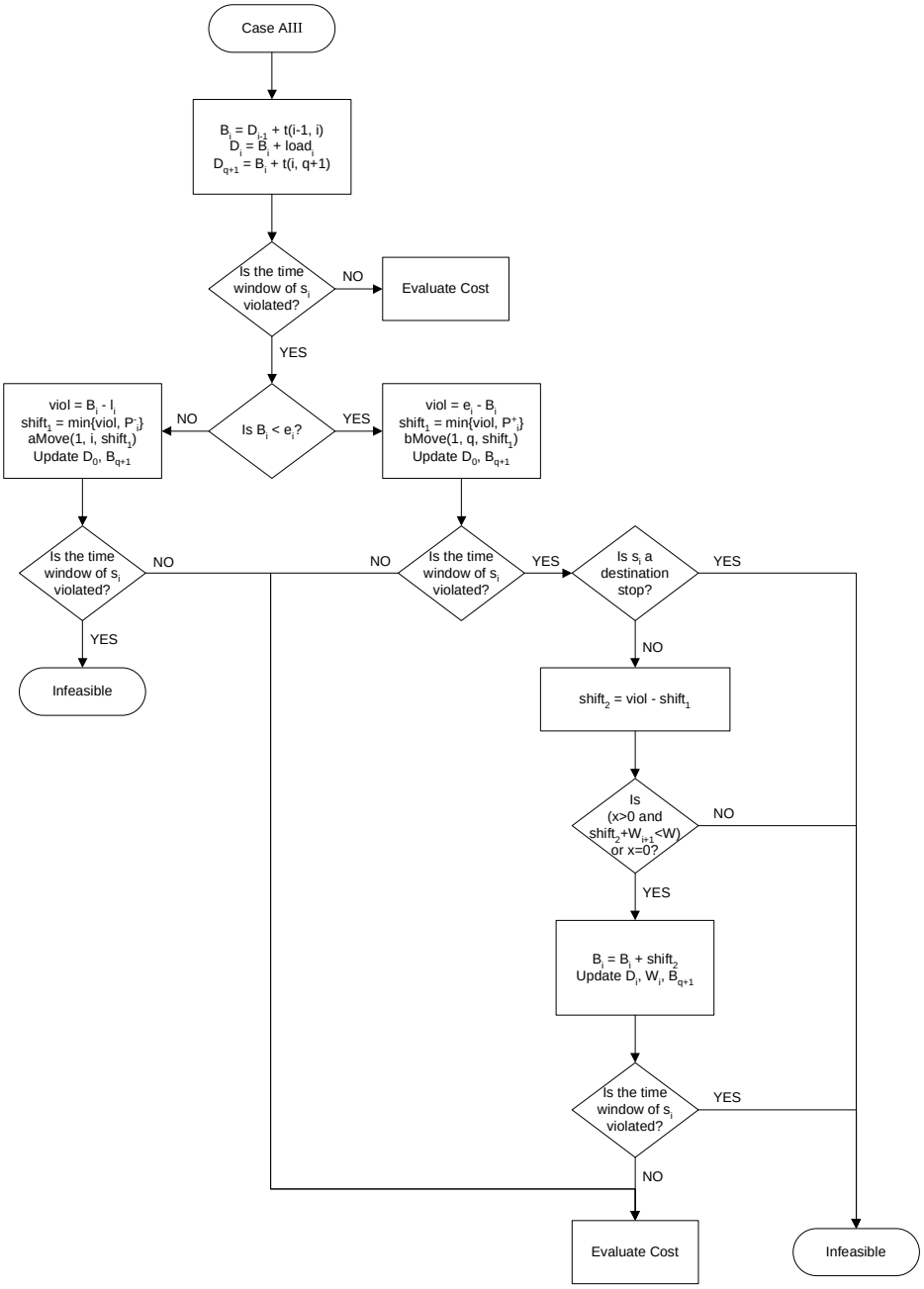


Figure 5.6: IRAU - Case AIII

5.1.2 Removing a request

Besides the insertion of a request as described in the previous section, the routing technique must also be able to remove a request from an existing schedule. The removal of a request is equivalent to the removal of its origin and destination stops. Hence, in the following discussion, we discuss the details involved in removing a stop.

Due to the feasibility requirement, care must be taken in the removal of a stop from an existing tour since the removal of a stop may result in an infeasible schedule. Just as in the insertion of a stop as described in the previous section, there are also three ways to remove a stop from a tour. They are:

Case RI - immediately after the (start) depot

Case RII - between other origin or destination stops

Case RIII - immediately before the (terminal) depot

Generally the removal of a stop creates a break in the tour resulting in some amount of unaccounted time. This unaccounted time can be reduced by shifting the stops preceding and/or succeeding the position of the stop to be removed. If this shifting cannot reduce the unaccounted time completely, the removal is infeasible and an error reflecting this infeasibility will be returned. An example of such a case is when the middle stop of a sequence of three pickup stops is to be removed; the removal of this stop may cause the wait time at the last stop to exceed the maximum allowable wait time. Figure 5.7 shows the procedures involved in the removal of a stop in all cases, that is, Cases RI, RII, and RIII, in the form of flowcharts. The notations used to describe these procedures are shown in Table 5.3.

5.2 Insertion Router - Disallow Unassigned

In this section, we discuss a different insertion heuristic. Unlike the insertion heuristic described in Section 5.1 where feasibility of the solution is maintained throughout the routing process and requests that cannot be routed feasibly are left unassigned, this new router allows infeasible solutions during routing and thus, no requests are left unassigned. Requests are assigned regardless of their violation of any constraints but the final insertion of a request is performed so as to minimize total violations. We refer to this router as **Insertion Router - Disallow Unassigned** (IRDU).

Before moving on with the discussion of this new router, we need to revisit the notion of the forward time slack representation in Equation 5.2. The forward time slack at stop s_i is written as:

Table 5.3: Notations for Figure 5.7.

s_i	stop to be removed
s_{ps}	stop immediately preceding s_i in the current tour
s_{ns}	stop immediately succeeding s_i in the current tour Case RI) s_{ps} s_i s_{ns} ..... s_q Case RII) s_0 ..... s_{ps} s_i s_{ns} ..... s_q Case RIII) s_0 ..... s_{ps} s_i s_{ns}
W	maximum wait time allowed at each stop
$t(i,j)$	the time taken to travel from stop s_i to stop s_j and $t(i,j)$ =(distance between stops s_i and s_j)/*speed*
D_i	departure time at stop s_i
A_i	arrival time at stop s_i where $A_i = D_{i-1} + t(i-1,i)$
B_i	planned arrival time at stop s_i where $B_i \geq A_i$
W_i	wait time at stop s_i
shift_j	the amount of time by which the schedule is to be shifted
gap	the break in the schedule caused by the removal of s_i
P_i^+ (S_i^+)	the amount of time all stops preceding (succeeding) s_i (inclusive) can be delayed
P_i^- (S_i^-)	the amount of time all stops preceding (succeeding) s_i (inclusive) can be advanced

$$F_i = \min_{i \leq j \leq q} \left\{ \sum_{i < p \leq j} W_p + (l_j - B_j) \right\} \tag{5.7}$$

F_i represents the largest possible delay in service time at stop s_i that does not cause any time window violations at other stops when feasibility must be maintained during insertion. Since infeasible solutions are allowed during routing, Cordeau et al. [6] suggests replacing $(l_j - B_j)$ by $(l_j - B_j)^+$. This new term represents the largest delay in service time at stop s_i that will not cause an increase in time window violations at other stops following stop s_i. Even if the time window for stop s_j is not satisfied in the current tour and stop s_j occurs after stop s_i, the delay in service time can nevertheless be increased by $\sum_{i<p\leq j} W_p$ at stop s_i without increasing the time window violations at stop s_j.

However, due to the multi-objective cost function considered in our model, care must be taken when delaying service time at stop s_i so as not to increase cumulative excess ride time or the violation of ride time. Increasing the start service time at stop s_i may increase the ride

Figure 5.7: IRAU - Cases RI, RII, and RIII.

time of requests whose origin stop is before s_i and destination stop is after s_i. Taking this into consideration, a modified forward time slack can be written as

$$F_i' = \delta(i) * \min_{i \leq j \leq q} \left\{ \sum_{i<p\leq j} W_p + \left(\min\left\{l_j - B_j, m_j - P_j\right\}\right)^+ \right\} \tag{5.8}$$

where P_j represents the ride time of the request whose destination stop is s_j if $n+1 \leq j \leq 2n$ and $P_j = 0$ otherwise and $\delta(i)$ is 1 if no passengers are in the vehicle when the vehicle arrives at stop s_i and $\delta(i)$ is 0 otherwise.

Due to the objective function considered, shown in Equation 1.1, it is clear that if passengers

are in the vehicle when the vehicle arrives at stop s_i, delaying the service time of stops after stop s_i can increase the overall excess ride time. Hence, such a delay is not desired.

It is possible that such a delay in service at stop s_i may increase the total route duration when s_i is not the (start) depot. Hence, an additional check must be made to ensure that such a delay will not increase the overall route duration.

5.2.1 Inserting a request

The insertion of a request i, or its origin and destination stops, into an existing tour can be summarized below. (modification of Cordeau et al. [6]).

1. Find all possible ways (j, k) and $j < k$ of inserting the origin and destination stops of request i into the tour and compute the schedule (after inserting these two stops) as follows:
   (a) Set $D_0 = e_1 - t(0, 1)$ (departure at start depot)
   (b) For $l = 1, \ldots, q-1$, update A_l, B_l, W_l and D_l
   (c) Set $B_q = D_{q-1} + t(q-1, q)$ (arrival at end depot)
   (d) For each origin stop s_j
      i. Compute F'_j, $W_T = \sum_{i<p\leq j} W_p$ and $H_T = \max\{0, H - (B_q - D_0)\}$
      ii. Set $B_j \leftarrow B_j + \min\{F'_j, W_T, H_T\}$; $D_j \leftarrow B_j + load_j$
      iii. Update B_l, D_l and W_l for each stop s_l for $l > j$ in tour v
   (e) Compute $COST^v_{j,k}$
2. Insert the origin and destination stops of request i into positions $(j, k)^*$ where

$$COST^{v^*}_{(j,k)^*} = \min_v \left(\min_j \left\{ \min_{k>j} COST_{(j,k)} \right\} \right)$$

5.2.2 Removing a request

Removing a request from a tour involves the removal of two stops: its origin and destination stops. Once the two stops are removed, the route is reconnected by linking the respective predecessor and successor of each deleted stop. This is followed by performing Steps 1d(i) to (iii) from Section 5.2.1 for each origin stop s_j that is scheduled after the removed origin stop.

Chapter 6

Tabu Search

In the previous chapter, we proposed two routing techniques, i.e., IRAU and IRDU, developed for the DARP as part of the cluster-first route-second approach, refer to Chapter 4 for discussion, considered in our research. In this chapter, we propose a clustering technique; clustering is performed via a Tabu Search algorithm.

Tabu Search (**TS**) is a point-based search scheme proposed by Glover [18, 19] and independently by Hansen [26], who called it the steepest ascent, mildest descent method. For an in-depth tutorial and surveys of TS, refer to Glover [18, 19, 23]. TS has been applied successfully to solve many combinatorial optimization problems. Glover [23] briefly surveys some of these applications, including; employee scheduling, probabilistic logic problems, space planning and architectural design, and neural network pattern recognition. To the best of our knowledge, two TS algorithms have been developed for the DARP.

Nanry et al. [32] develop a reactive TS implementation. In their formulation, they consider time window violation, capacity violation and total distance traveled. Solutions that violate time window and vehicle capacity constraints are allowed during searches. A single objective is considered, i.e., to minimize total time traveled. Randomly generated test instances of size 25, 50 and 100 were examined. The CPU times on an IBM RISC 5000 for all problems considered ranges from 0.001 to 19.18 minutes.

More recently, Cordeau et al. [6] develop a TS with vertex reinsertion heuristic. In their formulation, these authors consider time window violation, vehicle capacity, maximum route duration and maximum ride time constraints. The same objective sought by the authors above was considered, i.e., to minimize total distance traveled. The algorithm was tested on randomly generated instances with request sizes ranging from 24 to 144 and on six real-world data sets of size 200 and 295 provided by a Danish transporter. The CPU times on a Pentium 4, 2GHz computer for all problems considered ranges from 1.9 to 267.82 minutes. As indicated by these

authors, the consideration of maximum ride time and maximum route duration constraints in their formulations significantly complicates the evaluation of the neighborhood thus explaining the considerable computational effort observed.

The DARP modeled in our research is motivated by our experience working with transit agencies and also studies of real-world Dial-a-Ride systems considered in the literature [1, 2, 33, 48]. Several features set our model apart from those covered in the papers mentioned above. One difference is the multiple, competing, and often non-commensurate objectives considered in our model; only one objective is considered by these authors. As illustrated in Section 1.3, improving one objective can significantly worsen a competing objective. Hence, besides having flexibility in determining the types of schedules generated to reflect the policy considerations of the agency, the different competing objectives considered can better characterize the problem because in a real-world, the agency seeks to minimize system cost while not neglecting customer's satisfaction. Another difference is the specification of waiting at each stop while the vehicle is transporting passengers. On the one hand, disallowing waiting greatly constraints the type of schedules generated. On the other hand, no restriction on the amount of wait can result in an unchecked amount of idling. In addition to the specification of the maximum allowable waiting at any stop, no waiting is allowed at destination stops while the vehicle is transporting passengers. It is felt that while waiting is tolerated by passengers as new passengers are picked up, waiting at a destination stop is equivalent to keeping a customer in the vehicle even though the vehicle has arrived at the destination. Hence, it is not allowed in our model.

In the following sections, we introduce the TS algorithm that we developed for the version of DARP considered.

6.1 Fundamentals of Tabu Search

Starting from an initial solution, a TS moves to the best solution in the neighborhood. It continues to move from one solution to another until a stopping criterion is met. The unique characteristic of TS is its ability to use memory to guide the local descent search method beyond local optimality. When a local optimum is encountered, a move to the best neighbor is made even if this may cause a deterioration in the objective function value. This move is possible due to the notion of a Tabu List. The Tabu List records attributes of recent moves made and prohibits any repetition of such moves to prevent cycling. The duration that an attribute remains taboo is referred to as the Tabu List Tenure (or TLT). A special degree of freedom to overrule a taboo status of a move is possible by means of an aspiration criterion. If the present objective function value is better than the best objective function value found thus far, the aspiration criterion allows the descent method to move to this better solution.

6.2 Creating an initial solution

It has been shown by Ponterosso et al. [35] that metaheuristics, in general, benefit from starting with a good initial solution. Hence, it is desirable to develop a good initial solution generator. In view of the solution approaches, as shown in Section 4, an initial solution generator that complements both approaches is required. In particular, if all constraint requirements, as shown in Section 1.3 for constraints specification, must be met, the initial solution should reflect this criterion. Basically, if it is not possible to assign a request so that no constraints are violated, the request will be left unassigned. On the other hand, if all requests must be assigned regardless of constraint violations, then it is desirable that no requests are left unassigned in the initial solution. In addition to the approach consideration, the issue of resource utilization is also taken into account. Utilizing a minimum number of vehicles to service requests frees up other vehicles and thus drivers for secondary assignments. Even though minimizing fleet size is not defined formally as part of our objective, it is reasonable to assume that excessive usage of resources should be avoided. The number of vehicles employed in the initial solution will provide an upper bound on the number of vehicles required to service all requests. Naturally, the maximum number of vehicles available will serve as an absolute upper bound.

In developing an initial solution generator, an additional requirement is that the generator must be repeatable and be able to generate different solutions given a different start seed. This requirement is an important feature since it allows a modeler the ability to reproduce a particular solution, if needed, for further investigation or experimentation.

The initial solution generator developed in this research employs the routing techniques IRAU and IRDU (as presented in Section 5) to create an initial solution. The choice of router invoked is dictated by the scenario considered. There are essentially two phases to the generator: the first phase prioritizes requests while the second phase searches for the best feasible insertion of a request. This two-phase procedure utilizes a framework similar to insertion heuristics [28, 33] developed for the DARP.

Requests are first sorted in such a way that requests to be inserted first are given a higher priority. A rank is generated for each request to determine the order of insertion. A lower rank denotes a higher priority. The rank is computed based on a linear combination of two characteristics of each request: earliest service time and the length of the time window.

Earliest service time - A request with an earliest service time at a stop that occurs before other stops, should be assigned ahead of these stops in the tour. For a DPR, this will be the earliest service time at its origin stop. For a DDR, this will be the earliest service time at its destination stop.

Time window - Narrow time windows are more difficult to satisfy than wide ones. As a result, requests with narrower time windows are assigned first. Hence, the rank of request i can be computed as:

$$rank_i = c_1 * e_i + c_2 * tw_i$$

where $c_1, c_2 \in [0, 1]$.

Our initial solution generator can be described as follows: Requests are first sorted in non-decreasing order of their rank. Starting with a single vehicle, each request is inserted into the tour based on the best feasible insertion in the order of its rank. If it is infeasible to insert a request into the existing vehicles, increase the number of vehicles by one and insert this request into the new vehicle. Continue this process by assigning each request in the tour that minimizes cost. If the maximum number of vehicles has been employed and if it is not possible to insert a request feasibly in the existing tours, the request should be left unassigned if feasibility must be met. On the other hand if it is important that all requests be routed, then the request will be routed in the vehicle that results in the least constraint violations.

Note that the choice of c_j affects the rank of a request. Fixing the values of c_j allows the generation of the same solution which satisfies the repeatability requirement while randomly setting the values of c_j creates a more arbitrary ranking and hence, the possibility of a different schedule since the order in which a request is inserted affects how tours are generated. However, in the case where tw_i is fixed for all i, changing the values of c_1 and c_2 will not affect the rank of a request. In this case, we define $c_2 = rand(0, 0.1)$ for each request i where $rand(a, b)$ denotes a randomly generated real number with value between a and b. This random value can be created with a specific seed, hence ensuring repeatability of the initial solution generator.

6.3 Neighborhood Structures

The neighborhood, defined as a set of solutions which are a small perturbation from the current solution, to be considered determines the search region during each iteration of TS. Three neighborhood structures are considered in our algorithm. They are: (1) Single Request Move (SRM), (2) Paired Request Swap (PRS) and (3) Stops Swap. In all neighborhoods, the removal of a request from a vehicle is performed via the removal procedure described in the respective routing techniques. The same procedure is used for insertion. Refer to Section 5 for the routing techniques discussion.

A large search region may allow more solutions to be explored but is computationally expensive. Short-term memory structures are implemented to record move attributes for each neighborhood structure. The three neighborhood moves are discussed below:

Single Request Move (SRM)

SRM involves moving a request from its original vehicle to an alternate vehicle while leaving the vehicle assignment of the remaining requests unchanged. A single request move is performed by first removing the request, or its origin and destination stops, from the vehicle it is assigned to and then inserting the request into the best possible positions in the new tour. The removal and insertion of stops are performed using the respective routing technique employed. The number of neighborhood solutions considered in each iteration is of O(mn) with m vehicles and n requests.

The Tabu List structure for SRM is expressed as an $n \times m$ matrix.

$$TL_{SRM} = \begin{array}{c} \\ 1 \\ \vdots \\ i \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} v1 & \cdots & vj & \cdots & vm \end{array} \\ \begin{bmatrix} & & \vdots & & \\ & & \vdots & & \\ \cdots\cdots & & TL_{SRM}(i,j) & & \\ & & & & \\ & & & & \end{bmatrix} \end{array}$$

The entry $TL_{SRM}(i,j)$ represents the sum of the iteration at which request i was assigned to vehicle j plus TLT. Hence, request i cannot be moved to vehicle j whenever the current iteration, say k, is such that

$$k \leq TL_{SRM}(i,j).$$

It should be noted that this move is the only neighborhood structure considered that can result in a reduction of the number of vehicles utilized.

Paired Requests Swap (PRS)

Swapping requests i and j entails removing request i and j from their original tours v_i and v_j respectively where $v_i \neq v_j$ and then inserting requests i and j into the best point of insertion in the tour of vehicle v_j and v_i respectively. The Tabu List structure for PRS is expressed as an $n \times n$ matrix where n represents the total number of requests.

$$TL_{PRS} = \begin{array}{c} \\ 1 \\ \vdots \\ i \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & \cdots & j & \cdots & n \end{array} \\ \begin{bmatrix} & & \vdots & & \\ & & \vdots & & \\ \cdots\cdots & & TL_{PRS}(i,j) & & \\ & & & & \end{bmatrix} \end{array}$$

The entry $TL_{PRS}(i,j)$ represents the sum of the iteration at which request i and request j were swapped plus TLT. Hence, request i and j cannot be swapped again whenever the current iteration, say k is such that

$$k \leq TL_{PRS}(i,j).$$

If $n_1, n_2, \ldots, n_m$ represent the number of requests currently scheduled in vehicles $1, 2, \ldots, m$ and, without loss of generality, assume that $n_1 \geq n_2 \geq \ldots \geq n_m$, then the number of paired swaps considered in each iteration is $\mathrm{O}(m(n_1)^2)$ given m vehicles. Observe that every swap invokes the router twice. Hence, it is computationally expensive to consider all pairs of swaps. Instead of considering every pair, vehicles are randomly partitioned into groups and a local search is conducted in each sub-neighborhood. Naturally, the computational effort is dependent on the DARP problem size and the number of vehicles utilized.

Stops Swap Move (SSM)

SSM is applied to improve the route ordering in each tour by swapping two stops within a tour. Swapping of stops alters the ordering of stops within a tour. This allows the exploration of other possible stop sequences which in turn allows the determination of the best stop ordering of a given tour for the neighborhood structure considered. Improvement made is reflected either in the reduction of existing infeasibilities or the minimization of the overall objective function value. However, care must be taken when swapping stops to ensure the meeting of precedence relations of each od-pair. Refer to Section 1.3 for constraint specifications.

6.4 Intensification and Diversification Strategies

Intensification in TS allows exploitation of the neighborhood while diversification encourages exploration of the search space. From an initial solution, the solution undergoes a hierarchical employment of the neighborhood structures of SRM followed by PRS. From informal experimentation, it was found that the application of SSM does not improve the solution quality when the IRDU routing technique is employed. However, an improvement in solution quality

is observed when the IRAU routing technique is applied. Hence, SSM is invoked whenever a new best solution is identified after each neighborhood search when IRAU is employed as the routing technique.

At each iteration, intensification is carried out by moving to the best neighboring solution found as long as the move is not on the tabu list. However, a taboo move is allowed if the new solution is the best found so far. As the current solution is being updated, attributes pertaining to each move is updated in the Tabu List. Intensification of each neighborhood terminates when a previously specified number of unimproved solutions are encountered and in the case of a local search, intensification stops whenever there are no improvements. If no new solution is found after one round of intensification, diversification takes place. Diversification is carried out by randomly swapping requests from one vehicle with at most two other vehicles in the best solution. It has been observed that while performing too few swaps does not allow for sufficient diversification, performing too many swaps increases the likelihood of infeasibility. From informal experiments, the number of swaps performed between vehicles i and j, where n_i and n_j denote the number of requests respectively, is defined by

$$\min\left(m, \lceil \min\left(n_i, n_j\right)/4 \rceil\right) \quad \text{when IRAU is applied}$$

$$\min\left(m, \lceil \min\left(n_i, n_j\right)/2 \rceil\right) \quad \text{when IRDU is applied}$$

From the diversified solution, another round of intensification is performed. The different neighborhoods are investigated in succession repeatedly until a stopping criterion is met.

6.5 Stopping Criteria

Like most heuristics, the specification of a stopping criterion at which the heuristic will terminate must be determined. Several types of stopping criteria exist. Possible stopping criteria are:

- the current iteration is equal to the maximum allowable iterations
- the number of unimproved solutions equals the maximum allowable unimproved solution count
- the execution time exceeds the maximum allowable execution time

In our case, the algorithm terminates whenever the number of iterations equals the maximum allowable iterations.

6.6 Tabu Search Framework

The notations used to describe our TS metaheuristic are shown below. This is followed by an outline of the TS framework.

Symbol	Description
s	a DARP solution
x^*	best global solution
x	best local solution
$f(s)$	cost of solution s as expressed in Equation 1.1

Algorithm 6.6.1 TS

[Input] routing techniques: IRAU or IRDU
stopping criterion: Ω
parameters: c_1, c_2, *SRM_TLT* and *PRS_TLT*

[Output] returns best solution found

[TS1] Initialization
Initialize tabu lists with parameters *SRM_TLT* and *PRS_TLT*
Define *neighborhood-type* as follows:
0 - SRM
1 - PRS
neighborhood-type = 0
$u = 0$

[TS2] Generate an initial solution s using the initial solution generator with ranking parameters c_1 and c_2

[TS3] Update $x^* \leftarrow s$ and $x \leftarrow s$

[TS4] Intensification
$q = 0$
$\omega = 1$
while(q $\leq \omega$ or Ω is not met)**do**
Search *neighborhood-type.* For each solution s
if((s is tabu but $f(s) < f(x^*)$ or (s is not tabu but $f(s) < f(x)$)) **then**
$x \leftarrow s$
end if
if*($f(x) < f(x^*)$)* ***then***
$x^* \leftarrow x$
$q = 0$.
else
$q = q + 1$
end if

Update tabu status
end while
[TS5] If Ω is met, go to step [TS10]
[TS6] Invoke SSM neighborhood search when a new best solution is found
if IRAU is employed
[TS7] **if**(a best global solution is updated since the last neighborhood search) **then**
u = 0
else
u = u + 1
end if
[TS8] *neighborhood-type = (neighborhood-type + 1) mod 2*
[TS9] Diversification
if(*u = 2* and Ω is not met) **then**
create a new solution *s*
go to step [TS4]
end if
[TS10] Terminate TS and return the best solution

The sequence of steps described in the TS framework can be summarized as follows: The algorithm is first initialized in Step [TS1]. In Step [TS2], an initial solution is generated via the initial solution generator described in Section 6.2 and the solution is then updated as the *best* global solution in Step [TS3]. Step [TS4] is the intensification phase of TS. The algorithm continues searching[1] in this neighborhood until either more than one unimproving solutions are encountered or the stopping criterion for the algorithm is met. As indicated in Step [TS5], if the latter is true, Step [TS10] is invoked and the algorithm terminates. If IRAU is the routing technique selected for the run, SSM is invoked in step [TS6]. In Steps [TS7] and [TS8], u is updated followed by the *neighborhood-type*. Step [TS9] is the diversification phase of TS. Diversification is invoked as long as $u = 2$, that is, no better solution (when compared to the best) is found when the two neighborhoods (SRM and PRS) are considered consecutively, and the stopping criterion is not met. After diversification, intensification takes place. The algorithm terminates whenever the stopping criterion is met. The best solution is returned as indicated in Step [TS10].

[1]Two forms of searches can be selected; searches can be performed either via first search or best search. Unlike best search where all neighboring solutions are considered before the *best* solution is identified, in first search, the *first* solution found that is better than the current solution is selected. From informal testing, it was found that when comparing the two search selections, best search produces higher quality solutions than first search when the algorithms are allowed to run for the same amount of time. As a result, only best search is considered in subsequent experiments.

6.7 Results

As discussed in Section 4, the two approaches for solving the DARP considered are:

Approach I

Every solution must be feasible. Requests that cannot be serviced without violating any constraints will be left unassigned. This is carried out using the algorithm TSIRAU.

Approach II

Solutions need not be feasible but all requests must be assigned. This is carried out using the algorithm TSIRDU.

6.7.1 Random DARP instances

To the best of our knowledge, no test instances are available in the literature for the version of DARP studied in our research. To analyze the behavior of our algorithms developed for the two approaches, we have constructed a 30-request and a 150-request DARP instance. The instances are constructed in such a way that a feasible[2] solution with the given resources exists. This is particularly important in the case of *Approach I* where it is possible for requests to be left unassigned. Knowing in advance that the available resources (i.e., fleet size) are sufficient to meet the demand (i.e., all requests can be assigned), we can verify that the algorithm is indeed capable of routing all requests.

The DARP instances are constructed via a random DARP instance generator described in Appendix A.1. The characteristics of the two instances are summarized in Table 6.1.

6.7.2 Objective Function

The objective function dictates how the algorithm searches the solution space. The objective function, as shown in Equation 1.1 is given as

$$COST = \eta TT + \psi ERT$$

where $\eta, \psi > 0$, TT, and ERT denote the total travel time and excess ride time[3] respectively.

[2]We define the feasibility of a solution as one which meets all the constraints defined in Section 1.3
[3]The units for TT and ERT are reported in minutes in this research.

Table 6.1: Characteristics of randomly generated DARP instances

DARP instance	U30	U150
Service area (miles^2)	16	16
Number of requests n	30	150
Type of request	randomly generated	randomly generated
Time window (minutes)	10	10
Loading time (minutes)	5	5
Number of vehicles m	3	10
Vehicle average speed (miles/hour)	20	20
Number of ordinary seats	4	4
Number of wheelchair places	2	2
Average wait time (minutes)	30	30
Maximum ride time (minutes)	60	60
Maximum route duration (minutes)	480	480
Minimum trip length (minutes)	1.52	0.20
Maximum trip length (minutes)	13.99	13.99
Average trip length (minutes)	6.11	6.25
Number of DPRs	13	75

Even though the objective function can be set to be biased toward an objective, the objectives of the solution are not necessarily commensurate with the weight set. This is due to the fact that the different objectives are not commensurate. Hence, a weight set of $\eta = 1$ and $\psi = 1$ does not necessarily yield a solution where $TT \approx ERT$.

A tradeoff curve can be generated to explore the nature of the weights and how they contribute to the objective function. Based on the tradeoff curve, a set of suitable weights, for the specific data set, can then be selected according to the policy considerations of the agency. It should also be noted that like other math programming techniques, the algorithms work in such a way that two sets of weights may yield the same exact solution if they are proportional to one another, and when they are executed with the same random seed.

In the following sections, we conduct several experiments to gain insight into the performance of each algorithm on different sized DARP instances. For both algorithms, based on informal experimentations, we found the parameter settings shown in Table 6.2 to consistently produce good solutions.

The stopping criterion chosen in both algorithms were determined based on a tradeoff between computational effort and the quality of solutions. Both TSIRAU and TSIRDU are implemented in Java and compiled with the -O option. This option optimizes the byte-codes generated by the compiler resulting in quicker executions.

Table 6.2: Parameter settings for TSIRAU and TSIRDU algorithms

TL_{SRM} ..... $10 + n/m$
TL_{PRS} ..... $10 + n/m$
(TSIRAU) Ω ... 4×10^2
(TSIRDU) Ω ... 2×10^2

Stopping criterion: Ω denotes the maximum number of iterations considered

Table 6.3: Non-dominated solutions from tradeoff curve shown in Figure 6.1

TT	*ERT*
100.03	337.15
117.19	309.70
130.67	299.35
167.00	289.23
176.83	279.06
210.76	267.41
261.45	258.89
298.10	255.21
336.81	254.38
436.89	249.03

6.7.3 Generating and Using Tradeoff Curves

By varying weights on the objectives and applying TSIRAU (for *Approach I*) and TSIRDU (for *Approach II*) algorithms starting from different (randomly seeded) initial solutions, tradeoff curves for both U30 and U150 are obtained. The results obtained after applying the TSIRAU and TSIRDU algorithms to U30 are shown in Figures 6.1 and 6.2, respectively, while those for U150 are shown in Figures 6.3 and 6.4, respectively. The different symbols on the graphs represent solutions obtained with different sets of weights. Non-dominated solutions[4] from Figures 6.1–6.4 are reported in Tables 6.3–6.6 respectively; these solutions appear as points on the dotted line in the tradeoff curves and the tables report approximately evenly spaced (with respect to ERT) solutions.

Observe that different portions of the tradeoff curve depict certain operational policies and the solution chosen generally reflects the policy consideration of the transit agency. For

[4]Solutions that **dominate** other solutions but not themselves are called non-dominated solutions. We say a vector $\vec{x}$ **dominate** [5] a vector $\vec{y}$ if (1) $\vec{x}$ is *at least as good as* $\vec{y}$ for all objectives, and (2) $\vec{x}$ is *strictly better than* $\vec{y}$ for, at least, one objective.

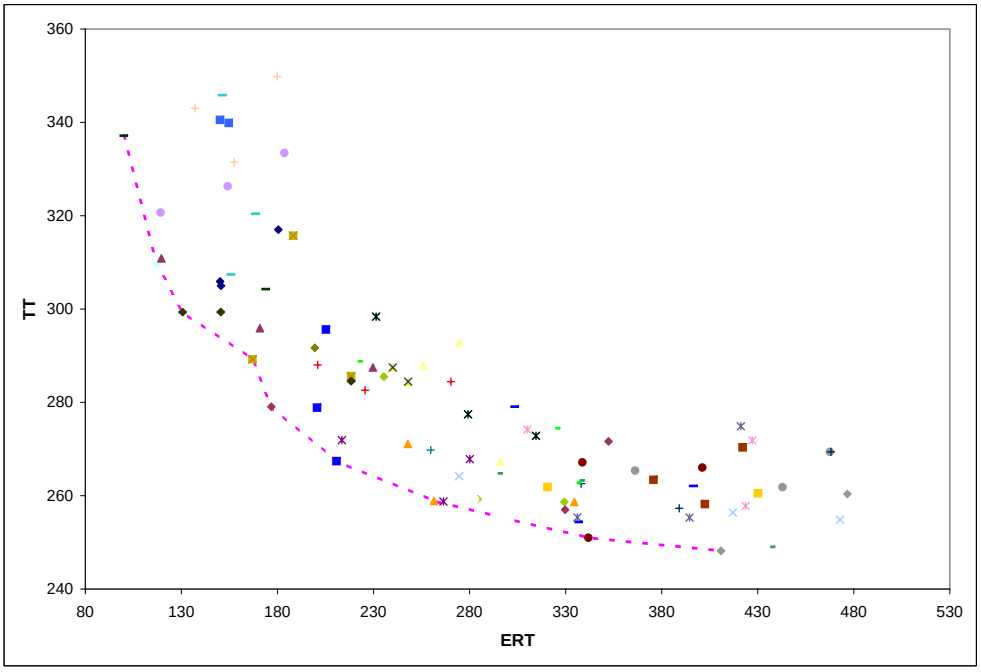


Figure 6.1: Tradeoff curve between total time traveled (TT) and excess ride time (ERT) generated for U30 using the TSIRAU algorithm

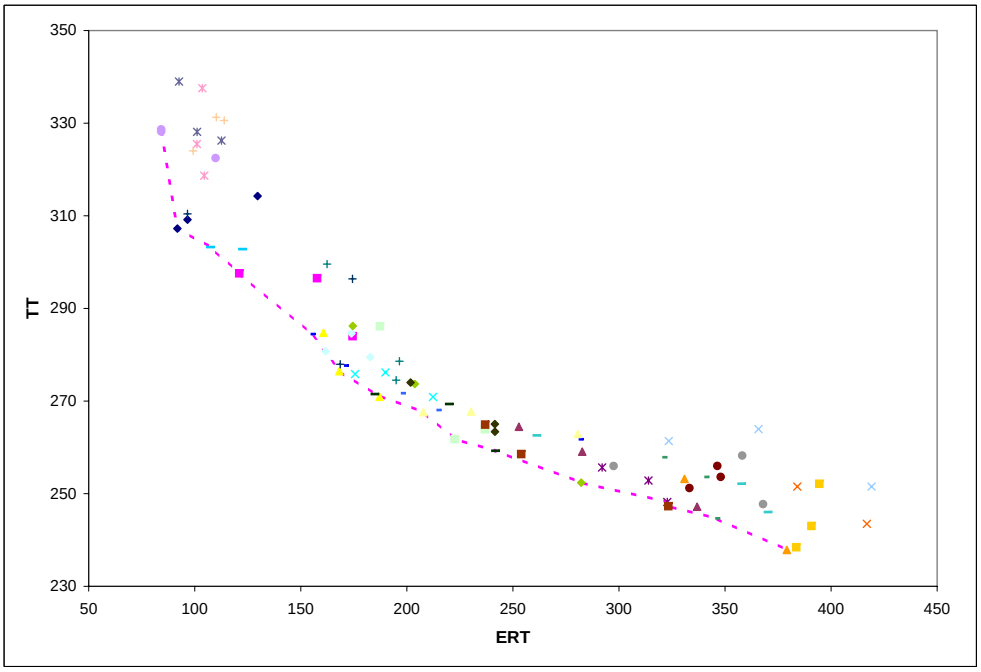


Figure 6.2: Tradeoff curve between total time traveled (TT) and excess ride time (ERT) generated for U30 using the TSIRDU algorithm

Table 6.4: Non-dominated solutions from tradeoff curve shown in Figure 6.2

TT	*ERT*
84.19	328.21
91.79	307.25
120.98	297.57
154.95	284.46
184.94	271.50
222.55	261.78
241.75	259.27
282.18	252.38
322.77	248.21
323.25	247.30
345.68	244.67
379.13	237.86

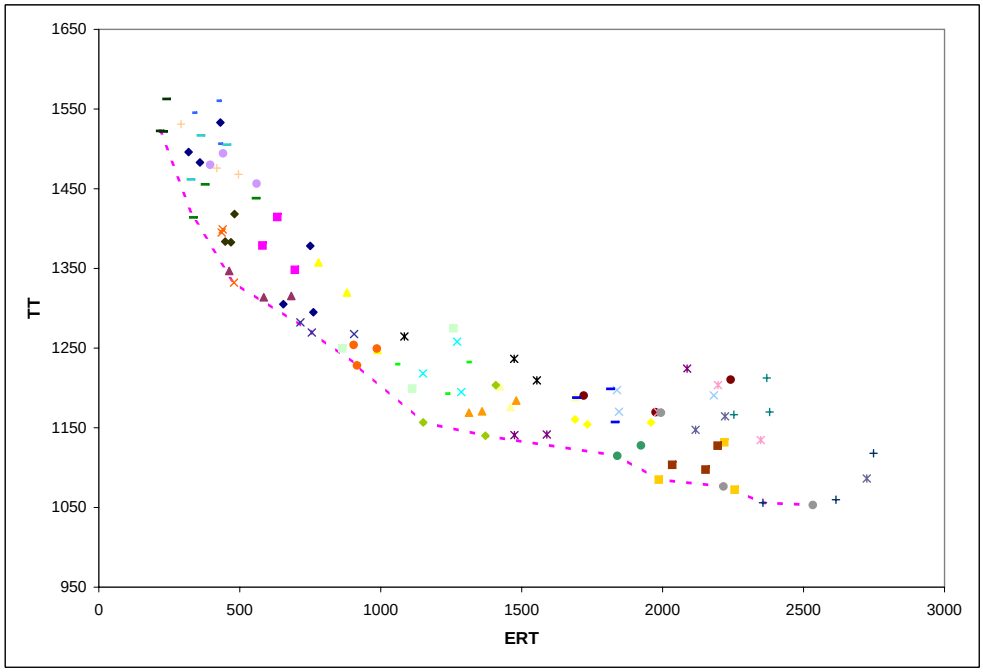


Figure 6.3: Tradeoff curve between total time traveled (TT) and excess ride time (ERT) generated for U150 using the TSIRAU algorithm

Table 6.5: Non-dominated solutions from tradeoff curve shown in Figure 6.3

TT	ERT
217.88	1522.59
335.10	1414.02
479.56	1332.13
755.69	1269.60
915.92	1228.28
1150.76	1156.78
1371.51	1140.04
1839.21	1114.87
1986.29	1084.97
2215.90	1076.33
2355.76	1055.90
2532.70	1053.10

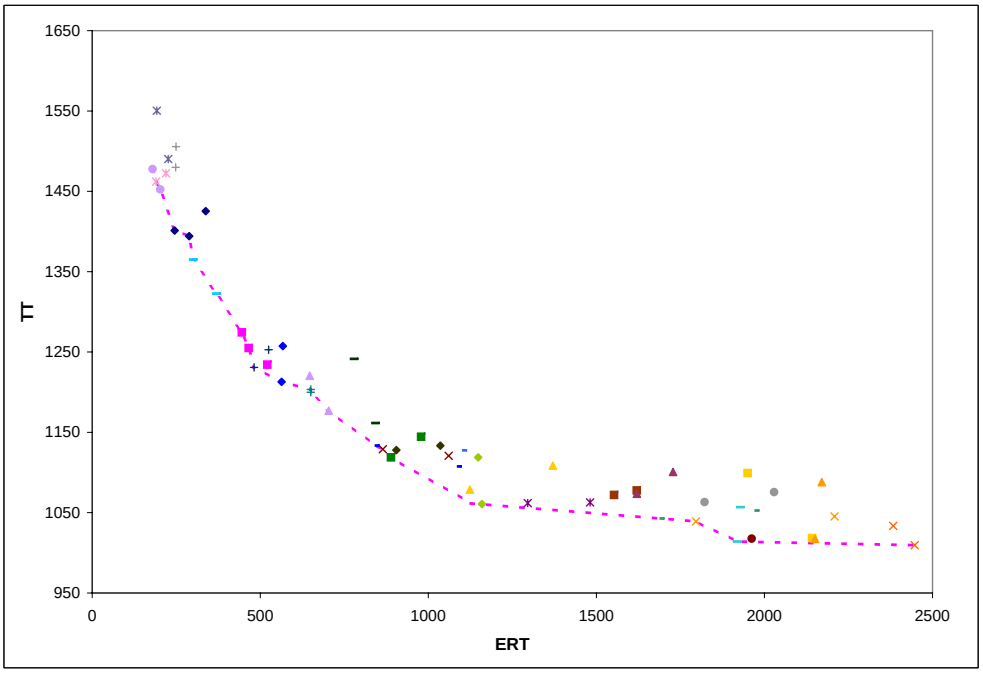


Figure 6.4: Tradeoff curve between total time traveled (TT) and excess ride time (ERT) generated for U150 using the TSIRDU algorithm

Table 6.6: Non-dominated solutions from tradeoff curve shown in Figure 6.4

TT	ERT
179.50	1477.75
245.34	1401.20
369.59	1322.66
481.50	1230.81
703.55	1176.96
888.83	1118.78
1126.48	1061.55
1796.24	1039.09
1918.59	1014.02
2447.04	1009.48

example, the top-left portion of the curve depicts a policy with high-level service emphasis while the lower-right portion of the curve depicts the situation where no level of service is implemented and the agency's main aim is to minimize system cost. Hence, the tradeoff curve provides a means to investigate the nature of the data and allows a dispatcher to determine a region which best represents the policy consideration of the agency. Typically, a modeler chooses a non-dominated solution from the tradeoff curve that best reflects the goals of the agency.

6.7.4 Variability Analysis

In this section, we investigate the variability of solutions generated when applying the TSIRAU and TSIRDU algorithms to the test problems, that is, U30 and U150. A side-effect of any practical heuristic is the variability of solutions obtained from different runs. However, with a properly selected stopping criterion, this variability should be small. Let us consider a hypothetical situation in which a policy maker has carefully examined the tradeoff curves shown in Section 6.7.3 and would like to use the algorithm to obtain a particular schedule that meets a certain policy. In particular, he would like to investigate the policy consideration in which *objectives TT and ERT have about the same value.* It is important to note that this particular policy considered is possible due to the nature of both U30 and U150 as depicted by the tradeoff curves. From the four tradeoff curves shown in Figures 6.1 – 6.4, this policy is achieved by setting the set of weights for both instances as shown in Table 6.7.

For the policy under consideration, we seek to find a solution that minimizes the cost, that is, $COST_{pol} = TT + ERT$, and satisfies the additional specification shown in Equation 6.1.

$$\min\{TT, ERT\} \geq 0.9 \max\{TT, ERT\} \tag{6.1}$$

Table 6.7: Weight sets identified for U30 and U150 to achieve the policy consideration where both objectives are approximately equal in value

Algorithm	Instance	η	ψ
TSIRAU	U30	31	5
	U150	9	2
TSIRDU	U30	31	5
	U150	11	2

Because the weights shown in Table 6.7 are used to guide the search and since the selection criteria are not specifically implemented in the algorithms, it is possible to generate solutions that do not satisfy the criteria. In our experiments, the solution with the minimum objective function value that satisfies the selection criteria is chosen.

Both algorithms TSIRAU and TSIRDU were run on U30 and U150 *once* for 2Ω iterations. Table 6.8 reports the percentage above the final solution obtained at $2^{-2}\Omega$, $2^{-1}\Omega$, and $2^{0}\Omega$ iterations. Here, Ω represents the pre-specified stopping criterion for each algorithm. In a separate experiment, we ran each algorithm on each instance for 10 runs each starting from arbitrary initial solutions. Table 6.9 reports the objective values and cost of the best solution, identified during the experiment. CPU times are in minutes on an Athlon XP 2600+.

Table 6.8: Results over varying fractions of stopping criterion generated by the TSIRAU and TSIRDU algorithms when each algorithm is applied once over U30 and U150 DARP instances

Instance	Algorithm	Vehicles	% above final solution found in run			Final Solution		
			$2^{-2}\Omega$	$2^{-1}\Omega$	$2^{0}\Omega$	TT	ERT	$Cost_{pol}$
U30	TSIRAU	3	0.84	0.00	0.00	**261.38**	**266.56**	527.94
	TSIRDU	3	2.78	2.78	2,78	**257.34**	**255.47**	512.81
U150	TSIRAU	10	25.96	12.92	5.69	**1189.29**	**1049.51**	2238.80
	TSIRDU	10	26.23	11.53	3.83	**1084.37**	**1076.54**	2160.91

From Table 6.9, it can be observed that the result is on average 2.81% from the best obtained in each experiment. Observe that the objectives, TT and ERT, are consistent with and reflect the policy selected based on the weight sets chosen for this experiment. It can also be deduced from Tables 6.8 and 6.9 that in general, better results can be obtained when the algorithms are run for a greater number of iterations. Nevertheless, solutions of very good quality are obtained for the prespecified stopping criterion for each algorithm.

It can also be observed that TSIRDU produced better results than TSIRAU for U150. However, it can be noted that TSIRAU converges earlier than TSIRDU. When dealing with larger sized DARP instances, this is a plus since with any practical heuristic, a tradeoff between time and solution quality is an important consideration.

Table 6.9: Analysis on the variability of solutions generated by the TSIRAU and TSIRDU algorithms

Instance	Algorithm	Vehicles	Run's Best Solution			Average[2]	Standard Deviation[3]	Average CPU (min)
			TT	ERT	$COST^1_{pol}$			
U30	TSIRAU	3	**265.35**	**266.83**	532.18	4.62	2.82	0.61
	TSIRDU	3	**251.62**	**278.51**	530.13	3.16	3.54	0.78
U150	TSIRAU	10	**1190.76**	**1202.51**	2393.27	2.70	1.92	4.55
	TSIRDU	10	**1118.53**	**1068.63**	2187.16	0.76	0.87	12.30
Average						2.81	2.29	4.56

[1] $COST_{pol} = TT + ERT$
[2] Average deviation (in percent) from best solution obtained during the run for all solutions found. Here, deviation $= 100 * \frac{|f - f^*|}{f^*}$ where $f = \eta TT + \psi ERT$, and f^* denotes the best solution found during the run.
[3] Standard deviation of computed deviations.

6.7.5 Performance Analysis

Heuristics developed for large optimization problems are often evaluated empirically, i.e., by applying the algorithm to an existing collection of specific instances and then comparing the observed solution and computational burden generated by existing algorithms in order to determine the performance of the heuristic developed. Refer to Rardin et al. [39] for a tutorial on the experimental evaluation of heuristics. Unfortunately, due to modeling variants and algorithm specifics, such a collection of instances for the version of DARP considered in this research is not available. In evaluating the performance of the algorithms developed, both algorithms were run over (a) a carefully constructed DARP instance with a built-in optimal solution and (b) a real-world DARP instance that has been previously studied and published by Baugh et al. [2].

In the following paragraphs, a summary of the above mentioned DARP instances will be given followed by a report of the results obtained.

Opt32 DARP Instance

Appendix A.1 describes the details involved in the construction of the data set. In creating Opt32, the parameters $n = 32$, $m = 4$, and $r = 1$ are used. The request table is summarized in Table 6.10. The optimal solution can be described as follows: Four vehicles are utilized and eight requests are serviced by each vehicle. Each tour starts (at the same time) from the depot and requests are serviced in the fashion of picking up request i and then request j followed by the dropping off of request i and then request j until all eight requests are serviced before returning to the depot. Figure 6.5 shows the circular tours represented in the optimal solution. Options exercised when running the algorithms are shown in Table 6.11.

Table 6.10: Summary of Opt32 and Opt80 request database

Instance	Opt32	Opt80
No. of requests	32	80
No. of available vehicles	4	4
No. of DPRs	32	32

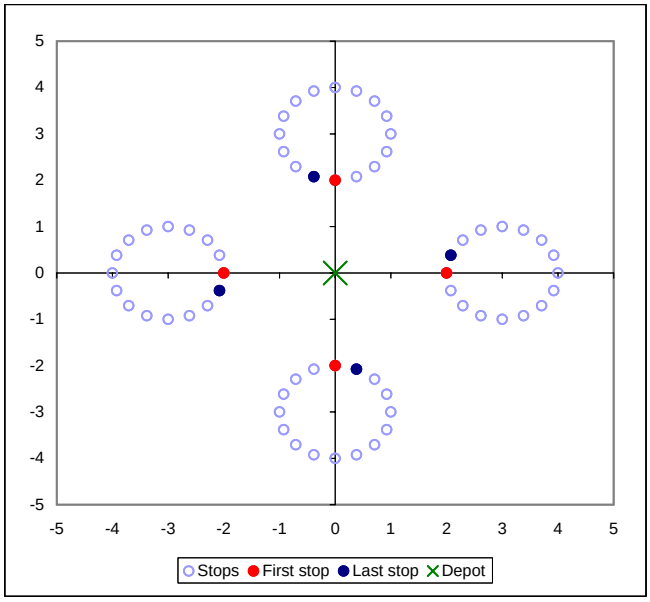


Figure 6.5: Tour pattern in the optimal solution of Opt32

Opt80 DARP Instance

Opt80 is created in the same manner as Opt32. The parameters used are $n = 80$, $m = 4$, and $r = 10$. The request table is summarized in Table 6.10 and options exercised when running the algorithms are similar to that for Opt32. Refer to Table 6.11 for options.

Real-world DARP Instance

The real-world DARP instance used in our study was provided by the Winston Salem Transit Authority (WSTA) in North Carolina and is the same data set previously studied by Baugh et al. [2]. The data set reflects the request demand for March 14, 1995. Table 6.12 summarizes the data.

Several observation should be noted here:

- Many requests are short trip requests. Many customers are senior citizens who live close to the hospital and day care center.

Table 6.11: Parameter settings on TSIRAU and TSIRDU for Opt32 and Opt80

Time Window	20 minutes
Vehicle Capacity (all vehicles)	4 ordinary seats
	2 wheelchair places
Average Speed	60 miles/hour
Fleet Size	4
Maximum Ride Time	60 minutes
Maximum Wait Time	30 minutes
Maximum Route Duration	480 minutes

Table 6.12: Summary of WSTA request database

Number of requests	332
Number of available vehicles	15
Number of DPRs	147
Number of DDRs	175
Minimum trip length (minutes)	0.10
Maximum trip length (minutes)	21.27
Average trip length (minutes)	3.84

- Most of the morning requests are DDRs and many of the afternoon requests are DPRs. Many of these are round-trip requests.
- Several of the requests contain the same od-pair.

The reader is referred to Baugh et al. [2] for a more complete discussion of the data set. These authors develop a cluster-first route-second approach where clustering is performed via a simulated annealing algorithm that incorporates the notion of a tabu list and routing is performed via a space-time nearest neighbor algorithm. In the model formulation, two time windows are specified for every request; one for the origin stop and one for the destination stop. Due to the specification of a fictitious time window (besides the negotiated time window) modeled in our algorithm, as discussed in Section 5, we were able to accommodate this two-time-window requirement modeled by these authors.

The multi-objective function addressed by these authors is one that minimizes total distance traveled, customers' disutility, and fleet size. Customers' disutility is quantified by the amount of time window violation. In their formulation, this includes time spent waiting to pick up a customer. In other words, no waiting is allowed at either the origin or destination stops. Constraints considered include *coupling*, *precedence*, and *capacity* constraints. Refer to Section 1.3 for the constraints. Options exercised in our runs are shown in Table 6.13.

Table 6.13: System specification for WSTA data

Time Window	40 minutes
Vehicle Capacity (all vehicles)	20 ordinary seats
	0 wheelchair places
Fleet Size	15
Average Speed	30 miles/hour
Maximum Ride Time	54 minutes
Maximum Wait Time	0 minutes
Maximum Route Duration	600 minutes

Results

Table 6.14 reports the objective values and the number of vehicles utilized by the best solution identified when both TSIRAU and TSIRDU on the WSTA data are applied *once* with varying weights and starting from arbitrary solutions. Table 6.15 shows the results published in Baugh et al. [2]. CPU times are in minutes on an Athlon XP 2600+.

Table 6.14: Results obtained by applying TSIRAU and TSIRDU once over varying weight sets for WSTA data

Weights		TSIRAU			TSIRDU		
η	ψ	Vehicles	TT	ERT	Vehicles	TT	ERT
16	1	8	1076.16	1596.34	6	1131.31	1838.89
4	1	7	1180.01	618.57	6	1146.30	495.84
1	2	8	1368.84	80.94	6	1420.85	106.69
CPU				136.21			659.00

Table 6.15: Previously published results for WSTA data in [2]

Method	Vehicles	TT	Disuitility
SA	10	1702	8
SA	9	1677	17
SA	11	1648	394
SA	11	1594	534
SA	15	1533	2123
WSTA[1]	14	2270	5697

[1] results from a commercial package used at WSTA

Table 6.16 reports the best solution obtained from running each algorithm five times on the Opt32 data. Since the algorithm developed by Baugh et al. [2] addresses a different objective, it is essential to stress that no direct comparison of the two procedures can be made. Referring

Table 6.16: Results obtained by applying TSIRAU and TSIRDU on Opt32 and Opt80 data

Weights		Optimal Solution			TSIRAU			TSIRDU		
η	ψ	Vehicles	TT	ERT	Vehicles	TT	ERT	Vehicles	TT	ERT
8	1	4	39.86	0.48	4	39.86	0.48	4	39.86	0.48
20	1	4	405.53	0.77	4	405.53	0.77	4	405.53	0.77
CPU							13.82			40.85

to Tables 6.14 and 6.16, several comments can be made about the results for the WSTA data and OPT32.

- It can be seen from the results that both algorithms, i.e., TSIRAU and TSIRDU, performed efficiently with the data sets.
- Even though minimizing fleet size is not an objective considered in our case, fewer vehicles were utilized in our solutions.
- In all solutions, disutility equals zero.
- In terms of computational effort, TSIRDU is much more expensive than TSIRAU.
- Both algorithms found the optimal solution for Opt32.

6.8 Summary

The results show that better solutions can be obtained when the algorithms are allowed to run for a longer number of iterations, the current chosen stopping criterion of $\Omega = 400$ and $\Omega = 200$ for TSIRAU and TSIRDU respectively produces good results at a reasonable computational effort.

The above results show that both TSIRAU and TSIRDU are useful heuristics for solving the multi-objective DARP. This is evident from the essentially better results obtained over previously published results as discussed in Section 6.7.5 and the optimal solution obtained over constructed DARP instances.

In general, TSIRDU produces better results than TSIRAU on the test problems, which suggests that limiting the exploration to only feasible solutions as in the case for TSIRAU may cause early convergence to suboptimal but rather good solutions. Though TSIRAU does not perform as well as TSIRDU in terms of solution quality, it is a fast algorithm and this proves to be an advantage when dealing with large-sized DARP instances.

Chapter 7

Scatter Search

In the previous chapter, Tabu Search was used in the clustering phase of the cluster-first route-second approach adopted in this research while routing was performed via one of the two insertion heuristics developed, as discussed in Section 5, to solve the multi-objective DARP. The two algorithms introduced were TSIRAU and TSIRDU. In both cases, a search begins from an initial solution. It continues to move from one solution to another until a stopping criterion is met. Scatter Search, in contrast, is a metaheuristic that starts from a set of initial solutions, normally two or more instead of one initial solution. To the best of our knowledge, prior to this research, a Scatter Search implementation for the DARP did not exist. In this chapter, a Scatter Search implementation for the DARP is presented.

In the following sections, we introduce the fundamentals of Scatter Search and discuss in detail how it is applied to the DARP.

7.1 Fundamentals of Scatter Search

Scatter Search (SS) is a population-based search scheme developed by Glover [20, 21]. Though developed in the 70s, it was only in recent years that more applications of SS started to appear in the literature. Refer to Glover et al. [22] for a preview of several applications of SS.

The fundamental principles and concepts of SS originate from strategies for creating composite decision and problem constraints. It operates on a relatively small population with a size generally not more than twenty. This population is called the *reference set*. Strategic designs based on exploiting the structure of the problem are used to operate on solutions; this limits room for randomization. New solutions are generated by combining solutions in the reference set. These solutions are then subjected to some improvement heuristic. Superior solutions replace those in the reference set.

An SS implementation consists of the following methods:

1. **Diversification Generation Method**: A collection of diverse trial solutions is generated given an arbitrary trial solution (or seed solution) as an input.

2. **Improvement Method**: The trial solution is transformed into one or more enhanced solutions. Feasibility of both before enhancement and after enhancement solutions is not a requirement even though the enhanced solution is normally feasible. If no enhancement takes place, the "enhanced" solution is considered to be the same as the original trial solution.

3. **Reference Set Update Method**: The reference set consisting of the best solutions must be organized in such a way that allows efficient access by other parts of SS. Besides a good organization method, alternate methods may also be included to provide other functional needs, e.g., adding a solution to the reference set or deleting a solution from the reference set.

4. **Subset Generation Method**: From the reference set, a new subset of solutions are generated. This set forms a basis for creating combined solutions. The most common subset generation method is the pairwise combination of reference solutions.

5. **Solution Combination Method**: The subset of solutions generated via the Subset Generation Method is transformed into one or more combined solutions. This is analogous to the GA crossover operator except that the Solution Combination Method must be capable of generating two or more solutions.

7.2 Diversification Method

The initial solution generator described in Section 6.2 can be adopted here to generate the initial set of solutions required to form the reference set. However, in SS, we require that this set of solutions be maximally diverse. Hence, we need to be able to define the notion of what it means for two solutions to be "maximally diverse."

When dealing with two binary strings, one natural distance measure is the Hamming distance. The Hamming distance is defined as the total number of bits that differ between two binary strings. More formally, the Hamming distance between two strings A and B is $\sum |A_i - B_i|$. In the case where only two vehicles are required, for example $[0, 1, 1, 0]$ and $[1, 0, 0, 1]$, the distance will be 4. However, these two solutions are analogous. In both cases, requests 1 and 4 are assigned to one vehicle and requests 2 and 3 to another. This simple example suggests that the Hamming distance is not an appropriate distance measure for our problem.

Before we can determine the diversity between two solutions, we need a measure that can be used to quantify the distance between two clusters. Let $(c_1^{(A)}, \ldots, c_m^{(A)})$ and $(c_1^{(B)}, \ldots, c_n^{(B)})$ represent clusters in solutions A and B respectively. The distance between each cluster in A and B can be calculated using the algorithm shown below.

We will use the following notation in describing the algorithm:

c_i	cluster i
$D(c_i^{(A)}, c_j^{(B)})$	distance between cluster i and j where $c_i^{(A)} \in A$ and $c_j^{(B)} \in B$.

Algorithm 7.2.1 Distance Matrix Generation
For each entry i, j in the distance matrix, do

[DMG1] Set $u = 0$ and $d = 0$.
[DMG2] **if**$(|c_i^{(A)}| \geq |c_j^{(B)}|\ )$ **then**
For each $k \in c_i^{(A)}$
if($c_{ik}^{(A)} \in c_j^{(B)}$) **then**
$u \leftarrow u + 1$
else
$d \leftarrow d + 1$.
end if
else
For each $k \in c_j^{(B)}$
if($c_{jk}^{(B)} \in c_i^{(A)}$) **then**
$u \leftarrow u + 1$
else
$d \leftarrow d + 1$
end if
end if
[DMG3] $D(c_i^{(A)}, c_j^{(B)}) = \max\left(\left||c_i^{(A)}| - |c_j^{(B)}|\right|, (|c_j^{(B)}| - u) \times d\right)$.

Once the distance matrix is generated, the diversity between two solutions, A and B, can be determined by solving the minimum weight matching problem. The Hungarian algorithm [30] can be adopted to solve this problem. The diversity between solutions A and B is denoted as *div(A,B)*.

Referring to our earlier example, $A = [0, 1, 1, 0]$ and $B = [1, 0, 0, 1]$, the distance matrix for A and B is given as follows:

		$c_1^{(B)}$	$c_2^{(B)}$
		$(2,3)$	$(1,4)$
$c_1^{(A)}$	$(1,4)$	4	0
$c_2^{(A)}$	$(2,3)$	0	4

It is clear that the minimum weight matching for this problem is 0. Hence, $div(A,B) = 0$. Based on the above discussion, one approach in generating an initial set of (diverse) solutions is to construct a new solution using information about the existing solutions set. Hence, given an existing set of solutions, S, a new solution can be constructed as follows: at each stage, one request is assigned to a vehicle that maximizes the minimum diversity between the new solution under construction and each solution in S. We will use the following notation in describing the diverse cluster construction heuristic:

S	set of (existing) solutions
n	number of requests
m_i	number of vehicles/clusters in solution i
s	new solution

Algorithm 7.2.2 Diverse Solution Construction Method
[Input]*:* a set of solutions S where $|S| \geq 1$.
[Output]*:* returns new solution s

[DSC1] Set $m_s = \min_{i \in S} \{m_i\}$.
[DSC2] For $k = 1,\ldots,n$,
  Set $D = 0$
  For $l = 1, \ldots, m_s$,
    Assign request k to vehicle l in s
    Compute $div(s,S_i)$ for each i.
    if$(\min_i \{div(s,S_i)\} > D)$ **then**
      $D \leftarrow \min_i \{div(s,S_i)\}$ and $l^* \leftarrow l$.
    end if
  Assign request k to vehicle l^* in solution s.
[DSC3] Route requests in each vehicle.
[DSC4] Return new solution s.

We propose a diversification generation method that incorporates the initial solution generation method discussed in Section 6.2 and the diverse solution construction method discussed above. A new solution is considered "maximally diverse" as long as the diversity between the new solution and every existing solution in the reference set is at least some factor times the maximum diversity known, i.e. $div(s,S_i) \geq fac \times div_{max}$.

7.3 Improvement Method

Improving a solution is analogous to intensification in Tabu Search. In SS, the neighborhood structures of a *single request move* (SRM) and a *paired requests swap*(PRS) as discussed in Section 6.3 can be selected individually or in combination to perform a local search on a solution. As the current solution is being updated, attributes pertaining to each move made is updated in an attractiveness matrix. The attractiveness matrix records the number of times two requests are assigned to a vehicle together. At the $(k+1)th$ iteration, for all i, j, we have

$$A_{k+1}(i,j) = \begin{cases} A_k(i,j)+1 & \text{if requests } i \text{ and } j \text{ are assigned to the same vehicle and } i \neq j \\ A_k(i,j) & \text{otherwise} \end{cases}$$

This attractiveness matrix is used in the combination method.

7.4 Reference Set Update Method

The reference set is represented as a $refSize \times 2$ matrix shown below:

$$\begin{bmatrix} COST(s_\alpha) & \alpha \\ COST(s_\beta) & \beta \\ \vdots & \vdots \\ COST(s_\gamma) & \gamma \end{bmatrix}$$

where $COST(s_\alpha) \leq COST(s_\beta) \leq \ldots \leq COST(s_\gamma)$. Here, $COST(s_\alpha), COST(s_\beta), \ldots$ and, $COST(s_\gamma)$ represent the cost for the corresponding solution stored in index $\alpha, \beta, \ldots$ and, γ in the solution set where $\alpha, \beta, \ldots, \gamma \in \{1, 2, \ldots, refSize\}$.

A new solution replaces a solution in the reference set only if its cost is less than $COST(s_\gamma)$.

7.5 Subset Generation Method

Given a reference set of *refSize* solutions and α ($\alpha \leq refSize$) of new solutions in the reference set, the number of subsets generated is given by

$$\frac{\alpha(\alpha-1)}{2} + \alpha\,(refSize - \alpha)\,.$$

In other words, the new subset of solutions considered are made up of a pairwise combination of all new solutions and a pairwise combination of each new solution with each old solution. Each subset contains at least one new solution.

7.6 Solution Combination Method

The Solution Combination Method needs to be able to combine solutions to produce two or more new solutions. In our case, solution combination is performed at the clustering phase. An $n \times n$ attractiveness matrix, where n represents the total number of requests, is maintained for each set of solutions generated in SS. A high attractiveness value indicates that two requests were often clustered together while a low attractiveness value indicates otherwise. Averaging the attractiveness values for two combined solutions produces a new attractiveness matrix for the new set of solutions. A new solution can then be generated by either assigning requests to vehicles based on *most*-attractive or *least*-attractive selections. The combination method based on *most*-attractive selection can be described as follows: Given a subset with solutions s_i and s_j, the number of vehicles utilized in the new solution will be given as $m = \min\{m_{s_i}, m_{s_j}\}$. Arbitrarily pick a request i and assign request i to the first vehicle. Next, select an unassigned request j^* such that $A(i, j^*) = \max_j \{A(i, j)\}$ and assigned j^* to the same vehicle as i. Continue in this process until n/m number of requests have been assigned to this vehicle. Repeat the process for subsequent vehicles until all requests are assigned. By the same token, the combination method can work in the case of *least*-attractive selection. In this case, the unassigned request j^* selected after request i has $A(i, j^*) = \min_j \{A(i, j)\}$.

7.7 Scatter Search Framework

The notations used to describe our SS metaheuristic are shown below. Following that, an outline of the SS framework is presented.

S	reference set
refSize	number of solutions desired
type	initial solution generation mechanism
	0 Initial Solution Generator
	1 Diverse Solution Construction Method
s	a subset of DARP solutions
x	a DARP solution
$\hat{x}$	an improved DARP solution

Algorithm 7.7.1 SS

[**Input**] routing techniques: IRAU or IRDU
size of reference set: *refSize.*
parameters: *fac, SRM_TLT* and *PRS_TLT*

[**Output**] returns best solution found

[SS1] Set $\boldsymbol{S} \leftarrow \emptyset$.
Set *type = 0.*
For improvement method:
Define tabu lists with parameters *SRM_TLT* and *PRS_TLT*
Define attractiveness matrix

[SS2] Generate solution x using *type* solution generation mechanism
Add solution x to $\boldsymbol{S}$
type = type + 1.
div_{max} = *div(x,S).*
if($|\boldsymbol{S}|$ = *refSize*) **then**
go to SS4.
else
go to SS3.
end if

[SS3] *attempts = 0*
while(*refSize* $\neq |\boldsymbol{S}|$ and *attempts* $<$ *2*) **do**
Generate solution x using *type* solution generation mechanism
Compute *div(x,*S_i*)* for *i = 1,...,*$|\boldsymbol{S}|$.
if($\min_i \{div(x, S_i)\} \geq fac \times div_{\max}$ and $s \notin \boldsymbol{S}$) **then**
add x to $\boldsymbol{S}$
attempts = 0
update div_{max}.
else
discard x
attempts = attempts + 1
end if
type = type + 1 mod *2*
end while

[SS4] Sort solutions in $\boldsymbol{S}$ in non-decreasing order of its cost
Make *NewSolutions* = TRUE.

[SS5] **while** (*NewSolutions*) **do**
Generate ***NewSubsets*** using the subset generation method
Make *NewSolutions* = FALSE.
while (***NewSubsets*** $\neq \emptyset$) **do**
For each subset $s \in$ ***NewSubsets***
Combine clusters in s to generate a new solution x.
Delete s from ***NewSubsets***
Apply improvement method to x to get $\hat{x}$
if($\hat{x} \notin \boldsymbol{S}$ and *COST(*$\hat{x}$*)* is better than the worst solution in $\boldsymbol{S}$), **then**
Replace worst solution in $\boldsymbol{S}$ with $\hat{x}$
Make *NewSolutions* = TRUE.

else
discard $\hat{x}$ and delete s from ***NewSubsets***
end if
end while
end while

[SS6] Apply the improvement method on the best solution

[SS7] Terminate SS and return the best solution.

The SS framework can be summarized as follows: The algorithm is first initialized in Step [SS1]. In Steps [SS2] – [SS4], the diversification method is employed to generate the reference set **S** which is then updated via the reference set update method. In Step [SS5], the subset generation method is applied to create new subsets of solutions. These subsets of solutions are then combined via the combination method and each new solution created undergoes improvement via the improvement method. This process continues until no new subset of solutions can be generated. In Step [SS6], the best solution obtained undergoes another round of improvement. Finally, the algorithm terminates in Step [SS7] and the best solution is returned.

7.8 Results

In this section, we discuss the results obtained by using the SS algorithm in combination with the two routing techniques discussed in Section 5. To continue our investigation of the two approaches as discussed in Section 4, we have:

Approach I

Solution must be feasible. Requests that cannot be serviced without violating any constraints will be left unassigned. This is carried out using the algorithm SSIRAU.

Approach II

Solution need not be feasible but all requests must be assigned. This is carried out using the algorithm SSIRDU.

Informal tests focused on selecting the appropriate neighborhood, as discussed in Section 7.3, for the improvement method of the SSIRAU and SSIRDU algorithms. Table 7.1 summarizes the selections for each algorithm. These settings were found to produce a wide variety of reasonable quality solutions while considering computational effort.

Table 7.1: Neighborhood considerations for the SSIRAU and SSIRDU algorithms

	SRM	PRS
SSIRAU	YES	YES
SSIRDU	YES	NO

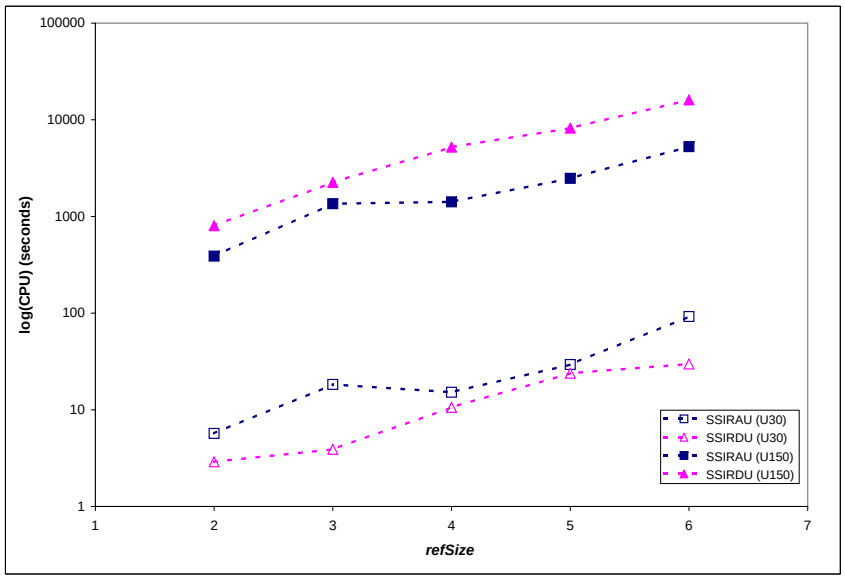


Figure 7.1: Effect of *refSize* over computation time for algorithms SSIRAU and SSIRDU on U30 and U150.

As no firm stopping criterion exists for SS, the specification of the size of the reference set indirectly determines how long the algorithm will run. To study the effect of varying *refSize* on the total computational time, both SSIRAU and SSIRDU algorithms were run (ten times each) with *refSize* taking on values from two to six over the DARP instances U30 and U150. Figure 7.1 shows the plot of average *log(CPU time)* versus *refSize* for both algorithms on U30 and U150 found during these runs. CPU times reported are on an Athlon XP 2600+. Figure 7.2 shows the corresponding average cost ratio obtained. The cost ratios are computed by taking the cost obtained over the best cost found during these runs. The cost ratio for run i is computed as

$$cost\ ratio\ i = \frac{cost_i}{best}$$

Referring to Figure 7.2, in the case of SSIRAU, a *refSize* of at least three should be set in order to achieve a cost ratio that is less than 10% of the best found in the experiments. As for SSIRDU, the results show that all solutions are within 10% of the best found and a small difference in solution quality was observed for varying *refSize* values. This suggests that a

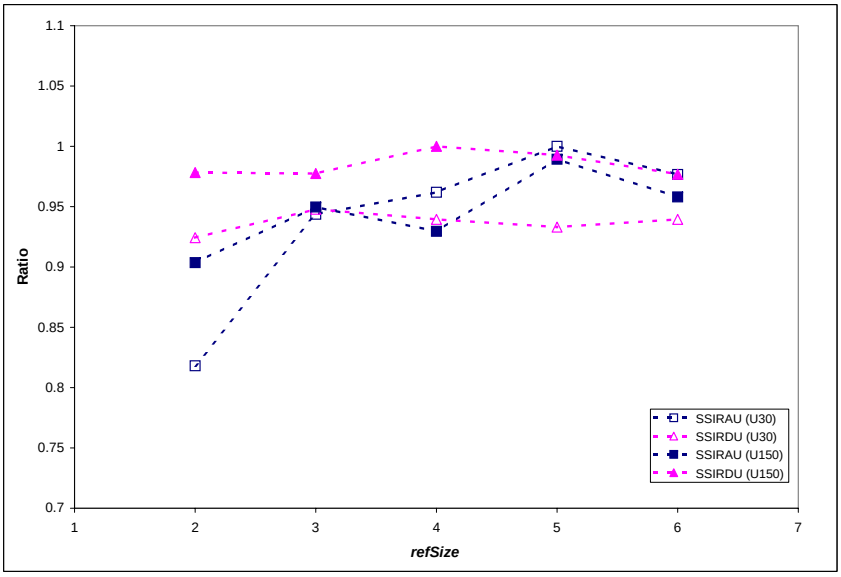


Figure 7.2: Effect of *refSize* over cost ratio for algorithms SSIRAU and SSIRDU on U30 and U150.

small reference set may suffice. It can also be noted that for the same *refSize*, SSIRDU requires a (significantly) larger computational time than SSIRAU. In addition, for varying *refSize*, the computational time for U150 ranges from 50 to 90 times more than U30 for SSIRAU while those for SSIRDU ranges from 170 to 570 times more. In the case of SSIRDU, the computational time is long. Taking into consideration the computational time and solution quality, the parameter settings summarized in Table 7.2 are used for the SSIRAU and SSIRDU algorithms.

Table 7.2: Parameter settings for SSIRAU and SSIRDU algorithms

SSIRAU	*refSize*	4
	Improvement method ...	SRM followed by PRS
SSIRDU	*refSize*	2
	Improvement method ...	SRM only

By varying weights on the objectives and applying SSIRAU (for *Approach I*) and SSIRDU (for *Approach II*) algorithms starting from different (randomly seeded) initial solutions, tradeoff curves for both U30 and U150 were plotted. As in Section 6.7.3, the tradeoff curves provide the modeler with different solutions to choose from. Particularly, this allows her to pick a solution that best represents the agency's goal. The tradeoff curves for U30 are shown in Figures 7.3 and 7.4 while those for U150 are shown in Figures 7.5 and 7.6. The different symbols on the graphs represent solutions obtained with different sets of weights. Non-dominated solutions are reported in Tables 7.3–7.6; approximately evenly spaced (with respect to ERT) solutions are

selected from corresponding tradeoff curves shown in Figures 7.3–7.6 respectively where these solutions appear on dotted lines in the curves.

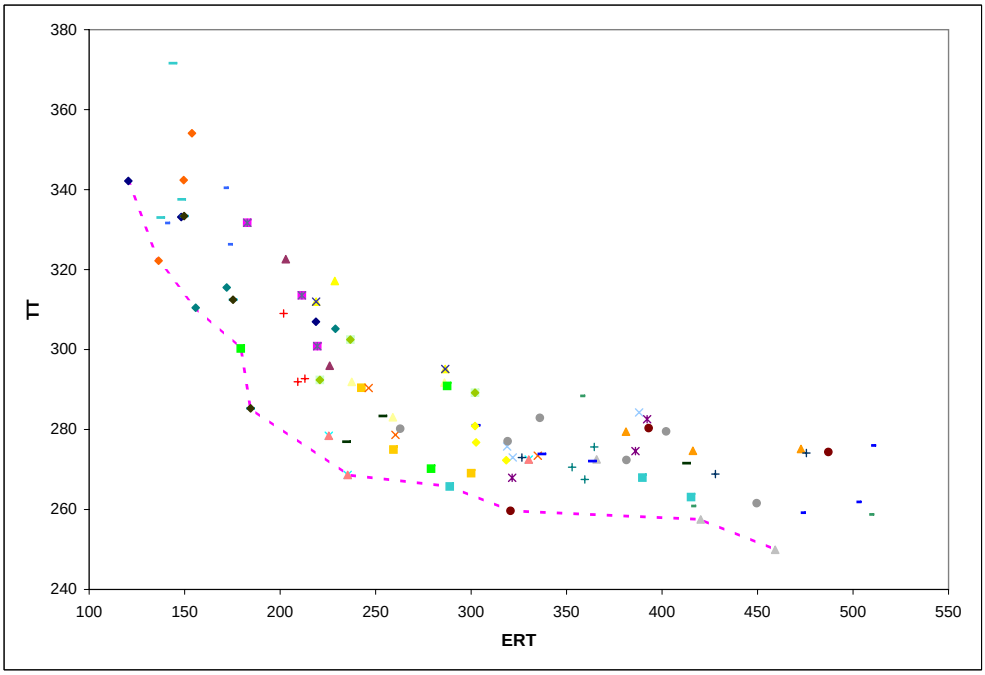


Figure 7.3: Tradeoff curve between total time traveled (TT) and excess ride time (ERT) generated for U30 using the SSIRAU algorithm

Using the tradeoff analysis conducted above, as shown in Section 6.7.4, we consider the policy consideration in which *objectives TT and ERT have about the same value*. The weight sets chosen are shown in Table 7.7. The selection criterion described in Section 6.7.4 is adopted in the determination of a solution.

We ran each algorithm ten times on U30 and U150 DARP instances. Refer to Section 6.7.1 for the description of these DARP instances. The results of the best solutions obtained are reported in Table 7.8. We repeated the experiment on Opt32 and Opt80. Refer to Section 6.7.5 for the description of these DARP instances. The results of the best solutions obtained are reported in Table 7.9.

In all solutions, no requests were left unassigned when the SSIRAU algorithm was applied. It can be observed that the variability in solutions generated for U30, when SSIRDU is applied, is relatively high. This is consistent with the "spread out" solutions shown in Figure 7.4. While increasing *refSize* may reduce this variability since there is a higher probability for combinations to take place between solutions, the cost ratios obtained over varying *refSize* as shown in Figure 7.2 show that increasing *refSize* does not necessarily produce higher quality solutions. While SSIRDU is outperformed by SSIRAU on U30, it generated higher quality

Table 7.3: Non-dominated solutions from tradeoff curve shown in Figure 7.3

TT	ERT
120.50	342.16
136.37	322.22
155.78	310.46
179.39	300.27
184.56	285.30
235.37	268.64
288.81	265.76
320.63	259.66
420.22	257.51
459.21	249.93

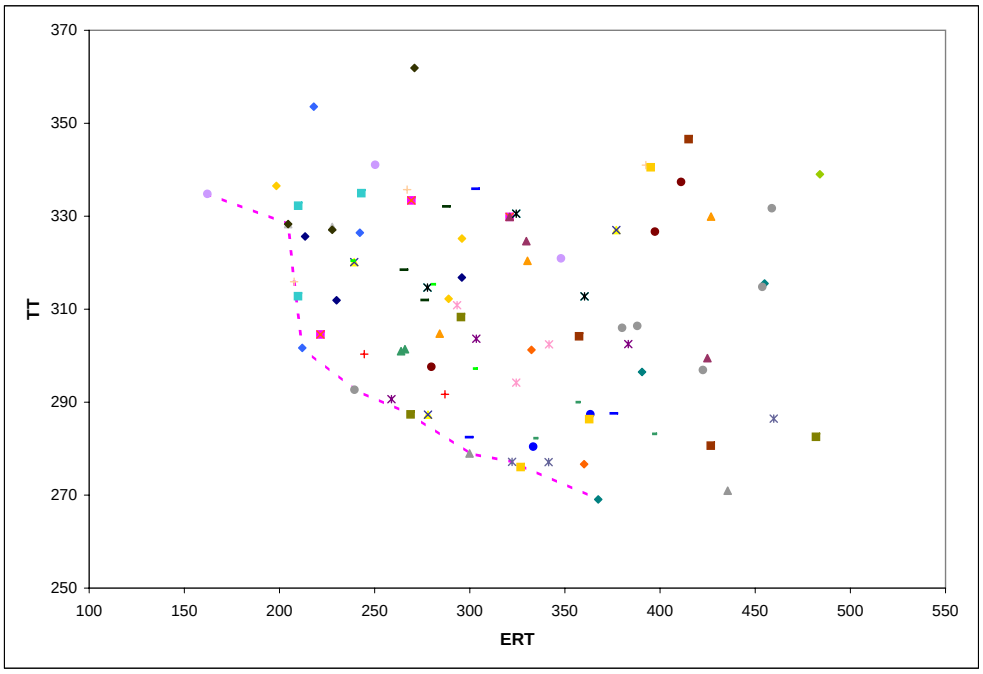


Figure 7.4: Tradeoff curve between total time traveled (TT) and excess ride time (ERT) generated for U30 using the SSIRDU algorithm

Table 7.4: Non-dominated solutions from tradeoff curve shown in Figure 7.4

TT	*ERT*
162.10	334.82
204.54	328.32
207.71	315.91
211.96	301.68
239.41	292.67
268.89	287.37
299.88	278.96
322.21	277.16
367.49	269.08

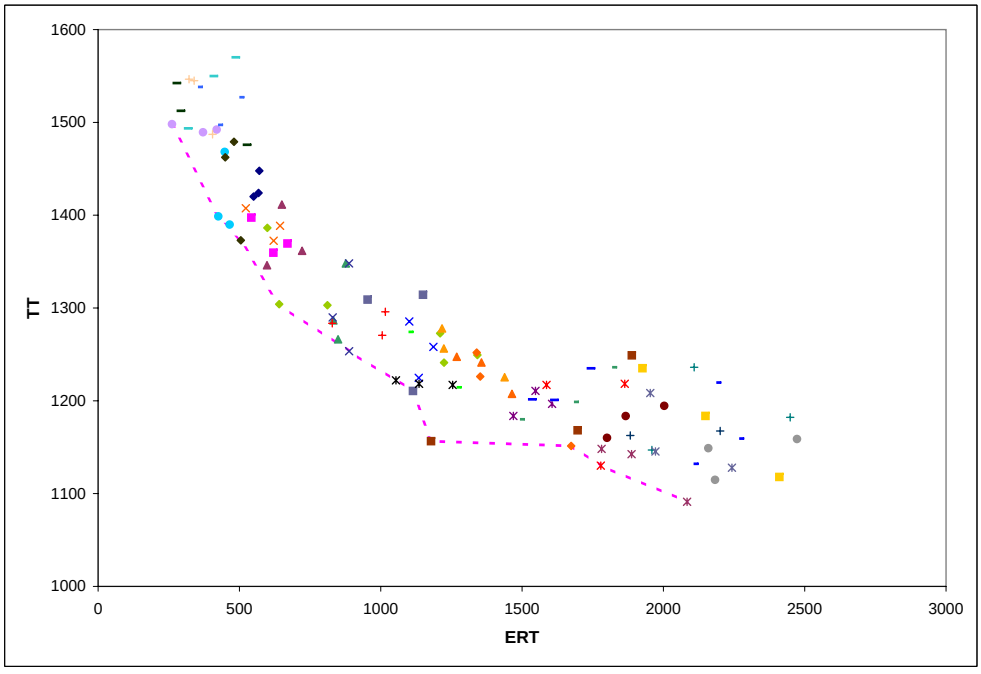


Figure 7.5: Tradeoff curve between total time traveled (TT) and excess ride time (ERT) generated for U150 using the SSIRAU algorithm

Table 7.5: Non-dominated solutions from tradeoff curve shown in Figure 7.5

TT	ERT
261.59	1498.12
425.29	1398.74
504.89	1372.92
640.55	1304.08
888.07	1253.41
1114.08	1210.63
1178.02	1156.49
1673.77	1151.37
1778.74	1130.14
2083.69	1091.20

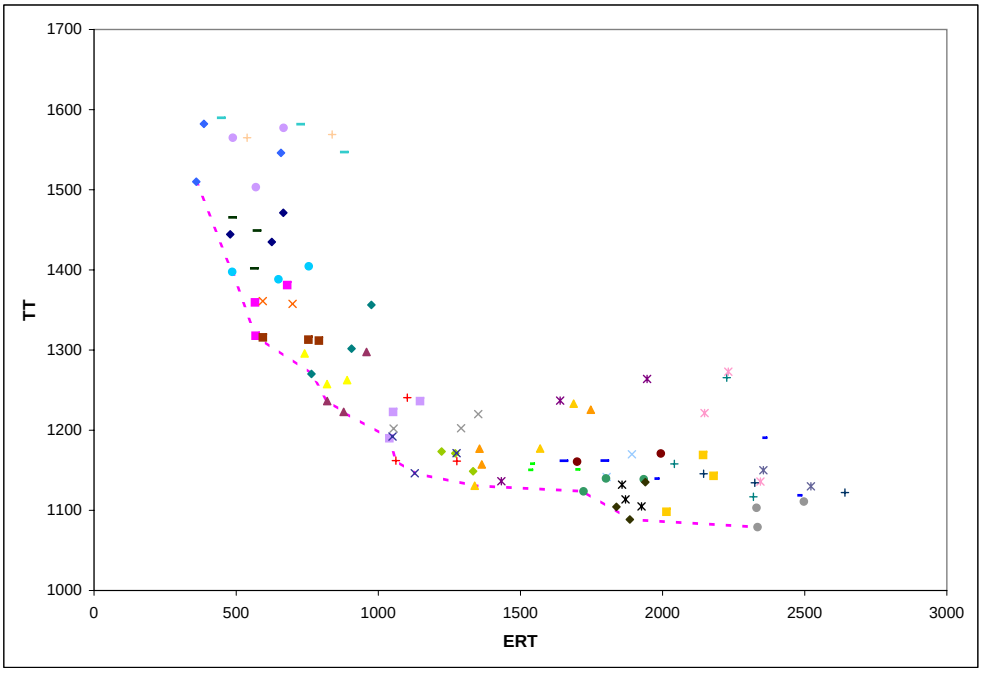


Figure 7.6: Tradeoff curve between total time traveled (TT) and excess ride time (ERT) generated for U150 using the SSIRDU algorithm

Table 7.6: Non-dominated solutions from tradeoff curve shown in Figure 7.6

TT	ERT
359.88	1510.04
486.28	1397.60
568.37	1317.83
764.72	1270.22
820.35	1236.40
878.30	1222.94
1039.01	1189.95
1061.71	1162.02
1128.17	1146.25
1338.92	1130.71
1721.98	1123.72
1884.98	1088.54
2334.18	1079.08

Table 7.7: Weight sets identified for U30 and U150 to achieve the policy consideration where both objectives are equally important. These weights are determined from the tradeoff curves shown in Figures 7.3, 7.5, 7.4, and 7.6

Algorithm	Instance	η	ψ
SSIRAU	U30	5	1
	U150	5	1
SSIRDU	U30	5	1
	U150	9	2

Table 7.8: Analysis on the variability of solutions generated by the SSIRAU and SSIRDU algorithms when applied to U30, and U150

Instance	Algorithm	Vehicles	Run's Best Cost			Average[2]	Standard Deviation[3]	Average CPU (min)
			TT	ERT	$COST^1_{pol}$			
U30	SSIRAU	3	**277.97**	**254.29**	532.26	2.90	2.02	0.32
	SSIRDU	3	**286.24**	**260.85**	547.09	6.47	7.72	0.032
U150	SSIRAU	10	**1189.77**	**1083.87**	2273.64	3.58	2.23	25.88
	SSIRDU	10	**1184.42**	**1081.74**	2266.16	2.68	2.07	17.26
Average						2.61	3.51	11.55

[1] $COST_{pol} = TT + ERT$
[2] Average deviation (in percent) from best solution obtained during the run for all solutions found. Here, deviation $= 100 * \frac{|f - f^*|}{f^*}$ where $f = \eta TT + \psi ERT$, and f^* denotes the best solution found during the run.
[3] Standard deviation of computed deviations.

Table 7.9: Results obtained by applying SSIRAU and SSIRDU on Opt32 and Opt80 data

Weights		Optimal Solution			SSIRAU			SSIRDU		
η	ψ	Vehicles	TT	ERT	Vehicles	TT	ERT	Vehicles	TT	ERT
8	1	4	39.89	0.48	4	39.89	0.48	4	39.89	0.48
20	1	4	405.53	0.77	4	405.53	0.77	4	405.53	0.77
CPU							1.94			4.48

solutions than SSIRAU on U150.

It can also be observed that a greater increase in computing time with respect to problem size is more prominent for SSIRDU than SSIRAU. In fact, the average computing time for SSIRDU is more than 500 times longer when applied to U150 as compared to U30. On the other hand, only a hundred fold increase in computing time is observed when SSIRAU is applied to U150 as compared to U30. This is true despite the fact that a larger *refSize* number is used for SSIRAU. With a population-based search, the longer computing time for each iteration of IRDU contributes significantly to the run time of the algorithm. This suggests that with larger DARP instances, SSIRDU may not be practical. In the case of Opt32 and Opt80, note that the optimal solutions are obtained for both SSIRAU and SSIRDU.

In another experiment, we ran SSIRAU and SSIRDU on the WSTA data, as shown in Tables 6.12 and 6.13, *once* with varying weights, each time starting from an arbitrary solution. Table 7.10 reports the objective values and the number of vehicles utilized in the best solution.

Several comments can be made about the results for the WSTA data.

- It can be seen from the results that both algorithms, i.e., SSIRAU and SSIRDU, performed efficiently with the data sets.
- Even though minimizing fleet size is not an objective considered in our case, no more

Table 7.10: Results obtained by applying SSIRAU and SSIRDU once over varying weight sets for WSTA data.

Weights		SSIRAU			SSIRDU		
η	ψ	Vehicles	TT	ERT	Vehicles	TT	ERT
16	1	8	1108.48	1959.26	8	1091.53	1138.77
4	1	8	1160.70	628.47	9	1262.10	341.55
1	2	9	1360.04	101.47	9	1407.10	82.47
CPU				124.80			1139.35

vehicles were utilized in our solutions as compared to that of Baugh et al. [2] as seen in Table 6.15.

- In all solutions, disutility equals zero.
- In terms of computational effort, SSIRAU is reasonable for the problem size considered while SSIRDU is impractical.
- Fewer vehicles are utilized for SSIRAU as compared to SSIRDU.

7.9 Summary

In this chapter, we presented two Scatter Search algorithms for the DARP. The foremost issue that we had to consider when dealing with a population-based algorithm like Scatter Search is the time required for solution evaluation since this greatly affects the overall runtime of the algorithm. As evident from applying the SSIRDU algorithm to the WSTA data, the higher computing time as compared to the SSIRAU algorithm significantly renders the algorithm less efficient when solving large sized DARPs. Even though essentially better results were obtained over previously published results as discussed in Section 6.7.5, the high computing time required indicates that the SSIRDU algorithm is impractical for solving the WSTA data since time is an important factor in real world operations.

In terms of the test problems, SSIRAU outperforms SSIRDU on U30 both in terms of solution quality and the smaller percentage variability of generated solutions from the best found in the runs. However, the reverse is true for U150.

Chapter 8

Comparison of TSIRAU, TSIRDU, SSIRAU and SSIRDU

8.1 Comparison using U150 test problem

In this section, we compare the performance of TSIRAU, TSIRDU, SSIRAU, and SSIRDU on U150. Refer to Section 6.7.1 for a description of U150. The majority of the previous runs take less than 25 minutes. In this analysis, each algorithm was run for about 25 minutes. Therefore the current setting for SSIRAU and SSIRDU will suffice since the computational time is about 25 minutes, on average. To achieve this computational time, Ω was increased from 400 to 1500 for TSIRAU and the Ω for TSIRDU was increased from 100 to 400. All other parameters were as in Tables 6.2 and 7.2.

Figure 8.1 shows that, for the current parameters selected, the TSIRDU heuristic clearly outperforms the other heuristics in terms of quality of solution generated if time is not a factor. TSIRAU performed better than the other algorithms initially, particularly for the first three minutes. From three to slightly over four minutes, SSIRDU produced better results than the other heuristics. After slightly past four minutes, TSIRDU was clearly better than the rest. Since it is reasonable to think that taking at least five minutes to generate a solution is acceptable, TSIRDU is the best candidate algorithm for this particular data set.

Using the parameter settings as indicated above, tradeoff curves for TSIRAU, TSIRDU, SSIRAU, and SSIRDU were generated. The resulting tradeoff curves are plotted in Figure 8.2. Non-dominated solutions are reported in Table 8.1.

Figure 8.2 shows that with the parameters selected, TSIRDU clearly outperforms the other three metaheuristics. This is evident by the fact that all solutions on the TSIRDU tradeoff curve are non-dominated solutions. It can also be observed that the next better performing

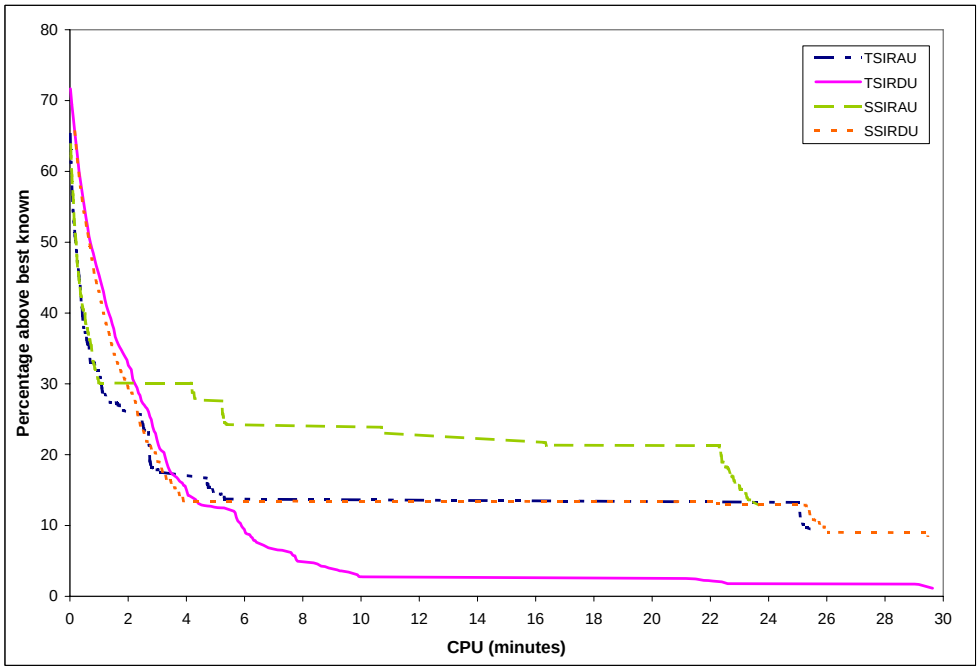


Figure 8.1: Comparison of TSIRAU, TSIRDU, SSIRAU, and SSIRDU on U150

metaheuristic is TSIRAU which generated better solutions for the top-left and lower-right portions of the tradeoff curve when compared to SSIRAU and SSIRDU. However, SSIRDU produced better solutions than TSIRAU at a few points in the middle of the tradeoff curve. Comparing SSIRAU with SSIRDU, one can deduce that SSIRAU generated better solutions than SSIRDU over the top-left portion of the tradeoff curve while SSIRDU yielded better solutions for the middle to lower-right portions of the tradeoff curve. Of course the fact that TSIRDU outperforms the other metaheuristics for U150 does not necessarily mean that it is a good DARP heuristic. In the following section the four metaheuristics are compared in terms of their performance on the WSTA data.

8.2 Comparison using WSTA data

In this section, the performance of the four metaheuristics on two sets of WSTA data is compared. The first set of data (or WSTA1) is presented in Section 6.7.5. The second WSTA data set (or WSTA2) is from June 7, 2002. Table 8.2 summarizes the data. WSTA2, like WSTA1, has many short trip requests and most of the morning requests are DDRs while many of the afternoon requests are DPRs. Many of these requests are round-trip requests.

The results for WSTA1 shown in Table 8.3 are a consolidation of results reported in Tables 6.14 and 7.10 when the four metaheuristics are applied to WSTA1 *once*. The experiment

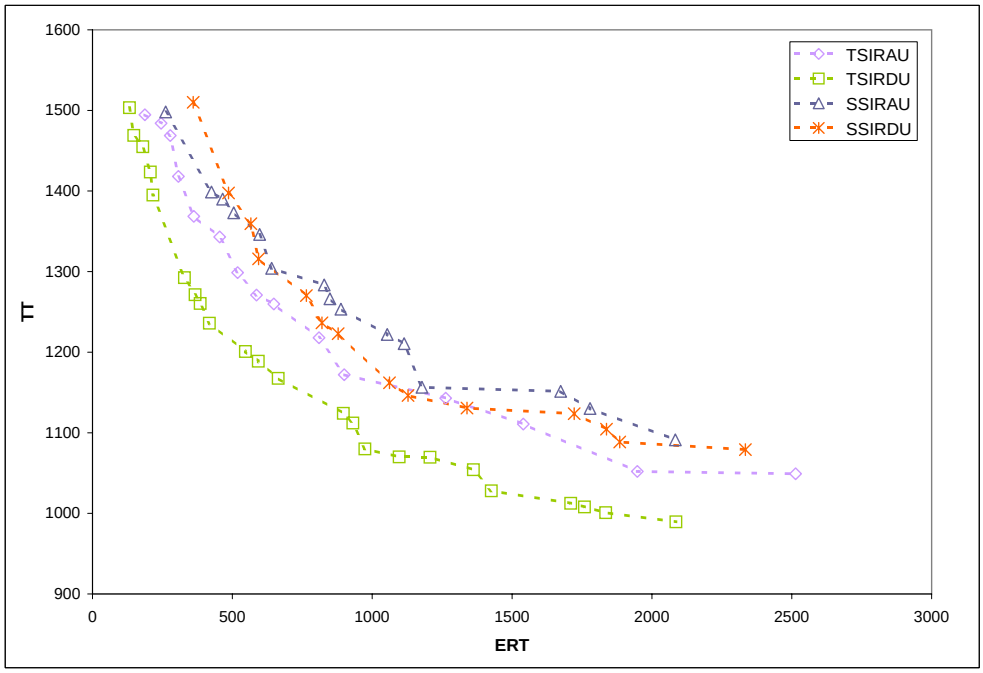


Figure 8.2: Comparison of tradeoff curves generated when applying TSIRAU, TSIRDU, SSIRAU, and SSIRDU on the U150

was repeated for WSTA2 with the exception that SSIRDU was omitted. This omission is due to the high computational time required by the algorithm. Options exercised in running WSTA2 were similar to that for WSTA1, as shown in Table 6.13, with the following exceptions:

- fleet size is set to 16
- average speed is set to 15 miles/hour
- maximum ride time is set to 90 minutes

Customer disutility was found to be zero in all solutions. The best cost found with varying weight sets are shown in bold in the table. For WSTA2, all three heuristics generated solutions with lower total travel time and fewer vehicles as compared to the commercial solution. The computational time incurred when the IRAU routing technique was used is reasonable but the time incurred when the IRDU routing technique was used is not. This result seems to contradict the previous result shown in Section 8.1 where TSIRDU performed better than the other heuristics. This is attributed to the nature of the DARP instance. In the case of U150, the trip length of each od-pair is comparable to the length of the time window while in the case of WSTA data, most trip lengths are small with respect to the wider time window of 20 minutes considered by the agency. The smaller trip length of requests is also noted by Baugh

Table 8.1: Non-dominated solutions from tradeoff curves shown in Figure 8.2

TT	ERT
359.88	1510.04
486.28	1397.6
566.11	1359.45
593.89	1315.77
764.72	1270.22
820.35	1236.40
878.30	1222.94
1061.71	1162.02
1128.17	1146.25
1338.92	1130.71
1721.98	1123.72
1837.59	1104.42
1884.98	1088.54
2334.18	1079.08

Table 8.2: Summary of WSTA2 request database

Number of requests	369
Number of available vehicles	16
Number of DPRs	158
Number of DDRs	211
Minimum trip length (minutes)	0.15
Maximum trip length (minutes)	61.81
Average trip length (minutes)	12.13

et al. [2] for WSTA1. Generally in the case of the IRAU routing technique, since only feasible insertions are considered, narrower trip lengths with wide time windows increase the potential for feasible insertions. This characteristic may explain the better performance of Tabu Search when the IRAU routing technique is adopted. To investigate the validity of this argument, we ran the four heuristics ten times each on a modified U150 data set. The modified U150 instance differs from the original U150 by the length of the time window. In U150, time windows were set to 10 minutes. In the modified U150, time windows are set to 60 minutes. The parameter settings were as in Tables 6.2 and 7.2. The same weight sets as U150, shown in Table 6.7, was used. Table 8.4 reports the best solution found which is highlighted in bold. With the current parameter settings, TSIRAU generated the best solution. The result seems to provide some validation of our argument. It can also be observed that SSIRAU and SSIRDU generated better solutions than TSIRDU. This observation warrants further investigation – specifically,

Table 8.3: Comparison of the four heuristics on the WSTA1 and WSTA2 for varying weights. Here, η represents the weight corresponding to the total travel time objective while ψ represents that for the excess ride time objective

		Weights									CPU Time
		$\eta = 16\ \psi = 1$			$\eta = 4\ \ \psi = 1$			$\eta = 1\ \ \psi = 2$			
Data	Heuristic	Vehicles	TT	ERT	Vehicles	TT	ERT	Vehicles	TT	ERT	(min)
WSTA1	TSIRAU	8	1076.16	1596.34	7	1180.01	618.57	8	**1368.84**	**80.94**	136.21
	TSIRDU	6	1131.31	1838.89	6	**1146.30**	**495.84**	6	1420.85	106.69	659.00
	SSIRAU	8	1108.48	1959.26	8	1160.70	628.47	9	1360.04	101.47	124.80
	SSIRDU	8	**1091.53**	**1138.77**	9	1262.10	341.55	7	1407.10	82.47	1193.35
WSTA2[a]	TSIRAU	11	**2800.56**	**2931.72**	11	**3106.78**	**1285.82**	10	**3490.03**	**540.97**	101.61
	TSIRDU	11	2921.36	3054.18	11	3052.16	2049.18	11	3485.47	724.26	292.41
	SSIRAU	11	3008.14	2638.37	10	2941.34	1960.86	12	3376.52	764.58	156.83

[a]The commercial solution has a total travel time of 5497.33, 16 vehicles used, and disutility of 1384.84.

an investigation of the performance of metaheuristics with respect to varying time window specifications. In the next section, the four metaheuristics are compared in terms of their performance on new 150-request and 250-request data sets.

Table 8.4: Results obtained by applying TSIRAU, TSIRDU, SSIRAU and SSIRDU to the modified U150

Algorithm	Vehicles	Run's Best Cost			Average[1]	Standard Deviation[2]	Average CPU[3]
		TT	ERT	Cost			
TSIRAU	9	**901.05**	**921.84**	11755.23	1.72	1.28	13.48
TSIRDU	9	929.35	966.82	12156.49	2.87	1.82	12.08
SSIRAU	9	922.06	864.78	11872.22	1.25	0.90	89.89
SSIRDU	9	925.00	829.32	11833.64	1.65	1.57	46.23

[1] - percentage of average deviation from best solution found in the run
[2] - standard deviation in percentage of average deviation from best solution found in the run
[3] - CPU time (in minutes) on a Athlon XP 2600+

8.3 Comparison using 150-request and 250-request DARP instances with varying time window duration

8.3.1 150-request DARP instances

To further compare the four metaheuristics, three DARP data sets were constructed via the random DARP instance generator described in Appendix A.1. The characteristics of the three instances are summarized in Table 8.5.

Using the parameter settings in Tables 6.2 and 7.2, each metaheuristic was run ten times, each time starting from an arbitrary solution. Table 8.6 reports the results obtained. The best solution found for each data set is in bold. The results support the argument that when the time window is large as compared to the average trip length, with the current parameter settings, TSIRDU does not necessarily generate better solutions than the other metaheuristics. In fact, for U150C, TSIRAU outperforms the other metaheuristics by producing the best solution.

8.3.2 250-request DARP instance

In the previous section, the metaheuristics were compared using 150-request DARP instances. In this section, we continue to explore the effect of time window duration on the performance of the metaheuristics. A 250-request DARP instance, or U250, was created via the DARP instance generator discussed in Appendix A.1. The characteristics of U250 are summarized in Table 8.7. Using the same parameter settings as discussed in Section 8.1, the four metaheuristics were applied to U250. Figure 8.3 shows the plot of percentage above best known solution versus

Table 8.5: Characteristics of U150A, U150B, and U150C

DARP instance	U150A	U150B	U150C
Service area (miles^2)	16	16	16
Number of requests n	150	150	150
Type of request	randomly generated	randomly generated	randomly generated
Time window (minutes)	**10**	**20**	**30**
Loading time (minutes)	5	5	5
Number of vehicles m	10	10	10
Vehicle average speed (miles/hour)	20	20	20
Number of ordinary seats	4	4	4
Number of wheelchair places	2	2	2
Average wait time (minutes)	30	30	30
Maximum ride time (minutes)	60	60	60
Maximum route duration (minutes)	480	480	480
Minimum trip length (minutes)	0.12	0.07	0.47
Maximum trip length (minutes)	14.28	13.81	14.28
Average trip length (minutes)	6.58	5.99	6.75
Number of DPRs	73	67	76

CPU time. The diamond-shaped symbol on each curve represents the solution if the algorithms were terminated via the experimental stopping criterion. Next, we increased the time window duration of U250 from 10 minutes to 60 minutes. We repeated the experiment and Figure 8.4 shows the plot of percentage above best known solution versus CPU time obtained.

Referring to Figures 8.3 and 8.4, observe that TSIRDU outperformed the other metaheuristics when the time window duration is set at 10 minutes and time is not a deciding factor. However, TSIRAU outperforms the other metaheuristics when the time window durations is increased from 10 to 60 minutes. This further validates that the duration of time windows with respect to the average trip length should be taken into consideration when solving the DARP. For example, if the duration of time windows is set at 60 minutes for U250 and the experimental stopping criterion is adopted, TSIRAU will be the candidate metaheuristic for solving the DARP since it produces a high quality solution in the least amount of CPU time.

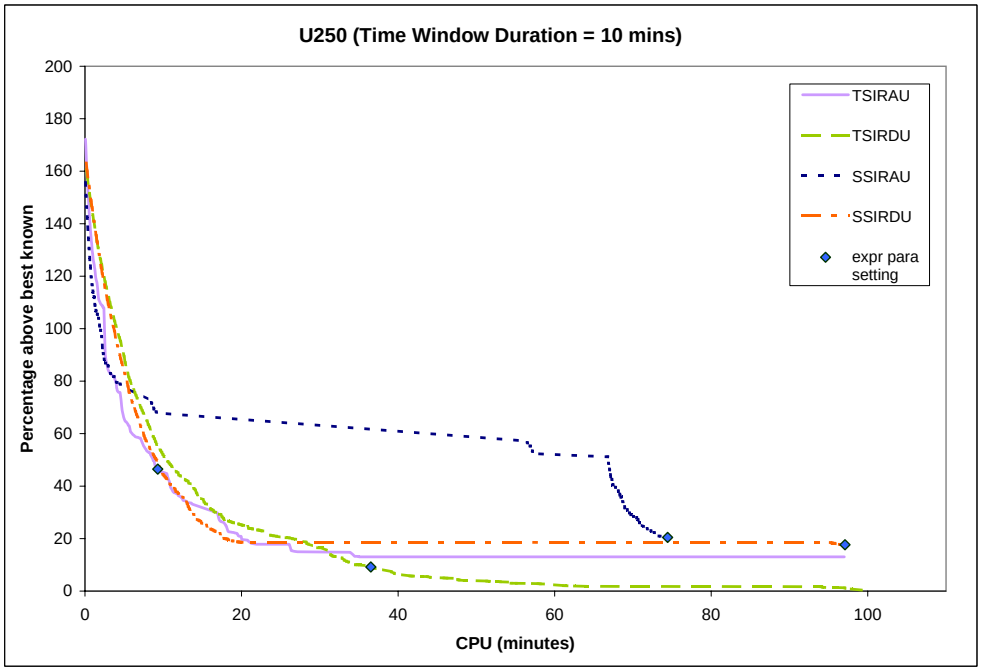


Figure 8.3: Comparison of TSIRAU, TSIRDU, SSIRAU, and SSIRDU on U250 (with 10-minute time window duration)

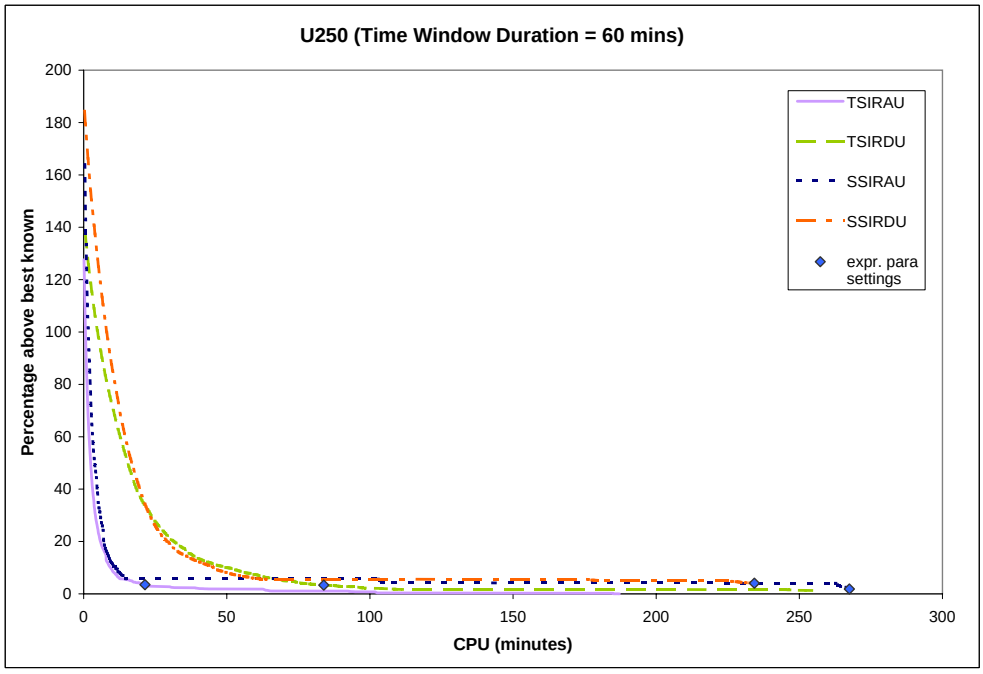


Figure 8.4: Comparison of TSIRAU, TSIRDU, SSIRAU, and SSIRDU on U250 (with 60-minute time window duration)

Table 8.6: Results obtained by applying TSIRAU, TSIRDU, SSIRAU and SSIRDU to U150A, U150B, and U150C

Algorithm	Vehicles	Run's Best Cost[1]			Average[2]	Standard Deviation[3]	Average CPU[4]
		TT	ERT	Cost			
U150A data							
TSIRAU	9	1520.83	505.15	2025.98	3.47	3.16	4.30
TSIRDU	9	**1437.14**	**366.94**	**1804.08**	0.16	0.22	13.67
SSIRAU	9	1502.66	536.07	2038.73	4.11	5.46	28.38
SSIRDU	9	1487.98	605.82	2093.80	4.69	4.22	11.00
U150B data							
TSIRAU	9	1324.06	75.76	1399.82	2.91	3.59	5.11
TSIRDU	9	**1320.31**	**23.01**	**1343.32**	1.65	2.33	13.16
SSIRAU	9	1388.65	24.53	1413.18	2.48	3.02	65.61
SSIRDU	9	1407.00	72.61	1479.61	2.66	2.32	22.69
U150C data							
TSIRAU	9	**1358.28**	**56.26**	**1414.54**	0.49	0.52	7.48
TSIRDU	9	1389.36	30.28	1419.64	1.02	1.44	12.43
SSIRAU	9	1394.46	31.28	1425.74	0.43	0.53	40.91
SSIRDU	9	1402.20	30.99	1433.19	0.99	0.89	28.37

[1] - $COST = TT + ERT$
[2] - percentage of average deviation from best solution found in the run
[3] - standard deviation in percentage of average deviation from best solution found in the run
[4] - CPU time (in minutes) on a Athlon XP 2600+

Table 8.7: Characteristics of U250

DARP instance	U250
Service area (miles^2)	16
Number of requests n	250
Type of request	randomly
Time window (minutes)	10
Loading time (minutes)	5
Number of vehicles m	13
Vehicle average speed (miles/hour)	20
Number of ordinary seats	4
Number of wheelchair places	2
Average wait time (minutes)	30
Maximum ride time (minutes)	60
Maximum route duration (minutes)	480
Minimum trip length (minutes)	0.24
Maximum trip length (minutes)	13.00
Average trip length (minutes)	6.50
Number of DPRs	126

Chapter 9

Schedule Analysis

Given a weighted multi-objective function, the value of the final objective function provides little information regarding the nature of the schedule produced. For example, refer to Table 9.1. Observe that all three solutions may have identical cost values but the nature of the schedules is vastly different. Clearly, more emphasis on customer satisfaction is evident in solution number 1 as compared to solution numbers 2 or 3. By the same token, more emphasis is placed on reducing system cost in solution number 3 as compared to solution number 1.

Table 9.1: An example of three solutions that share the same cost value but depict different policy considerations.

Sol. No.	TT	ERT	COST = TT + ERT
1	200	50	250
2	125	125	250
3	100	150	250

This simple example illustrates that the objective function value alone is inadequate in capturing the nature of the generated schedule. One way of overcoming this limitation is to present the solution in terms of certain relevant statistics as demonstrated by Jaw et al. [28] and Madsen et al. [33]. However, the saying "a picture is worth a thousand words" suggests that a visual display of the solution is another means of characterizing the nature of a DARP solution.

In the following sections, we will introduce the relevant statistics and demonstrate how they can be used to analyze a schedule. Following that, we will introduce a graphical user interface that we developed which allows the user to display a DARP solution in several views.

9.1 Statistical Display

As demonstrated by Jaw et al. [28] and Madsen et al. [33], the following statistics can be used to better characterize a DARP solution:

ATT - average travel time; travel time measures the time taken to travel from one stop to another stop, it does not include loading and waiting times. Hence, ATT measures the average travel time per request.

AMCU - average maximum capacity utilization; the maximum capacity utilization is computed as a ratio by taking the maximum concurrent riders found in each vehicle over the maximum capacity of the vehicle. Hence, AMCU represents maximum capacity utilized on average among all vehicles used.

ARTR - average ride time ratio; ride time ratio represents the actual ride time of a customer over his direct ride time. The direct ride time of a customer refers to the travel time between a customer's origin and destination stop while the actual ride time of a customer refers to the time a customer arrives at his destination stop minus the time he leaves his origin stop. In a schedule, if a customer is transported from his origin to destination stop directly, his actual ride time equals his direct ride time. In this case, the ride time ratio is 1. ARTR is computed as:

$$\text{ARTR} = \frac{1}{n}\sum_{i=1}^{n} (\text{actual ride time})_i/(\text{direct ride time})_i$$

AST - average slack time; total slack time refers to the inactive time that a vehicle spends waiting at a stop to pickup a customer while the vehicle is empty. AST measures the average inactive time of each vehicle.

MWT - maximum wait time; waiting is allowed only at origin stops, refer to Section 1.3 for our model constraints specification. In this case, wait time refers to the time a vehicle carrying passengers spends at a stop while waiting to pickup customers. Hence, MWT gives the maximum time customers may have to wait at an (origin) stop.

NUR - number of unassigned requests; only in the TSIRAU algorithm can requests be left unassigned.

PTWMOC - percentage of time with more than one customer; over the time where at least one customer is serviced, PTWMOC denotes the percentage of time where the vehicle is servicing more than one customer.

VP - vehicle productivity; VP gives the number of passengers serviced per hour. It is computed as:

$$\text{VP} = 60n/(\text{total travel time})$$

To illustrate how the schedule generated differs when different weight sets are considered, we investigate three different schedules:

Schedule A customer satisfaction weights are higher than system cost weights.

Schedule B customer satisfaction weights are approximately equal to system cost weights.

Schedule C system cost weights are higher than customer satisfaction weights.

From the tradeoff curves generated for U30 and U150 when applied to the TSIRAU and TSIRDU algorithms (as shown in Section 6.7.3) for tradeoff curves, respective weight sets reflecting the three schedules were selected. If $\lambda_{\text{schedule}} = (\eta/\psi)_{\text{schedule}}$, then

$$\lambda_A > \lambda_B > \lambda_C$$

where η and ψ are the associated weights corresponding to the objective TT and ERT respectively. The result is shown in Table 9.2.

Several observations can be made from the results reported in Table 9.2:

1. No requests were left unassigned when using TSIRAU.
2. ATT is a decreasing function of λ. This is consistent with the emphasis placed on the weight sets since a larger λ value implies that more emphasis is placed on the minimization of total travel time, hence, we expect the average travel time to decrease with increasing λ.
3. $ARTR$ is an increasing function of λ. Again, this is consistent with the emphasis placed on the weight sets since a larger λ implies that less emphasis is placed on the minimization of the cumulative excess ride time. Hence, we expect the $ARTR$ to increase with increasing λ.
4. $ARTR$ is proportional to $PTWMOC$. As the average ride time of each passenger increases, the percentage of time that more than one passengers travel together increases also.

Table 9.2: Analysis of schedules generated for the three policies: A, customer satisfaction is valued more than system costs; B, customer satisfaction and system costs are valued equally; and C, customer satisfaction is valued less than system cost, when applying the TSIRAU and TSIRDU algorithms to the DARP instances U30 and U150.

Instance	Policy	NUR	AMCU	ARTR	MWT[1]	ATT[2]	VP[3]	AVS[4]	PTWMOC[5]
					TSIRAU				
U30	A	0	0.33	1.46	0.00	11.24	2.70	9.70	18.24
	B	0	0.39	2.14	12.27	8.63	2.68	33.41	44.86
	C	0	0.67	3.29	4.01	8.30	2.69	39.63	51.85
U150	A	0	0.23	1.17	0.00	10.15	2.72	29.12	3.11
	B	0	0.42	2.24	19.43	7.71	2.78	59.03	20.37
	C	0	0.57	3.85	25.62	7.08	2.77	44.74	39.85
					TSIRDU				
U30	A	-	0.39	1.37	0.00	10.52	2.74	14.15	17.62
	B	-	0.44	2.16	0.00	8.53	2.72	35.25	46.64
	C	-	0.44	3.72	16.03	7.80	2.61	46.40	60.70
U150	A	-	0.23	1.14	0.00	9.80	2.69	37.48	4.31
	B	-	0.40	1.95	15.45	7.25	2.76	64.48	23.51
	C	-	0.48	2.82	5.04	6.82	2.85	56.04	34.51

(1 = in minutes; 2 = in minutes/job; 3 = jobs/hour; 4 = in minutes/vehicle; 5 = in percents.)

9.2 Graphical Display

The purpose of the graphical user interface is to provide the user with the ability to visually examine each DARP solution. The GUI is implemented using Java Swing and bar charts are implemented using JFreeChart. The GUI provides the following viewing options:

- client, location, request, vehicle, and stop tables in table format.
- statistics as discussed in Section 9.1 in the form of bar charts.
- individual vehicle route and spatial distribution.

Figure 9.1 shows the route and spatial distribution for vehicle V01 of a particular DARP solution. The route view, in the top left corner, shows the sequence of stops to be serviced and provides temporal information. The schedule of vehicle V01 is displayed in the form of a request-time graph. The vertical axis marks the list of requests serviced by a particular order while the horizontal axis marks the arrival time and departure time of the vehicle. Light rectangles represent the time windows at pickup stops while dark rectangles represent the time windows at dropoff stops. Vehicle arrivals are represented by light dots while vehicle departures

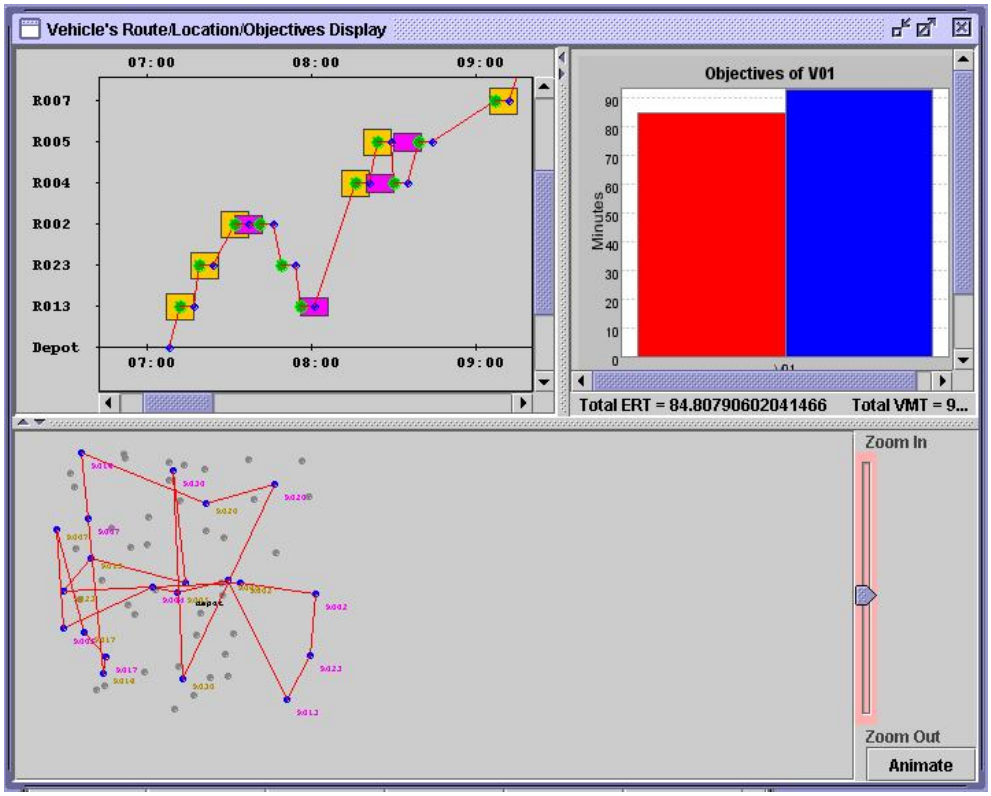


Figure 9.1: Graphical view of route, spatial distribution and objective of vehicle V01.

are represented by dark dots. Observe that request R013 has two time windows while request R023 has only one time window. This provides a visual cue that request R013 is a DDR while request R023 is a DPR. Also observe that request R002 has overlapping time windows and the vehicle arrives at the destination near the end of the dropoff time window. It is also intuitively obvious from the graph that the vehicle picks up request R013, request R023, and request R002 before dropping off request R002, request R023, and request R013.

The spatial view, in the bottom left corner, provides a bird's eye view of the stops to be serviced and the order in which stops are visited. Dark dots represent stops in the schedule of vehicle V01 and the text identify the request associated with each stop. Light dots represent the remaining stops in this particular data set. When the "Animate" button is clicked, each edge in the graph is highlighted in the sequence to be visited by the vehicle. The scroll bar, in the bottom right corner, allows the user to resize this view. Observe that the locations serviced by vehicle V01 are not concentrated in one area.

The solution's corresponding objectives are displayed in the form of a bar chart as seen in the top right corner. In this particular case, total ERT and total TT are displayed.

9.3 Summary

In the above discussion, we have shown alternate ways in which a user can analyze a DARP solution. A DARP solution can be characterized in terms of relevant statistical information or can be examined visually via a graphical user interface. Both displays provide useful information that will allow a user to better characterize the nature of a DARP schedule.

Chapter 10

Conclusions and Future Research

In this thesis, the many-to-many, advanced reservation, multi-vehicle, single-depot, static DARP was examined. The dual objectives considered are to minimize total travel time and total excess ride time. This is subject to round trip, coupling, precedence, time windows, vehicle capacity, ride time, route duration, and wait time constraints. While the problem can be solved exactly, current methods are too slow for most practical applications. We have instead focused our research in developing methods that can generate high quality solutions within reasonable computing time.

By adopting the cluster-first route-second approach, various combinations of clustering and routing techniques were considered for the exploration of (particularly) two solution approaches. They are (1) to disallow constraints violation and leave a request unassigned if necessary during searches, and (2) to allow constraints violation and no requests can be left unassigned during searches. Refer to Chapter 4 for discussion. As a result, two routing techniques were developed to supplement these approaches. They are IRAU (for *Approach I*) and IRDU (for *Approach II*). Details of these techniques are given in Chapter 5. Clustering is performed either via Tabu Search, refer to Chapter 6 for details, or Scatter Search as described in Chapter 7. The four metaheuristics considered are TSIRAU, TSIRDU, SSIRAU, and SSIRDU.

The four metaheuristics use the concept of a neighborhood, or a candidate list of moves, to search the solution space; a neighborhood is defined as a set of solutions which are a small perturbation from the current solution. Each move is evaluated and the candidate move is selected to replace the current solution. Three moves were considered. They are single request move (SRM), paired requests swap (PRS), and stops swap move (SSM). Detailed discussion of each move can be found in Section 6.3. For the search to proceed quickly, the consideration of moves in combination with routing technique is a key factor. IRAU is, in general, a faster routing technique as compared to IRDU since only feasible insertions are considered during

search. As a result, the choice of neighborhood is largely dependent on the routing technique employed. It was found that TSIRDU and SSIRDU do not benefit from the inclusion of SSM; observe that both metaheuristics adopt the IRDU routing technique. In addition, it was also observed that some benefits can be gained by including PRS in TSIRDU and SSIRDU, but IRDU paired with PRS was found to be computationally expensive. As a result, PRS was excluded from SSIRDU since Scatter Search considers a reference set of (at least, two or more) solutions which requires computational efficiency in the evaluation of neighborhoods.

In Chapter 8, comparisons of the metaheuristics over test problems and two real-world problems were presented. Tradeoff curves were generated for the U150 data set. TSIRDU outperformed the other heuristics, evidenced by the non-dominated solutions it generated. In addition, TSIRDU produced the best cost on the U150A and U150B data sets. However, TSIRAU generated better solution on U150C, WSTA1 for weight sets $\eta = 1$ and $\psi = 2$, and WSTA2 for all weight sets considered in terms of the objectives considered in this research. It was found that while considering only feasible solutions during search, as in the case with TSIRAU, limits the exploration of solutions if time windows are large as compared to the average trip length of requests. Such a limitation appears to have little effect on the exploration of solutions. While the SSIRAU and SSIRDU algorithms did not generate better solutions (in the long run) as compared to the TSIRAU and TSIRDU algorithms, better solutions were found earlier in the run which suggests that there is benefit in considering multiple solutions.

The objective function value of a solution may not be the best measure of its quality especially when dealing with a multi-objective function. Operating statistics, as discussed in Chapter 9, provide means of characterizing the nature of a DARP. When utilized with a graphical user interface, where spatial and temporal distributions, and route schedule are represented visually, the modeler will be equipped with resources to better analyze schedules.

Future Research

- The current model formulation does not consider all issues like driver's preference for routes, customer's driver preference, customers who do not want to travel with other customers etc. More sophisticated models can be considered.

- The specification of an objective function guides the search, while the weight associated with each objective remains unchanged during search. An interesting future research work is to examine dynamic weights.

- The global performance of SS, along with the local performance of TS, suggests a combination of the two metaheuristics. One possible combination is to replace the improvement

method in SS with TS. However, by doing so, one could be faced with high computational cost. Since SS is inherently parallel, in contrast to TS, this high computational cost can be reduced if processors are run in parallel and multiple solutions can be improved at any point in time.

- Another interesting idea for future research is a GUI expert system that incorporates statistical information about a problem and preferences of the modeler to produce solutions by adopting the metaheuristics presented here. Such a system should require minimal human guidance, be intuitive, and user friendly.

Bibliography

[1] Alfa, A. S. 1986. Scheduling of vehicles for transportation of elderly. *Transportation Planning and Tech.* 11(3):203–212.

[2] Baugh, J.W. Jr., G.K.R. Kakivaya, J.R. Stone. 1998. Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. *Engineering Optimization* 30:91–123.

[3] Bodin, L., B. Golden, A. Assad, M. Ball. 1983. Routing and Scheduling of vehicles and crews. The state of art. *Computers of Operations Research* 10:69-211.

[4] Chapleau, L., J. Ferland, J. M. Rousseau. 1981. *Clustering for Routing in dense area.* University of Montreal Transportation Research Center Publication. No. (206).

[5] Collette, Y., P. Siarry. 2003. *Multiobjective Optimization.* Springer Verlag. Berlin, Germany.

[6] Cordeau, J.-F., L. Gilbert. 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Center Part B: Methodological (In-press).*

[7] Cordeau, J.-F., G. Larporte. 2003. The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French, and Italian Operations Research Societies* 1:89-101

[8] Daganzo, C. 1978. Approximate analytic model of many-to-many demand responsive transportation systems. *Transportation Research* 12:325-333.

[9] Desaulniers, G., J. Desrosiers, A. Erdmann, and M.M. Solomon, and F. Soumis. 2002. The VRP with Pickup and Delivery. In *The vehicle routing problem, SIAM Monographs on Discrete Mathematics and Applications* 9:157–193.

[10] Dumas, Y., F. Soumis, J. Desrosiers. 1991. The pickup and delivery problem with time windows. *European Journal Operational Research* 54:7–22.

[11] Desrosiers, J., Y. Dumas, M.M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. *In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing, Handbooks in Operations Research and Management Science*, Elsevier Science, Amsterdam, 8:35–139.

[12] Dumas Y., J. Desrosiers, F. Soumis. 1989a. Large scale multi-vehicle dial-a-ride problems. *Les Cahiers du GERAD*. G–89–30. École des Hautes Études Commerciales, Montrél.

[13] Dumas Y., F. Soumis, J. Desrosiers. 1989b. Optimizing the schedule for a fixed vehicle path with convex inconvenience cost. *Les Cahiers du GERAD*. G–89–08. École des Hautes Études Commerciales, Montrél.

[14] Fu, L. 2002. Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. 2002. *Transportaion Research Part B* 36:485-506.

[15] Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability, A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, CA.

[16] Gendreau, M., G. Laporte, D. Vigo. 1998. Heuristics for the travelling salesman problem with pickup and delivery. CRT-98-10. Centre de Recherche sur les Transports, Université de Montréal.

[17] Gillet, B., J. Johnson. 1976. Multi-terminal vehicle dispatch algorithm. *Omega* 4:711-718.

[18] Glover, F. 1989. Tabu Search - Part I. *ORSA Journal of Computing* 1(3):190-206.

[19] Glover, F. 1990. Tabu Search - Part II. *ORSA Journal of Computing* 2(1):4–32.

[20] Glover, F., M. Laguna, R. Marti. 2000. Fundamentals of Scatter Search and Path Relinking. *Control and Cyberbetics* 39(3):653–684.

[21] Glover, F., M. Laguna, R. Marti. 2002. "Scatter Search", in *Theory and Applications of Evolution Computation : Recent Trends*. A. Ghosh and S. Tsutsui, Springer-Verlag. 519–529.

[22] Glover, F., M. Laguna, R. Marti. *New Ideas and Applications of Scatter Search and Path Relinking*. To appear in New Optimization Techniques in Engineering, G. Onwubolu (Ed.), Springer-Verlag.

[23] Glover, F.. 1990. Tabu Search: A tutorial. *Interfaces* 20:74–95.

[24] Grefenstette, J. J. 1986. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man and Cyberbetics* 16(1):122-128.

[25] Healy, P., R. Moll. 1995. Theory and Methodology: A new extension of local search applied to the dial-a-ride problem. *European Journal of Operational Research* 83:83–104.

[26] Hansen P., B. Jaumard. 1990. Algorithms for the maximum satisfiability problem. *Computing* 44:279–303.

[27] Ioachim, I., J. Desrosiers, Y. Dumas, M. M. Solomon, D. Villeneuve. 1995. A request clustering algorithm for door-to-door handicapped transportation. *Transportation Science* 31(1):60–71.

[28] Jaw, J.-J., A. R. Odoni, H. N. Psaraftis, N.H.M. Wilson. 1986. A heuristic algorithm for

the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research* 20B(3):243–257.

[29] Kikuchi, S., J.-H. Rhee. 1989. Scheduling method for demand-responsive transportation system. *Journal of Transportation Engineering* 115(6):630-645.

[30] Lawler, E. L. 1976. *Combinatorial Optimization: Networks and Matroids* New York: Holt, Rinehard and Winston.

[31] Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys. 1985. *The traveling salesman problem* Great Britain: Wiley-Interscience.

[32] Nanry, W. P., J. W. Barnes. 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research B* 34:107-121.

[33] Madsen, O.B.G., H.F. Ravn, J.M. Rygaard. 1995. A heuristic algorithm for the dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research* 60:193–208.

[34] Patrick, H., R. Moll. 1993. A new extension of local search applied to the Dial-A-Ride Problem. *European Journal of Operational Research* 83:83–104.

[35] Ponterosso, P., D. S. J. Fox. 1999. Heuristically Seeded Genetic Algorithms Applied to Truss Optimization. *Engineering with Computers* 15:345–355.

[36] Psaraftis, H. N. 1980. A dynamic programming solution to the single vehicle, many-to-many, immediate request dial-a-ride problem. *Transportation Science* 14(2):130–154.

[37] Psaraftis, H. N. 1983. An exact algorithm for the single vehicle, many-to-many dial-a-ride problem with time windows. *Transportation Science* 17(3): 351–357.

[38] Psaraftis, H. N. 1983. k-Interchange procedures for local search in a precedence constrained routing problem. *European Journal of Operational Research* 13:391–402.

[39] Rardin, R., R. Uzsoy. 2001. Experimental Evaluation of Heuristic Optimization Algorithms: A tutorial. *Journal of Heuristics* 7:261–304.

[40] Roy, S., J.-M. Rousseau, G. Lapalme, J. A. Ferland. 1984. Routing and scheduling for the transportation of disabled persons - The algorithm. Technical Report TP 5596E. Montreal: Centre de Recherche sur les Transports.

[41] Roy, S., J.-M. Rousseau, G. Lapalme, J. A. Ferland. 1984. Routing and scheduling for the transportation of disabled persons - The tests. Technical Report TP 5598E. Montreal: Centre de Recherche sur les Transports.

[42] Ruland, K. S., E. Y. Rodin. 1997. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers and Mathematics with Applications* 33(12):1–13.

[43] Savelsbergh, M.W.P.. 1992. The vehicle routing problem with time windows: Minimizing

route duration. *ORSA Journal of Computing* 4:146–154.

[44] Schaffer, J. D., R. A. Caruana, L. J. Eshelman, R. Das. 1989. A study of control parameters affecting online performance of genetic algorithms for function optimization. *in Schaffer* 51–60.

[45] Sexton, T., L. Bodin. 1985a. Optimizing single vehicle many-to-many operation with desired delivery times: I Scheduling. *Transportation Science* 19:378–410.

[46] Sexton, T., L. Bodin. 1985b. Optimizing single vehicle many-to-many operation with desired delivery times: II Routing. *Transportation Science* 19:411–435.

[47] Stein, D. M. 1978. Scheduling dial-a-ride transportation systems. *Transportation Science* 12:232-249.

[48] Toth, P., D. Vigo. 1997. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science* 31(1):60–71.

[49] Van der Bruggen, L. J. J., J. K. Lenstra, P. C. Schuur. 1993. Variable-depth search for the single-vehicle pickup and delivery problem with time windows. *Transportation Science* 27:298-311.

[50] Wilson, N. H. M., J. M. Sussman, H. K. Wong, T. Higonnet. 1971. Scheduling algorithms for a dial-a-ride system. USL TR-70-13. Boston: M.I.T. Urban Systems Laboratory.

[51] Wilson, N. H. M., H. Weissberg. 1976. Advanced dial-a-ride algorithms research project: Final Report. R76-20. Boston: M.I.T. Department of Civil Engineering.

[52] Wilson, N. H. M., N. H. Colvin. 1977. Computer control of the Rochester dial-a-ride system. R77-31. Boston: M.I.T. Department of Civil Engineering.

[53] Wolpert, D. H., W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1):67–82.

Appendix A

Multi-Vehicle DARP Instance Construction

In this section, we present two methods of generating DARP instances. The first approach generates a DARP instance by construction such that the optimal solution can be computed in advance. The second approach generates a DARP instance randomly but guarantees that all requests in that instance can be serviced by the predetermined fleet.

A.1 m-vehicle DARP instance construction

In this section, we will describe how one can construct a DARP instance whose optimal solution containing m (for $m \leq 4$) tours or vehicles is known. To do so, we will first consider the case where $m = 1$. The depot is located at the origin $(0, 0)$. Assume n (n, even) desired pickup requests (or DPR), where each request is indexed as i and requires zero loading time, are serviced by the tour. Since each request is associated with an od-pair, we will index the origin and destination stop of request i as i and $i + n$ respectively. The $2n$ locations visited by this tour will be generated, evenly spaced, on a circle with radius r, centered at $(3r, 0)$. Hence, the tour sequence is given by

$$< 0, 1, 2, 1 + n, 2 + n, 3, 4, \ldots, n - 1, n, 2n - 1, 2n, 0 >$$

where 0 represents the depot and stop 1 is located at $(2r, 0)$. The request time for request i, rt_i, can be generated as follows:

$$
\begin{aligned}
rt_1 &= 2r \\
rt_2 &= 2r + 2r\sin(\alpha/2) \\
rt_i &= rt_1 + 4(i-1)\sin(\alpha/2) \quad i \text{ even} \\
rt_i &= rt_2 + 4(i-1)\sin(\alpha/2) \quad i \text{ odd}
\end{aligned}
$$

The objective is to minimize the cost function which is expressed a linear combination of cumulative travel time, TT, and cumulative excess ride time, ERT. It is given as:

$$COST = n * TT + ERT \tag{A.1}$$

Theorem A.1 *The tour constructed is optimal for cost function A.1 and the optimal solution value is given as*

$$n\,(2r + (2n-1)a + c) + n(2a - b)$$

where $a = 2r\sin(\alpha/2)$, $b = 2r\sin(\alpha)$ *and* $c = 2r(1 + \sin(\alpha/2)^2)$. *Here,* $\alpha = \pi/n$. *Refer to Figure A.1 for a schematic representation of a, b and c.*

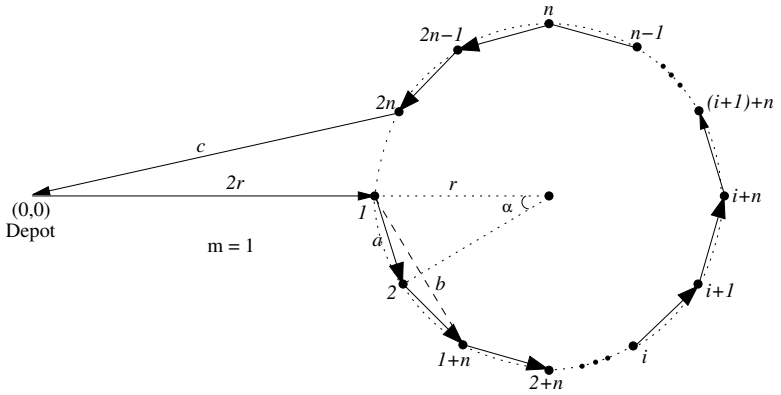


Figure A.1: Optimal route for 1-vehicle case.

We will show that the tour constructed is optimal for the DARP instance with the cost function at A.1. Observe that if the objective is to minimize cumulative excess ride time, then a tour that visits each destination stop immediately after its origin stop will result in a tour with zero excess ride time. By the same token, if the objective is to minimize cumulative travel time, it is clear that a tour that visits each stop on the circle in sequence, starting from stop i, is optimal in terms of minimizing total travel time. With the given objective, based on construction, it suffices to show that the sequence $A =< i, i+1, i+n, (i+1)+n >$ results in a cost that is lower than the sequence $B =< i, i+n, i+1, i+1+n >$ for $1 \leq i \leq n-1$.
Referring to the Figure A.1, the cost for tour A, expressed in terms of a and b, is given as

$$\begin{aligned} TT &= 3a \\ ERT &= 2(2a-b) = 4a-2b \\ COST_A &= n(3a)+4a-2b \end{aligned}$$

By the same token, the cost for tour B is given as

$$\begin{aligned} TT &= 2b+a \\ ERT &= 0 \\ COST_B &= n(2b+a) \end{aligned}$$

We want to show that $COST_A - COST_B < 0$. Observe,

$$\begin{aligned} COST_A - COST_B &= (3na+4a-2b)-(2bn+na) \\ &= 2na+4a-2b-2bn \end{aligned}$$

Since $b = 2r\sin(\alpha) = 2a\cos(\alpha/2)$, we have

$$\begin{aligned} COST_A - COST_B &= 2na+4a-(2+2n)2a\cos(\alpha/2) \\ &= 2a\left(n+2-(2+2n)\cos(\alpha/2)\right) \\ &< 0 \end{aligned}$$

for $\alpha = \Pi/n$ and $n \geq 2$, even.

Finally, in determining the sequence of stops to visit, it should be clear that between every two partial sequences of stops $u =< i, i+1, i+n, (i+1)+n >$ for $1 \leq i \leq n-3$ and $v =< i+2, i+3, (i+2)+n, (i+3)+n >$, the optimal way of linking them is $< u, v >$ as this will result in additional total travel time of na and zero excess ride time. Hence, this shows that a tour that minimizes total cost is given by $< 0, 1, 2, 1+n, 2+n, 3, 4, \ldots, 2n-1, 2n >$. Since n is even, the optimal cost is given as

$$\begin{aligned} \text{TT} &= \begin{array}{c}\text{travel time}\\ \text{from depot}\\ \text{to stop 1}\end{array} + \begin{array}{c}\text{travel time}\\ \text{partial}\\ \text{sequences}\end{array} + \begin{array}{c}\text{linking}\\ \text{partial}\\ \text{sequences}\end{array} + \begin{array}{c}\text{travel time}\\ \text{from stop 2n}\\ \text{to depot}\end{array} \\ &= 2r + n/2(3a) + (n/2-1)a + c \end{aligned}$$

$$\begin{aligned} \text{ERT} &= \text{request 1 to } n \\ &= n/2(4a-2b) \end{aligned}$$

Hence, the cost is given as:

$$\begin{array}{rcccc} \text{COST} & = & n * TT & + & ERT \\ & = & n(2r + 3an/2 + (n/2 - 1)a + c) & + & (n/2)(4a - 2b) \\ & = & n(2r + (2n - 1)a + c) & + & n(2a - b). \end{array}$$

Now, for the case where $1 < m \leq 4$, the above tour construction can be repeated by setting the center of the circular tours at $(3r, 0)$, $(-3r, 0)$ and $(0, -3r)$ for the $2nd$, $3rd$ and $4th$ vehicle respectively. Observe that the shortest travel between any two locations $s_i = (x_i, y_i)$ and $s_j = (x_j, y_j)$ where locations s_i and s_j are from different tours, is greater than $2r$ where $2r$ represents the distance from the depot to the first serviced stop on each tour. Hence, there is no benefit traveling from one stop to another stop from different tours. The optimal solution value can be generalized as

$$nm\,(2r + (2n - 1)a + c) + nm(2a - b)$$

In accordance to the DARP database specification as discussed in Section 1.4, the following information is included to complete the description of the DARP instance:

- Vehicle database
  - number of ordinary seats in vehicle k: $CO_k = m$
  - number of wheelchair places in vehicle k: $CW_k = 0$
- Client database
  - no wheelchair required for all requests
- Request database
  - Since each request is of type DPR, as discussed in Section 1.3, only one time window specification at the origin stop needs to be created for each request. Set

    $$tw = 5r$$

    Time window $[e_i, t_i]$ for request i, for $i = 1, \ldots, n$ can be computed as follows:

    $$e_i = rt_i - tw/2$$
    $$l_i = rt_i + tw/2$$
- Model

- maximum ride time; $m_i = 60$ for all i
- average speed is 60
- maximum route duration; $H = 480$

A.2 Random DARP instance construction

In this section, we will describe how to construct a DARP instance that contains a feasible solution given a desired fleet size. In other words, all constraints described in Section 1.3 must be met. The constraints are:

1. **same route**: every route starts and ends at the depot
2. **precedence**: for every request i, the od-pair $(i, i+n)$ must belong to the same route
3. **coupling**: the origin stop i must be visited before the destination stop $i+n$
4. **time windows**: the pickup time window for DPR and dropoff time window for DDR must not be violated
5. **vehicle capacity**: the maximum concurrent passengers requiring ordinary seats and wheelchair places in vehicle k at any instant must not exceed co_k (number of ordinary seats) and cw_k (wheelchair places) respectively
6. **maximum ride time**: the total time that request i is in a vehicle must not exceed m_i
7. **route duration**: the total duration of route k must not exceed h_k
8. **no waiting at destination stops**: no waiting at destination stops are allowed
9. **maximum waiting time**: the total waiting time at origin stops must not exceed w when the vehicle is carrying passengers

The instance is generated by first creating the ordering of stops in each tour followed by request-dependent parameter specification.

An area of service must first be defined. Without loss of generality, we will assume that the area of service is defined as $l \times l$ (in square miles). Let n represent the number of requests in the DARP and m the fleet size. Within the service area $l \times l$, randomly generate a depot stop followed by $2n$ distinct service locations for the n origin and n destination stops required. Let $c_k = \lfloor co_k + cw_k \rfloor / 2$ where co_k and cw_k represent the specified number of ordinary seats and wheelchair places in vehicle k respectively. Without loss of generality, we assume that

$co_k \geq cw_k$. This assumption simplifies the assignment of capacity requirement for each request. Next, randomly index each stop as i where $i = 1, \ldots, 2n$ and set $nn = \lfloor n/m \rfloor$. Enumerating each tour k from 0 to $(m-1)$, we can create tour k in sequence as follows until m tours are created:

For $k = 0, \ldots, m-2$,

$$T_k =< 0, (k*nn)+1, \ldots, (k*nn)+c_k, ((k*nn)+1)+n, \ldots, ((k*nn)+c_k)+n, \\ ((k*nn)+c_k)+1, \ldots, ((k*nn)+c_k)+c_k, \ldots, (k*nn)+n, 0 >$$

and for $k = m-1$

$$T_k =< 0, (k*nn)+1, \ldots, (k*nn)+c_k, ((k*nn)+1)+n, \ldots, ((k*nn)+c_k)+n, \\ ((k*nn)+c_k)+1, \ldots, ((k*nn)+c_k)+c_k, \ldots, 2n, 0 >$$

We have just created m tours where each tour starts from the depot (indexed 0), visits each stop where the origin and destination stops of request i are indexed as i and $i+n$ respectively, and then return to the depot. Ordering requests in this manner meets the requirement of constraints (1) to (3). What remains to be determined for each request are its request time, type of capacity required and time window specification.

Only one time window is defined for each request, a time window at the origin stop for desired pickup requests (DPRs) and a time window at the destination stop for desired dropoff requests (DDRs). For each index $i < n$, randomly determine if request i is a DPR or a DDR and assign its capacity requirement type as follows:

$$\text{ordinary seat if } i \ mod \ c_k \leq co_k/2 \tag{A.2}$$

$$\text{wheelchair place if } i \ mod \ c_k > co_k/2 \tag{A.3}$$

Specifying capacity requirement in this manner satisfies constraint (5).

To simplify notation for the purpose of discussion for the remaining section, we rewrite each tour T_k as $T =< s_0, s_1, \ldots, s_q$ where stops s_0 and s_q represent depot stops. If s_i is the origin stop of a DPR or the destination stop of a DDR, the time window at this stop can be defined as follows:

$$[e_{s_i}, e_{s_i} + rand(tw_l, tw_u)]$$

where $0 < tw_l \leq tw_u$ and $rand(a, b)$ is an integer number randomly generated between a (inclusive) and b (inclusive), and

$$e_{s_i} = e_0 + \sum_{j=1}^{i} t(s_{j-1}, s_j) + \sum_{j=1}^{i-1} load_{s_j}$$

where

$$\begin{aligned}
e_0 &= \text{start time of the scheduling horizon} \\
t(s_i, s_j) &= \text{time required to get from stop } i \text{ to stop } j \text{ and } t(s_i, s_j) = d(s_i, s_j)/speed \\
d(s_i, s_j) &= \text{distance (in miles) required to get from stop } i \text{ to stop } j \\
load_{s_i} &= \text{loading time at stop } i
\end{aligned}$$

Setting the time window in the above manner ensures constraints (4), (8) and (9) to be satisfied. Finally, we need to check that the specified maximum ride time and route duration are met.

Appendix B

U30 Data Set

Table B.1: U30 Model Table

MRT FIXED 60 TW FIXED WAIT 30 SPEED 20 ROUTE 480

Table B.2: U30 Vehicle Table

ID	Capacity	Wheelchairs
V01	4	2
V02	4	2
V03	4	2

Table B.3: U30 Client Table

ID	Name	Address	Wheelchair
C001	“name1”	null	n
C002	“name2”	null	n
C003	“name3”	null	y
C004	“name4”	null	n
C005	“name5”	null	n
C006	“name6”	null	y
C007	“name7”	null	n
C008	“name8”	null	n
C009	“name9”	null	y
C010	“name10”	null	n
C011	“name11”	null	n
C012	“name12”	null	n
C013	“name13”	null	y
C014	“name14”	null	n
C015	“name15”	null	n
C016	“name16”	null	y
C017	“name17”	null	n
C018	“name18”	null	n
C019	“name19”	null	y
C020	“name20”	null	n
C021	“name21”	null	y
C022	“name22”	null	n
C023	“name23”	null	n
C024	“name24”	null	y
C025	“name25”	null	n
C026	“name26”	null	n
C027	“name27”	null	y
C028	“name28”	null	n
C029	“name29”	null	n
C030	“name30”	null	y

Table B.4: U30 Location Table

ID	Address	x-coordinate	y-coordinate
L001	“depot”	0	0
L002	“addr1”	0.538671377	0.0030497183215678625
L003	“addr2”	0.82858205	0.005913711618497075
L004	“addr3”	0.295861225	0.7976533001545705
L005	“addr4”	-0.842058054	0.5419477867826861
L006	“addr5”	1.970032959	-0.15326406837444928
L007	“addr6”	0.647380496	-1.414584214445902
L008	“addr7”	0.644549627	0.04857306530620287
L009	“addr8”	-0.135501397	-0.13123211279949887
L010	“addr9”	-1.353336878	-1.6172067678836877
L011	“addr10”	-0.511723594	-0.05804173742488494
L012	“addr11”	-1.868745815	-0.6728498739245476
L013	“addr12”	1.755203404	1.8599475000841053
L014	“addr13”	-1.974996023	0.824697371443468
L015	“addr14”	-1.269010391	0.04249577795726989
L016	“addr15”	0.125553072	-1.7077880578412112
L017	“addr16”	-1.490946478	0.9943576252514399
L018	“addr17”	0.169734523	-0.7885078796464096
L019	“addr18”	-1.230985848	-1.5627310948424893
L020	“addr19”	-1.670501027	0.5209366339224064
L021	“addr20”	-0.914553121	1.8933997552438488
L022	“addr21”	1.863682957	1.325593977506485
L023	“addr22”	0.959191801	1.873226954530855
L024	“addr23”	-1.451702247	0.3696513801558501
L025	“addr24”	-0.249462437	1.5590618152414222
L026	“addr25”	1.027554395	1.2661145357862527
L027	“addr26”	1.525631272	-1.7612511003824376
L028	“addr27”	-0.627634663	-1.3457897883850238
L029	“addr28”	0.284500026	1.7323354608219828
L030	“addr29”	-0.097402273	1.2558253275260283
L031	“addr30”	-0.87890193	-0.3322116113669158
L032	“addr31”	0.723885916	-0.7719230171019644
L033	“addr32”	0.572826106	0.7001205198328115
L034	“addr33”	-1.550663553	-0.740109955428895
L035	“addr34”	-1.250999578	-1.3684321995845812
L036	“addr35”	0.220997486	-0.46376145476054065
L037	“addr36”	-1.215219567	-1.1212605410822478
L038	“addr37”	-1.584510464	1.9944900572852635
L039	“addr38”	-0.25090487	1.8459458421485215
L040	“addr39”	0.312728921	1.2230728596838527
L041	“addr40”	1.342138821	1.502189300096516
L042	“addr41”	-0.473793385	-0.09653357030736531
L043	“addr42”	0.550039951	-0.17741584524412213
L044	“addr43”	-1.862185717	-0.1277024201357535
L045	“addr44”	-0.118664627	-1.25674556581704
L046	“addr45”	-1.75025724	1.6811056959644288
L047	“addr46”	1.891916336	-1.1079624277054996
L048	“addr47”	-0.939490997	1.957122860370485
L049	“addr48”	-0.587857926	0.5956055808696217
L050	“addr49”	0.591697111	-1.0808077919873371
L051	“addr50”	-0.778152247	-0.47310652855885893
L052	“addr51”	-0.163466929	-1.921264679359048
L053	“addr52”	-1.123992761	0.845010339020881
L054	“addr53”	1.370941585	0.45645357434216205
L055	“addr54”	-0.034161236	1.7930845295754243
L056	“addr55”	0.362266133	-1.4287637054852698
L057	“addr56”	-1.692823387	1.457780129967206
L058	“addr57”	-0.569635972	1.0085237582599897
L059	“addr58”	-1.620040081	-0.2424991658088218
L060	“addr59”	-0.039680441	-1.44883968809923
L061	“addr60”	-0.200276764	1.714578292

Table B.5: U30 Request Table

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R001	C001	L002	L005	5/27/2003	7:24	D	5	1	7:18	7:28
R002	C002	L003	L006	5/27/2003	7:38	D	5	1	7:32	7:42
R003	C003	L004	L007	5/27/2003	7:48	D	5	1	7:43	7:53
R004	C004	L008	L011	5/27/2003	8:26	D	5	1	8:20	8:30
R005	C005	L009	L012	5/27/2003	8:35	D	5	1	8:30	8:40
R006	C006	L010	L013	5/27/2003	8:54	D	5	1	8:48	8:58
R007	C007	L014	L017	5/27/2003	9:10	P	5	1	9:05	9:15
R008	C008	L015	L018	5/27/2003	9:18	P	5	1	9:13	9:23
R009	C009	L016	L019	5/27/2003	10:07	D	5	1	10:01	10:11
R010	C010	L020	L021	5/27/2003	10:18	P	5	1	10:13	10:23
R011	C011	L022	L025	5/27/2003	7:39	D	5	1	7:33	7:43
R012	C012	L023	L026	5/27/2003	7:15	P	5	1	7:10	7:20
R013	C013	L024	L027	5/27/2003	8:02	D	5	1	7:56	8:06
R014	C014	L028	L031	5/27/2003	8:13	P	5	1	8:08	8:18
R015	C015	L029	L032	5/27/2003	8:55	D	5	1	8:50	9:00
R016	C016	L030	L033	5/27/2003	8:35	P	5	1	8:29	8:39
R017	C017	L034	L037	5/27/2003	9:17	P	5	1	9:12	9:22
R018	C018	L035	L038	5/27/2003	9:24	P	5	1	9:19	9:29
R019	C019	L036	L039	5/27/2003	10:08	D	5	1	10:02	10:12
R020	C020	L040	L041	5/27/2003	10:15	P	5	1	10:10	10:20
R021	C021	L042	L045	5/27/2003	7:33	D	5	1	7:28	7:38
R022	C022	L043	L046	5/27/2003	7:48	D	5	1	7:43	7:53
R023	C023	L044	L047	5/27/2003	7:22	P	5	1	7:16	7:26
R024	C024	L048	L051	5/27/2003	8:24	P	5	1	8:19	8:29
R025	C025	L049	L052	5/27/2003	9:04	D	5	1	8:58	9:08
R026	C026	L050	L053	5/27/2003	9:18	D	5	1	9:12	9:22
R027	C027	L054	L057	5/27/2003	9:30	P	5	1	9:25	9:35
R028	C028	L055	L058	5/27/2003	10:20	D	5	1	10:15	10:25
R029	C029	L056	L059	5/27/2003	9:56	P	5	1	9:50	10:00
R030	C030	L060	L061	5/27/2003	10:55	D	5	1	10:50	11:00

Appendix C

U150 Data Set

Table C.1: U150 Model Table

MRT	FIXED	60	TW	VARY	WAIT	30	SPEED	20	ROUTE	480

Table C.2: U150 Vehicle Table

ID	Capacity	Wheelchairs
V01	4	2
V02	4	2
V03	4	2
V04	4	2
V05	4	2
V06	4	2
V07	4	2
V08	4	2
V09	4	2
V10	4	2

Table C.3: U150 Client Table

ID	Name	Address	Wheelchair	ID	Name	Address	Wheelchair
C001	“name1”	null	n	C076	“name76”	null	n
C002	“name2”	null	n	C077	“name77”	null	n
C003	“name3”	null	y	C078	“name78”	null	y
C004	“name4”	null	n	C079	“name79”	null	n
C005	“name5”	null	n	C080	“name80”	null	n
C006	“name6”	null	y	C081	“name81”	null	y
C007	“name7”	null	n	C082	“name82”	null	n
C008	“name8”	null	n	C083	“name83”	null	n
C009	“name9”	null	y	C084	“name84”	null	y
C010	“name10”	null	n	C085	“name85”	null	n
C011	“name11”	null	n	C086	“name86”	null	n
C012	“name12”	null	y	C087	“name87”	null	y
C013	“name13”	null	n	C088	“name88”	null	n
C014	“name14”	null	n	C089	“name89”	null	n
C015	“name15”	null	y	C090	“name90”	null	y
C016	“name16”	null	n	C091	“name91”	null	n
C017	“name17”	null	n	C092	“name92”	null	n
C018	“name18”	null	y	C093	“name93”	null	y
C019	“name19”	null	n	C094	“name94”	null	n
C020	“name20”	null	n	C095	“name95”	null	n
C021	“name21”	null	y	C096	“name96”	null	y
C022	“name22”	null	n	C097	“name97”	null	n
C023	“name23”	null	n	C098	“name98”	null	n
C024	“name24”	null	y	C099	“name99”	null	y
C025	“name25”	null	n	C100	“name100”	null	n
C026	“name26”	null	n	C101	“name101”	null	n
C027	“name27”	null	y	C102	“name102”	null	y
C028	“name28”	null	n	C103	“name103”	null	n
C029	“name29”	null	n	C104	“name104”	null	n
C030	“name30”	null	y	C105	“name105”	null	y
C031	“name31”	null	n	C106	“name106”	null	n
C032	“name32”	null	n	C107	“name107”	null	n
C033	“name33”	null	y	C108	“name108”	null	y
C034	“name34”	null	n	C109	“name109”	null	n
C035	“name35”	null	n	C110	“name110”	null	n
C036	“name36”	null	y	C111	“name111”	null	y
C037	“name37”	null	n	C112	“name112”	null	n
C038	“name38”	null	n	C113	“name113”	null	n
C039	“name39”	null	y	C114	“name114”	null	y
C040	“name40”	null	n	C115	“name115”	null	n
C041	“name41”	null	n	C116	“name116”	null	n
C042	“name42”	null	y	C117	“name117”	null	y
C043	“name43”	null	n	C118	“name118”	null	n
C044	“name44”	null	n	C119	“name119”	null	n
C045	“name45”	null	y	C120	“name120”	null	y
C046	“name46”	null	n	C121	“name121”	null	n
C047	“name47”	null	n	C122	“name122”	null	n
C048	“name48”	null	y	C123	“name123”	null	y
C049	“name49”	null	n	C124	“name124”	null	n
C050	“name50”	null	n	C125	“name125”	null	n
C051	“name51”	null	y	C126	“name126”	null	y
C052	“name52”	null	n	C127	“name127”	null	n
C053	“name53”	null	n	C128	“name128”	null	n
C054	“name54”	null	y	C129	“name129”	null	y
C055	“name55”	null	n	C130	“name130”	null	n
C056	“name56”	null	n	C131	“name131”	null	n
C057	“name57”	null	y	C132	“name132”	null	y
C058	“name58”	null	n	C133	“name133”	null	n
C059	“name59”	null	n	C134	“name134”	null	n
C060	“name60”	null	y	C135	“name135”	null	y
C061	“name61”	null	n	C136	“name136”	null	n
C062	“name62”	null	n	C137	“name137”	null	n
C063	“name63”	null	y	C138	“name138”	null	y
C064	“name64”	null	n	C139	“name139”	null	n
C065	“name65”	null	n	C140	“name140”	null	n
C066	“name66”	null	y	C141	“name141”	null	y
C067	“name67”	null	n	C142	“name142”	null	n
C068	“name68”	null	n	C143	“name143”	null	n
C069	“name69”	null	y	C144	“name144”	null	y
C070	“name70”	null	n	C145	“name145”	null	n
C071	“name71”	null	n	C146	“name146”	null	n
C072	“name72”	null	y	C147	“name147”	null	y
C073	“name73”	null	n	C148	“name148”	null	n
C074	“name74”	null	n	C149	“name149”	null	n
C075	“name75”	null	y	C150	“name150”	null	y

Table C.4: U150 Location Table

ID	Address	x-coordinate	y-coordinate
L001	“depot”	0	0
L002	“addr1”	0.538671377	0.003049718
L003	“addr2”	0.82858205	0.005913712
L004	“addr3”	0.295861225	0.7976533
L005	“addr4”	-0.842058054	0.541947787
L006	“addr5”	1.970032959	-0.153264068
L007	“addr6”	0.647380496	-1.414584214
L008	“addr7”	0.644549627	0.048573065
L009	“addr8”	-0.135501397	-0.131232113
L010	“addr9”	-1.353336878	-1.617206768
L011	“addr10”	-0.511723594	-0.058041737
L012	“addr11”	-1.868745815	-0.672849874
L013	“addr12”	1.755203404	1.8599475
L014	“addr13”	-1.974996023	0.824697371
L015	“addr14”	-1.269010391	0.042495778
L016	“addr15”	0.125553072	-1.707788058
L017	“addr16”	-1.490946478	0.994357625
L018	“addr17”	0.169734523	-0.78850788
L019	“addr18”	-1.230985848	-1.562731095
L020	“addr19”	-1.670501027	0.520936634
L021	“addr20”	-0.914553121	1.893399755
L022	“addr21”	1.863682957	1.325593978
L023	“addr22”	0.959191801	1.873226955
L024	“addr23”	-1.451702247	0.36965138
L025	“addr24”	-0.249462437	1.559061815
L026	“addr25”	1.027554395	1.266114536
L027	“addr26”	1.525631272	-1.7612511
L028	“addr27”	-0.627634663	-1.345789788
L029	“addr28”	0.284500026	1.732335461
L030	“addr29”	-0.097402273	1.255825328
L031	“addr30”	-0.87890193	-0.332211611
L032	“addr31”	0.723885916	-0.771923017
L033	“addr32”	0.572826106	0.70012052
L034	“addr33”	-1.550663553	-0.740109955
L035	“addr34”	-1.250999578	-1.3684322
L036	“addr35”	0.220997486	-0.463761455
L037	“addr36”	-1.215219567	-1.121260541
L038	“addr37”	-1.584510464	1.994490057
L039	“addr38”	-0.25090487	1.845945842
L040	“addr39”	0.312728921	1.22307286
L041	“addr40”	1.342138821	1.5021893
L042	“addr41”	-0.473793385	-0.09653357
L043	“addr42”	0.550039951	-0.177415845
L044	“addr43”	-1.862185717	-0.12770242
L045	“addr44”	-0.118664627	-1.256745566
L046	“addr45”	-1.75025724	1.681105696
L047	“addr46”	1.891916336	-1.107962428
L048	“addr47”	-0.939490997	1.95712286
L049	“addr48”	-0.587857926	0.595605581
L050	“addr49”	0.591697111	-1.080807792
L051	“addr50”	-0.778152247	-0.473106529
L052	“addr51”	-0.163466929	-1.921264679
L053	“addr52”	-1.123992761	0.845010339
L054	“addr53”	1.370941585	0.456453574
L055	“addr54”	-0.034161236	1.79308453
L056	“addr55”	0.362266133	-1.428763705
L057	“addr56”	-1.692823387	1.45778013
L058	“addr57”	-0.569635972	1.008523758
L059	“addr58”	-1.620040081	-0.242499166
L060	“addr59”	-0.039680441	-1.448839688
L061	“addr60”	-0.200276764	1.714578292
L062	“addr61”	-0.764449919	1.65654824
L063	“addr62”	1.684003504	-0.319974487
L064	“addr63”	-0.845389841	0.489699059
L065	“addr64”	-1.516389997	1.417714277
L066	“addr65”	-1.68546934	0.772557388
L067	“addr66”	-1.819265487	1.818119024
L068	“addr67”	0.888061646	-0.614245176
L069	“addr68”	0.7129107	0.500049753
L070	“addr69”	0.991717826	-1.040267818
L071	“addr70”	-0.936137309	-1.32792221
L072	“addr71”	-1.863942733	0.617949665
L073	“addr72”	-0.330685781	-1.636359303
L074	“addr73”	-0.130844209	-1.414312976
L075	“addr74”	-1.38213961	-0.500908775

Table C.4: U150 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L076	“addr75”	-0.131230409	1.553713851
L077	“addr76”	-0.415963095	1.276240593
L078	“addr77”	-0.491789008	-0.5131041
L079	“addr78”	-1.487249413	-0.075777387
L080	“addr79”	0.315492453	0.993580143
L081	“addr80”	0.78126818	-0.846209293
L082	“addr81”	1.76482676	1.422385709
L083	“addr82”	-1.103008865	0.476439002
L084	“addr83”	1.484099718	1.528807861
L085	“addr84”	-1.253725614	-1.942979628
L086	“addr85”	1.231467505	0.854822328
L087	“addr86”	-0.008420322	-1.122184992
L088	“addr87”	-0.111244	-1.109538162
L089	“addr88”	-0.578714578	-0.269714574
L090	“addr89”	0.238877373	-1.36288626
L091	“addr90”	-1.111496255	1.914034926
L092	“addr91”	1.439004337	1.368071898
L093	“addr92”	1.932290507	0.510766104
L094	“addr93”	1.870778201	0.182656602
L095	“addr94”	-0.743337697	-0.285680464
L096	“addr95”	1.443355674	-0.624003253
L097	“addr96”	0.949866249	-1.063847324
L098	“addr97”	0.444631654	-1.302650051
L099	“addr98”	0.658382262	-1.731099303
L100	“addr99”	-0.319494262	-1.754381215
L101	“addr100”	1.887163201	-1.621058354
L102	“addr101”	1.911591987	1.051501014
L103	“addr102”	-0.820343529	-0.987829378
L104	“addr103”	1.491303348	0.321493205
L105	“addr104”	-0.861824263	0.035468396
L106	“addr105”	-0.371228499	0.869749628
L107	“addr106”	0.077226734	-0.700370254
L108	“addr107”	0.844595994	-0.560422263
L109	“addr108”	0.559420185	-1.402198436
L110	“addr109”	-0.043052931	0.417134303
L111	“addr110”	0.145637745	0.60675784
L112	“addr111”	-1.669922725	-0.134333006
L113	“addr112”	-1.601854089	1.936623149
L114	“addr113”	1.136829942	-0.140573503
L115	“addr114”	-1.492612988	0.277935161
L116	“addr115”	0.031261237	0.772157604
L117	“addr116”	-0.12979531	1.123220459
L118	“addr117”	1.897026004	1.475856843
L119	“addr118”	-0.470704928	-0.346463808
L120	“addr119”	0.231708739	0.331267346
L121	“addr120”	-0.409296457	-1.881285912
L122	“addr121”	1.246809597	1.402359877
L123	“addr122”	0.56822152	-1.596677151
L124	“addr123”	-0.092369142	-1.497164478
L125	“addr124”	1.060565167	-0.541352884
L126	“addr125”	-0.093163011	-1.51653983
L127	“addr126”	-1.750747916	1.894261513
L128	“addr127”	1.959994052	-1.405172305
L129	“addr128”	-0.587877521	1.261061182
L130	“addr129”	-1.169074744	-1.189072957
L131	“addr130”	-0.020518965	1.420888785
L132	“addr131”	1.139252261	0.674670499
L133	“addr132”	-1.383129705	-0.113816837
L134	“addr133”	-0.727981015	-0.472678492
L135	“addr134”	0.400513905	0.126350994
L136	“addr135”	-0.384168595	0.909541999
L137	“addr136”	1.358501346	-0.424151029
L138	“addr137”	1.845472864	0.472560366
L139	“addr138”	-0.517182671	-1.557757608
L140	“addr139”	0.149490052	-0.08625837
L141	“addr140”	-1.225650722	1.400787751
L142	“addr141”	-0.957227342	1.251623834
L143	“addr142”	-0.034962254	1.875368113
L144	“addr143”	1.810682948	0.1812655
L145	“addr144”	0.268366089	-1.156106721
L146	“addr145”	0.648821772	-1.698470428
L147	“addr146”	-1.158582716	-1.030445769
L148	“addr147”	0.207050544	1.681711664
L149	“addr148”	0.598756568	-1.652393861
L150	“addr149”	0.515916042	1.535700458

Table C.4: U150 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L151	“addr150”	0.600429488	-1.72936654
L152	“addr151”	-0.831838515	1.697640637
L153	“addr152”	-1.556602768	0.153532577
L154	“addr153”	-1.852466241	-0.856151035
L155	“addr154”	0.917761344	-1.877271355
L156	“addr155”	1.841631059	1.981515617
L157	“addr156”	1.646162095	-0.237575699
L158	“addr157”	1.545065014	0.125504784
L159	“addr158”	-1.493234751	0.123959975
L160	“addr159”	-1.846257422	-0.689651698
L161	“addr160”	-0.652717078	-0.657355107
L162	“addr161”	-1.229394497	1.135309602
L163	“addr162”	1.829507562	-1.930298093
L164	“addr163”	-1.412305388	0.516174087
L165	“addr164”	-1.799133655	-0.584752495
L166	“addr165”	-1.321060678	1.21820489
L167	“addr166”	-1.628462852	-0.788487125
L168	“addr167”	1.232601383	0.070483401
L169	“addr168”	0.882695864	-1.856211052
L170	“addr169”	-1.728162894	1.01263672
L171	“addr170”	-0.816499853	-0.340135172
L172	“addr171”	0.056368962	-0.699268155
L173	“addr172”	-1.908559696	-1.442995091
L174	“addr173”	-0.872207168	0
L175	“addr174”	0.392725006	0.758795677
L176	“addr175”	-0.644425461	-0.682943639
L177	“addr176”	-1.753603491	1.032471395
L178	“addr177”	0.360325253	-1.62222849
L179	“addr178”	0.387903503	-1.73938906
L180	“addr179”	0.335471343	0.727321189
L181	“addr180”	0.084890287	-0.630621875
L182	“addr181”	0.557969382	-0.863517495
L183	“addr182”	0.379992482	1.128759891
L184	“addr183”	-1.226752976	0.218618648
L185	“addr184”	-1.637540033	-0.22146227
L186	“addr185”	-1.8409187	-1.084720939
L187	“addr186”	1.042460878	1.797009027
L188	“addr187”	1.71765758	-0.916649536
L189	“addr188”	1.858948479	-0.882244012
L190	“addr189”	0.714583633	1.217572763
L191	“addr190”	0.934008194	0.713040924
L192	“addr191”	0.178164311	-0.231986198
L193	“addr192”	-0.106371018	-0.299310632
L194	“addr193”	-1.508225527	-1.437120203
L195	“addr194”	-0.593121907	0.994600977
L196	“addr195”	-0.243303074	-0.477182368
L197	“addr196”	-1.447512284	-0.992687072
L198	“addr197”	-1.127673604	-0.846351128
L199	“addr198”	-0.296107414	-1.973681802
L200	“addr199”	0.175169003	-0.769746199
L201	“addr200”	0.830791293	1.624125174
L202	“addr201”	-1.002400188	-1.364892736
L203	“addr202”	0.947332532	1.825324944
L204	“addr203”	-0.348129192	-1.509251408
L205	“addr204”	1.894061179	1.742213728
L206	“addr205”	-0.885604509	-1.473751161
L207	“addr206”	0.722788348	-0.99619966
L208	“addr207”	1.079862954	1.043519461
L209	“addr208”	-0.339417893	1.541307962
L210	“addr209”	0.074271708	1.900090141
L211	“addr210”	-1.038893323	-1.843437238
L212	“addr211”	1.960498144	-0.399239019
L213	“addr212”	1.908568134	-0.212057814
L214	“addr213”	1.439551258	1.891318754
L215	“addr214”	-1.178442522	-0.498164157
L216	“addr215”	-0.954806012	-1.826713077
L217	“addr216”	0.11111202	1.975856042
L218	“addr217”	1.649141941	-0.390540493
L219	“addr218”	1.279347146	-1.396686626
L220	“addr219”	-0.361416161	-1.034247953
L221	“addr220”	1.202043334	0.33675538
L222	“addr221”	-1.302702453	1.035682757
L223	“addr222”	-0.879397863	-0.99835186
L224	“addr223”	0.516168039	1.919578421
L225	“addr224”	1.475818988	1.049560038

Table C.4: U150 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L226	“addr225”	-1.221557337	-0.518619058
L227	“addr226”	1.047948332	1.583489921
L228	“addr227”	0.409593792	1.981712732
L229	“addr228”	-0.175427609	0.976262744
L230	“addr229”	1.351186997	-0.813362618
L231	“addr230”	1.599811124	-0.483803789
L232	“addr231”	1.01268572	-1.82266199
L233	“addr232”	0.515446158	1.252524276
L234	“addr233”	-1.92534535	0.636712179
L235	“addr234”	1.586561246	0.576343331
L236	“addr235”	0.461354229	1.564665892
L237	“addr236”	0.711331962	-1.938869863
L238	“addr237”	0.070251973	1.174671648
L239	“addr238”	0.746631051	-0.299840223
L240	“addr239”	0.240504897	-1.850066877
L241	“addr240”	-1.061855353	-0.083566503
L242	“addr241”	0.962000679	1.488284574
L243	“addr242”	1.325727141	1.584009738
L244	“addr243”	-0.528924138	-0.1592703
L245	“addr244”	1.701754165	1.787909207
L246	“addr245”	0.148560034	1.549311451
L247	“addr246”	-1.71773723	-1.836950443
L248	“addr247”	-0.469673373	1.515780006
L249	“addr248”	-1.542299376	-0.609055549
L250	“addr249”	-0.239936596	-1.416688972
L251	“addr250”	0.822136704	-1.203816558
L252	“addr251”	-1.041496367	-0.710693736
L253	“addr252”	0.136817419	-0.027009079
L254	“addr253”	1.752894694	0.648089655
L255	“addr254”	-1.034017497	-0.014557178
L256	“addr255”	-0.394991883	-1.951905071
L257	“addr256”	1.42988812	-1.076797493
L258	“addr257”	0.59552865	-1.601081931
L259	“addr258”	1.852013421	-1.542085151
L260	“addr259”	1.967190752	1.339395014
L261	“addr260”	0.034460805	0.533685264
L262	“addr261”	-0.001438856	-1.381359339
L263	“addr262”	-0.522411768	0.438343895
L264	“addr263”	0.674898677	1.687770179
L265	“addr264”	-0.933053559	-1.759516919
L266	“addr265”	-0.857946438	0.090802739
L267	“addr266”	0.582622516	0.940533104
L268	“addr267”	0.073898544	-1.580968635
L269	“addr268”	-0.156875514	-1.327574997
L270	“addr269”	-1.146683102	1.3939199
L271	“addr270”	0.377970893	-0.094331124
L272	“addr271”	-1.138697532	-0.930625145
L273	“addr272”	1.230182828	-1.164041371
L274	“addr273”	-0.811740175	-0.804984765
L275	“addr274”	0.821904842	-0.984305763
L276	“addr275”	1.726120487	1.45784978
L277	“addr276”	-1.226585215	-1.235905042
L278	“addr277”	0.869427162	0.620548927
L279	“addr278”	-0.852189033	0.009856619
L280	“addr279”	0.948064896	-1.499718829
L281	“addr280”	1.800700018	0.957589796
L282	“addr281”	0.033851723	-1.717134244
L283	“addr282”	1.346235567	-1.667446219
L284	“addr283”	-0.092268127	0.842772779
L285	“addr284”	-0.29210623	0.946485934
L286	“addr285”	-1.256153909	1.54028486
L287	“addr286”	-1.695600727	-1.059396054
L288	“addr287”	-0.538715395	1.392915152
L289	“addr288”	1.628248578	-1.69985769
L290	“addr289”	-1.286140719	-1.431042876
L291	“addr290”	-0.693692072	-1.901935352
L292	“addr291”	1.132764013	-1.173176077
L293	“addr292”	-1.31100507	-1.586126078
L294	“addr293”	-1.293478079	0.82461541
L295	“addr294”	-1.403416311	-0.013274335
L296	“addr295”	-1.922849449	-1.849867239
L297	“addr296”	-0.941795382	0.40032856
L298	“addr297”	1.051006725	-1.527658233
L299	“addr298”	-0.288857859	1.96040014
L300	“addr299”	1.767303898	-0.608933319
L301	“addr300”	-1.014937239	-0.586275744

Table C.5: U150 Request Table

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R001	C001	L002	L005	5/27/2003	7:24	D	5	1	7:18	7:28
R002	C002	L003	L006	5/27/2003	7:38	D	5	1	7:32	7:42
R003	C003	L004	L007	5/27/2003	7:48	D	5	1	7:43	7:53
R004	C004	L008	L011	5/27/2003	8:26	D	5	1	8:20	8:30
R005	C005	L009	L012	5/27/2003	8:05	P	5	1	7:59	8:09
R006	C006	L010	L013	5/27/2003	8:16	P	5	1	8:10	8:20
R007	C007	L014	L017	5/27/2003	9:10	P	5	1	9:05	9:15
R008	C008	L015	L018	5/27/2003	9:18	P	5	1	9:13	9:23
R009	C009	L016	L019	5/27/2003	10:07	D	5	1	10:01	10:11
R010	C010	L020	L023	5/27/2003	10:18	P	5	1	10:13	10:23
R011	C011	L021	L024	5/27/2003	11:03	D	5	1	10:57	11:07
R012	C012	L022	L025	5/27/2003	10:41	P	5	1	10:36	10:46
R013	C013	L026	L029	5/27/2003	12:02	D	5	1	11:57	12:07
R014	C014	L027	L030	5/27/2003	12:09	D	5	1	12:04	12:14
R015	C015	L028	L031	5/27/2003	12:20	D	5	1	12:14	12:24
R016	C016	L032	L035	5/27/2003	7:03	P	5	1	6:58	7:08
R017	C017	L033	L036	5/27/2003	7:13	P	5	1	7:07	7:17
R018	C018	L034	L037	5/27/2003	7:52	D	5	1	7:47	7:57
R019	C019	L038	L041	5/27/2003	8:31	D	5	1	8:26	8:36
R020	C020	L039	L042	5/27/2003	8:44	D	5	1	8:38	8:48
R021	C021	L040	L043	5/27/2003	8:52	D	5	1	8:46	8:56
R022	C022	L044	L047	5/27/2003	9:49	D	5	1	9:44	9:54
R023	C023	L045	L048	5/27/2003	9:15	P	5	1	9:10	9:20
R024	C024	L046	L049	5/27/2003	9:30	P	5	1	9:25	9:35
R025	C025	L050	L053	5/27/2003	10:27	P	5	1	10:22	10:32
R026	C026	L051	L054	5/27/2003	11:13	D	5	1	11:07	11:17
R027	C027	L052	L055	5/27/2003	10:46	P	5	1	10:41	10:51
R028	C028	L056	L059	5/27/2003	12:12	D	5	1	12:07	12:17
R029	C029	L057	L060	5/27/2003	12:23	D	5	1	12:18	12:28
R030	C030	L058	L061	5/27/2003	12:02	P	5	1	11:57	12:07
R031	C031	L062	L065	5/27/2003	7:05	P	5	1	7:00	7:10
R032	C032	L063	L066	5/27/2003	7:48	D	5	1	7:43	7:53
R033	C033	L064	L067	5/27/2003	7:33	P	5	1	7:27	7:37
R034	C034	L068	L071	5/27/2003	8:41	D	5	1	8:36	8:46
R035	C035	L069	L072	5/27/2003	8:21	P	5	1	8:15	8:25
R036	C036	L070	L073	5/27/2003	8:30	P	5	1	8:25	8:35
R037	C037	L074	L077	5/27/2003	9:12	P	5	1	9:06	9:16
R038	C038	L075	L078	5/27/2003	9:50	D	5	1	9:45	9:55
R039	C039	L076	L079	5/27/2003	9:59	D	5	1	9:53	10:03
R040	C040	L080	L083	5/27/2003	10:47	D	5	1	10:42	10:52
R041	C041	L081	L084	5/27/2003	10:21	P	5	1	10:15	10:25
R042	C042	L082	L085	5/27/2003	11:19	D	5	1	11:13	11:23
R043	C043	L086	L089	5/27/2003	11:35	P	5	1	11:29	11:39
R044	C044	L087	L090	5/27/2003	11:47	P	5	1	11:41	11:51
R045	C045	L088	L091	5/27/2003	12:25	D	5	1	12:19	12:29
R046	C046	L092	L095	5/27/2003	7:06	P	5	1	7:00	7:10
R047	C047	L093	L096	5/27/2003	7:45	D	5	1	7:39	7:49
R048	C048	L094	L097	5/27/2003	7:20	P	5	1	7:14	7:24
R049	C049	L098	L101	5/27/2003	8:24	D	5	1	8:19	8:29
R050	C050	L099	L102	5/27/2003	8:05	P	5	1	7:59	8:09
R051	C051	L100	L103	5/27/2003	8:13	P	5	1	8:07	8:17
R052	C052	L104	L107	5/27/2003	9:05	P	5	1	9:00	9:10
R053	C053	L105	L108	5/27/2003	9:18	P	5	1	9:12	9:22
R054	C054	L106	L109	5/27/2003	9:25	P	5	1	9:20	9:30
R055	C055	L110	L113	5/27/2003	10:01	P	5	1	9:56	10:06
R056	C056	L111	L114	5/27/2003	10:07	P	5	1	10:01	10:11
R057	C057	L112	L115	5/27/2003	10:57	D	5	1	10:52	11:02
R058	C058	L116	L119	5/27/2003	11:38	D	5	1	11:33	11:43
R059	C059	L117	L120	5/27/2003	11:46	D	5	1	11:41	11:51
R060	C060	L118	L121	5/27/2003	11:58	D	5	1	11:53	12:03
R061	C061	L122	L125	5/27/2003	7:36	D	5	1	7:31	7:41
R062	C062	L123	L126	5/27/2003	7:20	P	5	1	7:14	7:24
R063	C063	L124	L127	5/27/2003	7:27	P	5	1	7:21	7:31
R064	C064	L128	L131	5/27/2003	9:04	D	5	1	8:59	9:09
R065	C065	L129	L132	5/27/2003	9:13	D	5	1	9:08	9:18
R066	C066	L130	L133	5/27/2003	8:51	P	5	1	8:45	8:55
R067	C067	L134	L137	5/27/2003	9:34	P	5	1	9:28	9:38
R068	C068	L135	L138	5/27/2003	10:10	D	5	1	10:05	10:15
R069	C069	L136	L139	5/27/2003	10:25	D	5	1	10:19	10:29
R070	C070	L140	L143	5/27/2003	11:00	D	5	1	10:54	11:04
R071	C071	L141	L144	5/27/2003	10:46	P	5	1	10:40	10:50
R072	C072	L142	L145	5/27/2003	10:52	P	5	1	10:46	10:56
R073	C073	L146	L149	5/27/2003	12:11	D	5	1	12:05	12:15
R074	C074	L147	L150	5/27/2003	11:41	P	5	1	11:36	11:46
R075	C075	L148	L151	5/27/2003	12:40	D	5	1	12:34	12:44

Table C.5: U150 Request Table *(continued)*

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R076	C076	L152	L155	5/27/2003	7:38	D	5	1	7:32	7:42
R077	C077	L153	L156	5/27/2003	7:55	D	5	1	7:49	7:59
R078	C078	L154	L157	5/27/2003	8:06	D	5	1	8:01	8:11
R079	C079	L158	L161	5/27/2003	8:13	P	5	1	8:07	8:17
R080	C080	L159	L162	5/27/2003	8:27	P	5	1	8:21	8:31
R081	C081	L160	L163	5/27/2003	9:12	D	5	1	9:06	9:16
R082	C082	L164	L167	5/27/2003	9:29	P	5	1	9:23	9:33
R083	C083	L165	L168	5/27/2003	10:13	D	5	1	10:07	10:17
R084	C084	L166	L169	5/27/2003	10:24	D	5	1	10:18	10:28
R085	C085	L170	L173	5/27/2003	11:09	D	5	1	11:04	11:14
R086	C086	L171	L174	5/27/2003	11:20	D	5	1	11:14	11:24
R087	C087	L172	L175	5/27/2003	10:58	P	5	1	10:53	11:03
R088	C088	L176	L179	5/27/2003	11:39	P	5	1	11:34	11:44
R089	C089	L177	L180	5/27/2003	11:51	P	5	1	11:45	11:55
R090	C090	L178	L181	5/27/2003	12:06	P	5	1	12:00	12:10
R091	C091	L182	L185	5/27/2003	7:31	D	5	1	7:26	7:36
R092	C092	L183	L186	5/27/2003	7:39	D	5	1	7:34	7:44
R093	C093	L184	L187	5/27/2003	7:56	D	5	1	7:51	8:01
R094	C094	L188	L191	5/27/2003	8:34	D	5	1	8:28	8:38
R095	C095	L189	L192	5/27/2003	8:43	D	5	1	8:37	8:47
R096	C096	L190	L193	5/27/2003	8:27	P	5	1	8:22	8:32
R097	C097	L194	L197	5/27/2003	8:59	P	5	1	8:53	9:03
R098	C098	L195	L198	5/27/2003	9:36	D	5	1	9:31	9:41
R099	C099	L196	L199	5/27/2003	9:21	P	5	1	9:16	9:26
R100	C100	L200	L203	5/27/2003	10:38	D	5	1	10:33	10:43
R101	C101	L201	L204	5/27/2003	10:07	P	5	1	10:01	10:11
R102	C102	L202	L205	5/27/2003	11:11	D	5	1	11:06	11:16
R103	C103	L206	L209	5/27/2003	12:00	D	5	1	11:54	12:04
R104	C104	L207	L210	5/27/2003	12:06	D	5	1	12:01	12:11
R105	C105	L208	L211	5/27/2003	11:50	P	5	1	11:45	11:55
R106	C106	L212	L215	5/27/2003	7:39	D	5	1	7:33	7:43
R107	C107	L213	L216	5/27/2003	7:12	P	5	1	7:06	7:16
R108	C108	L214	L217	5/27/2003	7:23	P	5	1	7:18	7:28
R109	C109	L218	L221	5/27/2003	8:18	P	5	1	8:13	8:23
R110	C110	L219	L222	5/27/2003	8:26	P	5	1	8:21	8:31
R111	C111	L220	L223	5/27/2003	9:12	D	5	1	9:06	9:16
R112	C112	L224	L227	5/27/2003	9:26	P	5	1	9:21	9:31
R113	C113	L225	L228	5/27/2003	9:35	P	5	1	9:30	9:40
R114	C114	L226	L229	5/27/2003	9:50	P	5	1	9:44	9:54
R115	C115	L230	L233	5/27/2003	10:32	P	5	1	10:26	10:36
R116	C116	L231	L234	5/27/2003	11:14	D	5	1	11:09	11:19
R117	C117	L232	L235	5/27/2003	11:30	D	5	1	11:24	11:34
R118	C118	L236	L239	5/27/2003	12:19	D	5	1	12:14	12:24
R119	C119	L237	L240	5/27/2003	11:55	P	5	1	11:49	11:59
R120	C120	L238	L241	5/27/2003	12:41	D	5	1	12:35	12:45
R121	C121	L242	L245	5/27/2003	7:05	P	5	1	7:00	7:10
R122	C122	L243	L246	5/27/2003	7:48	D	5	1	7:42	7:52
R123	C123	L244	L247	5/27/2003	7:24	P	5	1	7:19	7:29
R124	C124	L248	L251	5/27/2003	8:20	P	5	1	8:15	8:25
R125	C125	L249	L252	5/27/2003	9:01	D	5	1	8:55	9:05
R126	C126	L250	L253	5/27/2003	9:10	D	5	1	9:04	9:14
R127	C127	L254	L257	5/27/2003	9:56	D	5	1	9:50	10:00
R128	C128	L255	L258	5/27/2003	10:04	D	5	1	9:58	10:08
R129	C129	L256	L259	5/27/2003	9:45	P	5	1	9:39	9:49
R130	C130	L260	L263	5/27/2003	10:26	P	5	1	10:21	10:31
R131	C131	L261	L264	5/27/2003	10:38	P	5	1	10:32	10:42
R132	C132	L262	L265	5/27/2003	10:48	P	5	1	10:43	10:53
R133	C133	L266	L269	5/27/2003	11:36	P	5	1	11:31	11:41
R134	C134	L267	L270	5/27/2003	12:19	D	5	1	12:13	12:23
R135	C135	L268	L271	5/27/2003	11:59	P	5	1	11:53	12:03
R136	C136	L272	L275	5/27/2003	7:04	P	5	1	6:59	7:09
R137	C137	L273	L276	5/27/2003	7:51	D	5	1	7:45	7:55
R138	C138	L274	L277	5/27/2003	8:08	D	5	1	8:02	8:12
R139	C139	L278	L281	5/27/2003	8:56	D	5	1	8:51	9:01
R140	C140	L279	L282	5/27/2003	8:31	P	5	1	8:26	8:36
R141	C141	L280	L283	5/27/2003	8:43	P	5	1	8:38	8:48
R142	C142	L284	L287	5/27/2003	9:33	P	5	1	9:28	9:38
R143	C143	L285	L288	5/27/2003	9:39	P	5	1	9:34	9:44
R144	C144	L286	L289	5/27/2003	9:48	P	5	1	9:42	9:52
R145	C145	L290	L293	5/27/2003	11:14	D	5	1	11:09	11:19
R146	C146	L291	L294	5/27/2003	10:51	P	5	1	10:45	10:55
R147	C147	L292	L295	5/27/2003	11:34	D	5	1	11:29	11:39
R148	C148	L296	L299	5/27/2003	12:27	D	5	1	12:21	12:31
R149	C149	L297	L300	5/27/2003	11:57	P	5	1	11:52	12:02
R150	C150	L298	L301	5/27/2003	12:55	D	5	1	12:49	12:59

Appendix D

U150A Data Set

Table D.1: U150A Model Table

MRT	FIXED	60	TW	VARY	WAIT	30	SPEED	20	ROUTE	480

Table D.2: U150A Vehicle Table

ID	Capacity	Wheelchairs
V01	4	2
V02	4	2
V03	4	2
V04	4	2
V05	4	2
V06	4	2
V07	4	2
V08	4	2
V09	4	2
V10	4	2

Table D.3: U150A Client Table

ID	Name	Address	Wheelchair	ID	Name	Address	Wheelchair
C001	“name1”	null	n	C076	“name76”	null	n
C002	“name2”	null	n	C077	“name77”	null	n
C003	“name3”	null	y	C078	“name78”	null	y
C004	“name4”	null	n	C079	“name79”	null	n
C005	“name5”	null	n	C080	“name80”	null	n
C006	“name6”	null	y	C081	“name81”	null	y
C007	“name7”	null	n	C082	“name82”	null	n
C008	“name8”	null	n	C083	“name83”	null	n
C009	“name9”	null	y	C084	“name84”	null	y
C010	“name10”	null	n	C085	“name85”	null	n
C011	“name11”	null	n	C086	“name86”	null	n
C012	“name12”	null	y	C087	“name87”	null	y
C013	“name13”	null	n	C088	“name88”	null	n
C014	“name14”	null	n	C089	“name89”	null	n
C015	“name15”	null	y	C090	“name90”	null	y
C016	“name16”	null	n	C091	“name91”	null	n
C017	“name17”	null	n	C092	“name92”	null	n
C018	“name18”	null	y	C093	“name93”	null	y
C019	“name19”	null	n	C094	“name94”	null	n
C020	“name20”	null	n	C095	“name95”	null	n
C021	“name21”	null	y	C096	“name96”	null	y
C022	“name22”	null	n	C097	“name97”	null	n
C023	“name23”	null	n	C098	“name98”	null	n
C024	“name24”	null	y	C099	“name99”	null	y
C025	“name25”	null	n	C100	“name100”	null	n
C026	“name26”	null	n	C101	“name101”	null	n
C027	“name27”	null	y	C102	“name102”	null	y
C028	“name28”	null	n	C103	“name103”	null	n
C029	“name29”	null	n	C104	“name104”	null	n
C030	“name30”	null	y	C105	“name105”	null	y
C031	“name31”	null	n	C106	“name106”	null	n
C032	“name32”	null	n	C107	“name107”	null	n
C033	“name33”	null	y	C108	“name108”	null	y
C034	“name34”	null	n	C109	“name109”	null	n
C035	“name35”	null	n	C110	“name110”	null	n
C036	“name36”	null	y	C111	“name111”	null	y
C037	“name37”	null	n	C112	“name112”	null	n
C038	“name38”	null	n	C113	“name113”	null	n
C039	“name39”	null	y	C114	“name114”	null	y
C040	“name40”	null	n	C115	“name115”	null	n
C041	“name41”	null	n	C116	“name116”	null	n
C042	“name42”	null	y	C117	“name117”	null	y
C043	“name43”	null	n	C118	“name118”	null	n
C044	“name44”	null	n	C119	“name119”	null	n
C045	“name45”	null	y	C120	“name120”	null	y
C046	“name46”	null	n	C121	“name121”	null	n
C047	“name47”	null	n	C122	“name122”	null	n
C048	“name48”	null	y	C123	“name123”	null	y
C049	“name49”	null	n	C124	“name124”	null	n
C050	“name50”	null	n	C125	“name125”	null	n
C051	“name51”	null	y	C126	“name126”	null	y
C052	“name52”	null	n	C127	“name127”	null	n
C053	“name53”	null	n	C128	“name128”	null	n
C054	“name54”	null	y	C129	“name129”	null	y
C055	“name55”	null	n	C130	“name130”	null	n
C056	“name56”	null	n	C131	“name131”	null	n
C057	“name57”	null	y	C132	“name132”	null	y
C058	“name58”	null	n	C133	“name133”	null	n
C059	“name59”	null	n	C134	“name134”	null	n
C060	“name60”	null	y	C135	“name135”	null	y
C061	“name61”	null	n	C136	“name136”	null	n
C062	“name62”	null	n	C137	“name137”	null	n
C063	“name63”	null	y	C138	“name138”	null	y
C064	“name64”	null	n	C139	“name139”	null	n
C065	“name65”	null	n	C140	“name140”	null	n
C066	“name66”	null	y	C141	“name141”	null	y
C067	“name67”	null	n	C142	“name142”	null	n
C068	“name68”	null	n	C143	“name143”	null	n
C069	“name69”	null	y	C144	“name144”	null	y
C070	“name70”	null	n	C145	“name145”	null	n
C071	“name71”	null	n	C146	“name146”	null	n
C072	“name72”	null	y	C147	“name147”	null	y
C073	“name73”	null	n	C148	“name148”	null	n
C074	“name74”	null	n	C149	“name149”	null	n
C075	“name75”	null	y	C150	“name150”	null	y

Table D.4: U150A Location Table

ID	Address	x-coordinate	y-coordinate
L001	“depot”	0	0
L002	“addr1”	-1.538196555	0.223950404
L003	“addr2”	1.683537735	-1.981953296
L004	“addr3”	0.614200749	-0.993483745
L005	“addr4”	-0.415484199	-1.910197129
L006	“addr5”	1.879915136	1.304694884
L007	“addr6”	0.820609016	1.23214828
L008	“addr7”	-0.927593258	1.734326284
L009	“addr8”	-0.307924563	-0.391765655
L010	“addr9”	-0.723973985	1.982970791
L011	“addr10”	-0.78019476	1.388825307
L012	“addr11”	-0.175354644	0.611894078
L013	“addr12”	-1.81490473	-1.497289202
L014	“addr13”	-0.100077625	1.179253055
L015	“addr14”	0.170954417	0.463750753
L016	“addr15”	-1.642282369	-1.608362845
L017	“addr16”	0.16853601	-0.735088506
L018	“addr17”	1.082591071	0.588773302
L019	“addr18”	-0.933795516	1.489803345
L020	“addr19”	-1.013287414	-0.411139281
L021	“addr20”	-1.782354297	-0.922609581
L022	“addr21”	0.710924283	1.769658721
L023	“addr22”	-0.710660525	1.585696775
L024	“addr23”	0.152100145	-0.766896106
L025	“addr24”	-1.283382864	0.335491336
L026	“addr25”	0.132356151	-1.001106867
L027	“addr26”	1.351434302	-1.524912273
L028	“addr27”	-0.157932979	1.395121541
L029	“addr28”	0.263895575	0.792554806
L030	“addr29”	1.130055447	-1.838275453
L031	“addr30”	-0.836527838	1.011786899
L032	“addr31”	1.280722978	-1.744875107
L033	“addr32”	-1.465237687	0.344040239
L034	“addr33”	1.609300577	0.940146148
L035	“addr34”	-1.663450686	-0.773300879
L036	“addr35”	0.022800579	1.477907772
L037	“addr36”	0.99181738	0.466189524
L038	“addr37”	-0.364339305	-1.433926579
L039	“addr38”	0.400293609	-1.587883323
L040	“addr39”	-1.404526039	-1.339517929
L041	“addr40”	-0.166010619	1.324856586
L042	“addr41”	0.465608529	0.404844466
L043	“addr42”	1.830120591	-1.241008956
L044	“addr43”	-0.172077772	0.617848336
L045	“addr44”	0.574694171	-1.429872146
L046	“addr45”	-1.78980978	-0.365333131
L047	“addr46”	1.773951833	-0.937191528
L048	“addr47”	-0.99073828	-1.727474176
L049	“addr48”	0.639770188	-1.167641349
L050	“addr49”	0.7723765	-0.712362724
L051	“addr50”	1.369022483	0.013693829
L052	“addr51”	0.015472109	0.913467303
L053	“addr52”	1.375812826	1.93525083
L054	“addr53”	-0.301279468	-1.082041871
L055	“addr54”	-1.460599859	-0.413632195
L056	“addr55”	1.545942362	0.589984241
L057	“addr56”	1.859535587	-1.158403356
L058	“addr57”	-0.414052814	-0.837023387
L059	“addr58”	-0.835156757	-1.954416598
L060	“addr59”	0.610260341	0.406305678
L061	“addr60”	-0.101219326	-0.652687301
L062	“addr61”	-1.316900469	-1.537329276
L063	“addr62”	-1.488379095	-0.917335347
L064	“addr63”	1.405871576	1.482397982
L065	“addr64”	-1.384194718	0.53939122
L066	“addr65”	1.329866558	1.20320818
L067	“addr66”	-0.184078496	-1.401224701
L068	“addr67”	-1.272767788	0.346668523
L069	“addr68”	-1.996531624	0.03339096
L070	“addr69”	-0.128282648	1.550201693
L071	“addr70”	1.910474073	1.65053987
L072	“addr71”	-1.952918905	1.526179611
L073	“addr72”	-0.155057361	-0.993904449
L074	“addr73”	0.692467552	0.237287312
L075	“addr74”	0.785007155	1.073643785

Table D.4: U150A Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L076	“addr75”	-0.589543681	-0.900003053
L077	“addr76”	-0.612422888	1.03896159
L078	“addr77”	-1.995098383	0.575649729
L079	“addr78”	-1.02882896	0.300036859
L080	“addr79”	1.282080462	1.981298177
L081	“addr80”	-1.838465286	1.787983513
L082	“addr81”	0.04007765	-1.04988031
L083	“addr82”	1.642547601	-0.115634497
L084	“addr83”	1.487739926	-1.306134509
L085	“addr84”	1.333830897	1.495599262
L086	“addr85”	1.275938105	-1.932531139
L087	“addr86”	-0.679626915	0.255258015
L088	“addr87”	1.390283994	-1.591410989
L089	“addr88”	-0.011603055	1.360261849
L090	“addr89”	-0.65414763	0.222527136
L091	“addr90”	-1.924401176	-1.792911239
L092	“addr91”	0.440867339	-1.99439615
L093	“addr92”	-0.277205728	1.779302338
L094	“addr93”	0.267716198	-1.152253479
L095	“addr94”	-0.882085127	-0.352154252
L096	“addr95”	1.246062268	1.927688006
L097	“addr96”	1.952203353	1.480708264
L098	“addr97”	0.634232584	-0.388161542
L099	“addr98”	-1.337638402	0.118726984
L100	“addr99”	-1.708535404	-1.409455905
L101	“addr100”	1.311314171	1.559443561
L102	“addr101”	0.404999429	-0.666893096
L103	“addr102”	-0.035846357	1.601089613
L104	“addr103”	0.462419043	-1.884408589
L105	“addr104”	-0.455259064	-0.566065953
L106	“addr105”	0.94102828	-0.02689469
L107	“addr106”	-1.587657493	-0.873621649
L108	“addr107”	-1.497511275	1.150892721
L109	“addr108”	1.276587881	-1.261655361
L110	“addr109”	-0.077679183	-0.02376075
L111	“addr110”	1.572941982	-0.295343017
L112	“addr111”	1.550293188	1.545840282
L113	“addr112”	-1.950209892	-1.833728405
L114	“addr113”	-0.210874139	-1.77905713
L115	“addr114”	-0.927859649	1.087368877
L116	“addr115”	-0.104658224	-1.354772339
L117	“addr116”	0.025854995	-0.718523183
L118	“addr117”	-0.652496218	-1.151948633
L119	“addr118”	-1.789772703	-1.120049804
L120	“addr119”	0.093484973	-0.031669964
L121	“addr120”	-0.26437097	-1.688712635
L122	“addr121”	-0.820178119	-1.76800393
L123	“addr122”	-1.803143125	-0.480077669
L124	“addr123”	-0.417389579	0.038620473
L125	“addr124”	1.271714926	-0.061713822
L126	“addr125”	0.500807186	0.360447595
L127	“addr126”	0.898274928	1.636087552
L128	“addr127”	0.939208187	-1.870254868
L129	“addr128”	1.313164051	0.105919257
L130	“addr129”	-1.900219622	-0.111970139
L131	“addr130”	0.987133254	1.30445337
L132	“addr131”	-1.083570138	-1.306616446
L133	“addr132”	1.373069359	1.853643324
L134	“addr133”	1.009696132	1.129732606
L135	“addr134”	-0.614482933	1.71650993
L136	“addr135”	-1.595733065	-0.765381382
L137	“addr136”	-0.340315904	1.197239709
L138	“addr137”	-0.456383829	-1.74035466
L139	“addr138”	0.891810836	0.775486726
L140	“addr139”	-1.547526698	-0.599558359
L141	“addr140”	-0.058310318	1.069919217
L142	“addr141”	-1.660762626	-1.019360893
L143	“addr142”	1.882026093	-0.863787835
L144	“addr143”	-1.503142999	-1.952983484
L145	“addr144”	-0.191759137	0.339843
L146	“addr145”	0.052103004	-1.069379564
L147	“addr146”	-1.023782187	0.204201638
L148	“addr147”	-0.070433858	-1.94867053
L149	“addr148”	1.714409167	1.919684683
L150	“addr149”	1.338119762	-0.909018078

Table D.4: U150A Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L151	“addr150”	-1.439386467	-1.390400436
L152	“addr151”	1.153993197	-0.367152344
L153	“addr152”	-0.797346427	0.983858139
L154	“addr153”	-0.233993541	-0.537123949
L155	“addr154”	0.056080042	-1.328400162
L156	“addr155”	1.264925145	-0.049186466
L157	“addr156”	-0.309059638	-0.690963567
L158	“addr157”	-0.699700107	-0.723064386
L159	“addr158”	-1.176446614	1.368969358
L160	“addr159”	0.494560573	1.742798116
L161	“addr160”	-1.781861075	0.753586608
L162	“addr161”	-1.004458217	-0.997933483
L163	“addr162”	-0.402943166	-0.667699495
L164	“addr163”	-1.434977079	-1.229860591
L165	“addr164”	1.537317729	-0.108453729
L166	“addr165”	-1.457727432	0.418938071
L167	“addr166”	0.334931065	0.373616998
L168	“addr167”	1.087991429	-1.164109244
L169	“addr168”	-1.052434099	0.816764998
L170	“addr169”	-0.694542988	-1.976204556
L171	“addr170”	-0.620097136	1.068958417
L172	“addr171”	1.152077027	-1.859096577
L173	“addr172”	1.254964818	0.132837356
L174	“addr173”	0.495966208	0.028219058
L175	“addr174”	-0.181383302	0.90956792
L176	“addr175”	0.238567083	1.743669828
L177	“addr176”	-1.746664392	1.144452536
L178	“addr177”	-0.88425875	-1.779543211
L179	“addr178”	1.93125379	0.537308533
L180	“addr179”	1.606164259	1.88955024
L181	“addr180”	-1.884613303	-0.734216404
L182	“addr181”	-0.682736446	-1.04604687
L183	“addr182”	-0.378135817	0.579933504
L184	“addr183”	-1.714655456	-1.024989314
L185	“addr184”	1.084886492	-0.338455946
L186	“addr185”	1.817957119	0.528700474
L187	“addr186”	1.720037224	-1.398165746
L188	“addr187”	-1.177040254	-1.087505622
L189	“addr188”	0.860487336	1.136334952
L190	“addr189”	1.010146913	-0.768961775
L191	“addr190”	0.044534233	-1.314107086
L192	“addr191”	-1.082291581	1.661554032
L193	“addr192”	0.061823763	1.287907611
L194	“addr193”	-1.592800695	-0.702476657
L195	“addr194”	-0.880490499	1.162062115
L196	“addr195”	0.08710159	0.691867757
L197	“addr196”	-0.831488059	-1.060031913
L198	“addr197”	-1.933531139	-0.931501087
L199	“addr198”	-0.566056296	0.998299489
L200	“addr199”	1.506547431	-1.856234138
L201	“addr200”	-0.67732557	-1.129852927
L202	“addr201”	-1.370322773	-1.403015269
L203	“addr202”	-0.865083875	0.369480086
L204	“addr203”	0.161179771	1.451891478
L205	“addr204”	0.38561455	0.848761138
L206	“addr205”	1.621347736	-1.021485328
L207	“addr206”	0.045338572	1.919912127
L208	“addr207”	-0.548943652	0.197621703
L209	“addr208”	0.639267379	0.003989827
L210	“addr209”	0.645923087	0.667547492
L211	“addr210”	0.896778427	-1.764918046
L212	“addr211”	-0.609649516	-1.044127078
L213	“addr212”	1.523050035	-1.470906048
L214	“addr213”	1.124188169	1.04607335
L215	“addr214”	1.667993186	-1.961748366
L216	“addr215”	-1.458837301	-1.098083564
L217	“addr216”	1.796738051	1.747299633
L218	“addr217”	1.209625279	0.510043932
L219	“addr218”	-0.08019561	-1.372464025
L220	“addr219”	-1.17742443	1.943828089
L221	“addr220”	-1.480008213	0.549973571
L222	“addr221”	1.52532628	-0.660750808
L223	“addr222”	1.016933162	-0.930917758
L224	“addr223”	1.335578604	0.14619865
L225	“addr224”	1.239158398	-0.776285748

Table D.4: U150A Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L226	“addr225”	0.362085648	0.616266316
L227	“addr226”	1.127424782	1.661979064
L228	“addr227”	-1.655513459	-1.789271814
L229	“addr228”	-1.228025894	0.681984116
L230	“addr229”	-0.021699301	-0.539893487
L231	“addr230”	-1.109715136	-1.809481845
L232	“addr231”	-0.444522609	1.779330513
L233	“addr232”	-1.019208143	-1.883092092
L234	“addr233”	-1.357696827	-0.159816954
L235	“addr234”	0.641954586	-1.520564454
L236	“addr235”	-0.16712593	1.325372285
L237	“addr236”	-1.075040627	-0.913138087
L238	“addr237”	0.057747702	1.236227766
L239	“addr238”	1.092248281	-0.232244977
L240	“addr239”	0.493795508	-1.601327991
L241	“addr240”	-0.217027113	1.055479312
L242	“addr241”	0.812681617	-1.590221004
L243	“addr242”	-1.659621148	0.416861219
L244	“addr243”	1.321801943	1.989883822
L245	“addr244”	1.93503065	-1.602303782
L246	“addr245”	1.492921885	-1.447132979
L247	“addr246”	-0.87830381	-1.531574087
L248	“addr247”	1.440159105	-0.220109258
L249	“addr248”	0.596612362	1.88846433
L250	“addr249”	0.100362771	1.079521882
L251	“addr250”	-0.824236215	0.567812335
L252	“addr251”	0.185346658	-1.960166962
L253	“addr252”	-0.606291117	0.68975387
L254	“addr253”	-0.583158578	-1.674271875
L255	“addr254”	0.626397439	-0.231416428
L256	“addr255”	-1.878564098	0.947396222
L257	“addr256”	0.270326983	1.851473327
L258	“addr257”	0.95418408	0.423452668
L259	“addr258”	1.264293085	1.733739287
L260	“addr259”	-0.664038568	1.084983919
L261	“addr260”	1.118895347	-0.658070469
L262	“addr261”	-0.868940896	0.614368282
L263	“addr262”	-0.377134027	-0.150693942
L264	“addr263”	1.157911969	1.88539997
L265	“addr264”	1.524194554	1.623454225
L266	“addr265”	-1.682737902	-1.731098415
L267	“addr266”	0.737929105	0.765905935
L268	“addr267”	-1.56153445	1.790666248
L269	“addr268”	-1.98856088	-0.211178692
L270	“addr269”	0.386057189	1.876225959
L271	“addr270”	0.481833857	-1.116071217
L272	“addr271”	-1.464906556	-1.110400563
L273	“addr272”	-0.810866547	-0.725919827
L274	“addr273”	-0.584799785	-0.729171155
L275	“addr274”	-0.637301327	-0.702066836
L276	“addr275”	0.139658661	-0.200227255
L277	“addr276”	0.177329313	-0.836569547
L278	“addr277”	1.17726844	-0.638725317
L279	“addr278”	1.679350532	1.715900637
L280	“addr279”	-1.908866731	1.559488043
L281	“addr280”	0.711907959	0.101762103
L282	“addr281”	-1.940078211	-1.373209209
L283	“addr282”	0.922604442	-1.131031742
L284	“addr283”	1.31356006	0.295381265
L285	“addr284”	0.315512756	-1.169513016
L286	“addr285”	-0.943278775	-1.352699078
L287	“addr286”	1.313506116	-0.072734548
L288	“addr287”	0.019882989	1.431289512
L289	“addr288”	0.92575767	0.362286377
L290	“addr289”	-0.530871916	1.932530141
L291	“addr290”	-1.630838682	-1.562521677
L292	“addr291”	-0.677727385	-0.795998576
L293	“addr292”	0.371335778	-0.189600607
L294	“addr293”	-0.039689498	-1.684192194
L295	“addr294”	-1.613219954	1.297768562
L296	“addr295”	1.389569509	-1.748482335
L297	“addr296”	1.086652947	-1.111922393
L298	“addr297”	1.601381263	-0.82180779
L299	“addr298”	1.736556931	0.285669768
L300	“addr299”	-0.707527359	-0.960552937
L301	“addr300”	-0.544387602	-0.906169674

Table D.5: U150A Request Table

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R001	C001	L002	L005	5/27/2003	7:40	D	5	1	7:34	7:44
R002	C002	L003	L006	5/27/2003	7:57	D	5	1	7:51	8:01
R003	C003	L004	L007	5/27/2003	8:05	D	5	0	7:59	8:09
R004	C004	L008	L011	5/27/2003	8:15	P	5	1	8:10	8:20
R005	C005	L009	L012	5/27/2003	8:54	D	5	1	8:48	8:58
R006	C006	L010	L013	5/27/2003	9:07	D	5	0	9:02	9:12
R007	C007	L014	L017	5/27/2003	9:22	P	5	1	9:16	9:26
R008	C008	L015	L018	5/27/2003	10:03	D	5	1	9:57	10:07
R009	C009	L016	L019	5/27/2003	10:15	D	5	0	10:09	10:19
R010	C010	L020	L023	5/27/2003	10:25	P	5	1	10:20	10:30
R011	C011	L021	L024	5/27/2003	10:33	P	5	1	10:28	10:38
R012	C012	L022	L025	5/27/2003	10:49	P	5	0	10:44	10:54
R013	C013	L026	L029	5/27/2003	12:03	D	5	1	11:58	12:08
R014	C014	L027	L030	5/27/2003	12:17	D	5	1	12:11	12:21
R015	C015	L028	L031	5/27/2003	11:56	P	5	0	11:51	12:01
R016	C016	L032	L035	5/27/2003	7:06	P	5	1	7:01	7:11
R017	C017	L033	L036	5/27/2003	8:06	D	5	1	8:00	8:10
R018	C018	L034	L037	5/27/2003	7:36	P	5	0	7:31	7:41
R019	C019	L038	L041	5/27/2003	8:27	P	5	1	8:21	8:31
R020	C020	L039	L042	5/27/2003	8:34	P	5	1	8:29	8:39
R021	C021	L040	L043	5/27/2003	9:18	D	5	0	9:13	9:23
R022	C022	L044	L047	5/27/2003	10:12	D	5	1	10:06	10:16
R023	C023	L045	L048	5/27/2003	9:43	P	5	1	9:38	9:48
R024	C024	L046	L049	5/27/2003	10:35	D	5	0	10:30	10:40
R025	C025	L050	L053	5/27/2003	10:42	P	5	1	10:36	10:46
R026	C026	L051	L054	5/27/2003	10:50	P	5	1	10:44	10:54
R027	C027	L052	L055	5/27/2003	11:00	P	5	0	10:54	11:04
R028	C028	L056	L059	5/27/2003	12:19	D	5	1	12:14	12:24
R029	C029	L057	L060	5/27/2003	11:59	P	5	1	11:53	12:03
R030	C030	L058	L061	5/27/2003	12:11	P	5	0	12:05	12:15
R031	C031	L062	L065	5/27/2003	7:43	D	5	1	7:38	7:48
R032	C032	L063	L066	5/27/2003	7:56	D	5	1	7:51	8:01
R033	C033	L064	L067	5/27/2003	8:11	D	5	0	8:05	8:15
R034	C034	L068	L071	5/27/2003	8:22	P	5	1	8:16	8:26
R035	C035	L069	L072	5/27/2003	8:29	P	5	1	8:24	8:34
R036	C036	L070	L073	5/27/2003	9:23	D	5	0	9:18	9:28
R037	C037	L074	L077	5/27/2003	10:03	D	5	1	9:58	10:08
R038	C038	L075	L078	5/27/2003	9:40	P	5	1	9:35	9:45
R039	C039	L076	L079	5/27/2003	9:53	P	5	0	9:47	9:57
R040	C040	L080	L083	5/27/2003	11:14	D	5	1	11:09	11:19
R041	C041	L081	L084	5/27/2003	11:23	D	5	1	11:18	11:28
R042	C042	L082	L085	5/27/2003	11:04	P	5	0	10:58	11:08
R043	C043	L086	L089	5/27/2003	12:34	D	5	1	12:28	12:38
R044	C044	L087	L090	5/27/2003	12:43	D	5	1	12:37	12:47
R045	C045	L088	L091	5/27/2003	12:19	P	5	0	12:13	12:23
R046	C046	L092	L095	5/27/2003	7:46	D	5	1	7:40	7:50
R047	C047	L093	L096	5/27/2003	7:23	P	5	1	7:17	7:27
R048	C048	L094	L097	5/27/2003	7:37	P	5	0	7:31	7:41
R049	C049	L098	L101	5/27/2003	8:20	P	5	1	8:14	8:24
R050	C050	L099	L102	5/27/2003	9:10	D	5	1	9:05	9:15
R051	C051	L100	L103	5/27/2003	8:40	P	5	0	8:35	8:45
R052	C052	L104	L107	5/27/2003	10:10	D	5	1	10:05	10:15
R053	C053	L105	L108	5/27/2003	10:21	D	5	1	10:16	10:26
R054	C054	L106	L109	5/27/2003	10:37	D	5	0	10:32	10:42
R055	C055	L110	L113	5/27/2003	11:28	D	5	1	11:22	11:32
R056	C056	L111	L114	5/27/2003	10:58	P	5	1	10:52	11:02
R057	C057	L112	L115	5/27/2003	11:08	P	5	0	11:03	11:13
R058	C058	L116	L119	5/27/2003	12:27	D	5	1	12:22	12:32
R059	C059	L117	L120	5/27/2003	12:12	P	5	1	12:06	12:16
R060	C060	L118	L121	5/27/2003	12:19	P	5	0	12:14	12:24
R061	C061	L122	L125	5/27/2003	7:35	D	5	1	7:30	7:40
R062	C062	L123	L126	5/27/2003	7:16	P	5	1	7:10	7:20
R063	C063	L124	L127	5/27/2003	7:25	P	5	0	7:20	7:30
R064	C064	L128	L131	5/27/2003	8:07	P	5	1	8:02	8:12
R065	C065	L129	L132	5/27/2003	9:03	D	5	1	8:57	9:07
R066	C066	L130	L133	5/27/2003	9:20	D	5	0	9:14	9:24
R067	C067	L134	L137	5/27/2003	9:27	P	5	1	9:22	9:32
R068	C068	L135	L138	5/27/2003	10:16	D	5	1	10:11	10:21
R069	C069	L136	L139	5/27/2003	9:50	P	5	0	9:45	9:55
R070	C070	L140	L143	5/27/2003	10:43	P	5	1	10:38	10:48
R071	C071	L141	L144	5/27/2003	10:55	P	5	1	10:49	10:59
R072	C072	L142	L145	5/27/2003	11:08	P	5	0	11:02	11:12
R073	C073	L146	L149	5/27/2003	12:01	P	5	1	11:56	12:06
R074	C074	L147	L150	5/27/2003	12:55	D	5	1	12:49	12:59
R075	C075	L148	L151	5/27/2003	13:08	D	5	0	13:03	13:13

Table D.5: U150A Request Table *(continued)*

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R076	C076	L152	L155	5/27/2003	7:33	D	5	1	7:28	7:38
R077	C077	L153	L156	5/27/2003	7:43	D	5	1	7:38	7:48
R078	C078	L154	L157	5/27/2003	7:26	P	5	0	7:20	7:30
R079	C079	L158	L161	5/27/2003	8:34	D	5	1	8:28	8:38
R080	C080	L159	L162	5/27/2003	8:11	P	5	1	8:06	8:16
R081	C081	L160	L163	5/27/2003	8:52	D	5	0	8:46	8:56
R082	C082	L164	L167	5/27/2003	9:39	D	5	1	9:34	9:44
R083	C083	L165	L168	5/27/2003	9:15	P	5	1	9:09	9:19
R084	C084	L166	L169	5/27/2003	9:29	P	5	0	9:23	9:33
R085	C085	L170	L173	5/27/2003	10:16	P	5	1	10:11	10:21
R086	C086	L171	L174	5/27/2003	10:31	P	5	1	10:25	10:35
R087	C087	L172	L175	5/27/2003	11:12	D	5	0	11:07	11:17
R088	C088	L176	L179	5/27/2003	12:02	D	5	1	11:56	12:06
R089	C089	L177	L180	5/27/2003	12:11	D	5	1	12:05	12:15
R090	C090	L178	L181	5/27/2003	11:46	P	5	0	11:40	11:50
R091	C091	L182	L185	5/27/2003	7:39	D	5	1	7:33	7:43
R092	C092	L183	L186	5/27/2003	7:14	P	5	1	7:08	7:18
R093	C093	L184	L187	5/27/2003	7:25	P	5	0	7:19	7:29
R094	C094	L188	L191	5/27/2003	8:45	D	5	1	8:39	8:49
R095	C095	L189	L192	5/27/2003	8:26	P	5	1	8:20	8:30
R096	C096	L190	L193	5/27/2003	9:08	D	5	0	9:02	9:12
R097	C097	L194	L197	5/27/2003	9:51	D	5	1	9:45	9:55
R098	C098	L195	L198	5/27/2003	9:32	P	5	1	9:26	9:36
R099	C099	L196	L199	5/27/2003	9:40	P	5	0	9:34	9:44
R100	C100	L200	L203	5/27/2003	10:56	D	5	1	10:51	11:01
R101	C101	L201	L204	5/27/2003	10:39	P	5	1	10:33	10:43
R102	C102	L202	L205	5/27/2003	11:13	D	5	0	11:07	11:17
R103	C103	L206	L209	5/27/2003	11:59	D	5	1	11:53	12:03
R104	C104	L207	L210	5/27/2003	12:06	D	5	1	12:00	12:10
R105	C105	L208	L211	5/27/2003	11:50	P	5	0	11:45	11:55
R106	C106	L212	L215	5/27/2003	7:42	D	5	1	7:36	7:46
R107	C107	L213	L216	5/27/2003	7:15	P	5	1	7:10	7:20
R108	C108	L214	L217	5/27/2003	8:15	D	5	0	8:09	8:19
R109	C109	L218	L221	5/27/2003	9:00	D	5	1	8:55	9:05
R110	C110	L219	L222	5/27/2003	9:15	D	5	1	9:10	9:20
R111	C111	L220	L223	5/27/2003	8:51	P	5	0	8:46	8:56
R112	C112	L224	L227	5/27/2003	9:57	D	5	1	9:51	10:01
R113	C113	L225	L228	5/27/2003	9:38	P	5	1	9:32	9:42
R114	C114	L226	L229	5/27/2003	9:48	P	5	0	9:42	9:52
R115	C115	L230	L233	5/27/2003	11:20	D	5	1	11:14	11:24
R116	C116	L231	L234	5/27/2003	10:48	P	5	1	10:42	10:52
R117	C117	L232	L235	5/27/2003	11:04	P	5	0	10:58	11:08
R118	C118	L236	L239	5/27/2003	11:56	P	5	1	11:51	12:01
R119	C119	L237	L240	5/27/2003	12:09	P	5	1	12:03	12:13
R120	C120	L238	L241	5/27/2003	12:54	D	5	0	12:48	12:58
R121	C121	L242	L245	5/27/2003	7:51	D	5	1	7:45	7:55
R122	C122	L243	L246	5/27/2003	7:20	P	5	1	7:14	7:24
R123	C123	L244	L247	5/27/2003	7:35	P	5	0	7:30	7:40
R124	C124	L248	L251	5/27/2003	8:22	P	5	1	8:17	8:27
R125	C125	L249	L252	5/27/2003	9:03	D	5	1	8:58	9:08
R126	C126	L250	L253	5/27/2003	8:42	P	5	0	8:37	8:47
R127	C127	L254	L257	5/27/2003	9:29	P	5	1	9:23	9:33
R128	C128	L255	L258	5/27/2003	10:15	D	5	1	10:09	10:19
R129	C129	L256	L259	5/27/2003	9:53	P	5	0	9:47	9:57
R130	C130	L260	L263	5/27/2003	11:07	D	5	1	11:01	11:11
R131	C131	L261	L264	5/27/2003	11:20	D	5	1	11:14	11:24
R132	C132	L262	L265	5/27/2003	10:59	P	5	0	10:54	11:04
R133	C133	L266	L269	5/27/2003	12:24	D	5	1	12:19	12:29
R134	C134	L267	L270	5/27/2003	12:39	D	5	1	12:33	12:43
R135	C135	L268	L271	5/27/2003	12:13	P	5	0	12:07	12:17
R136	C136	L272	L275	5/27/2003	7:24	D	5	1	7:18	7:28
R137	C137	L273	L276	5/27/2003	7:31	D	5	1	7:26	7:36
R138	C138	L274	L277	5/27/2003	7:38	D	5	0	7:33	7:43
R139	C139	L278	L281	5/27/2003	8:28	D	5	1	8:23	8:33
R140	C140	L279	L282	5/27/2003	7:59	P	5	1	7:53	8:03
R141	C141	L280	L283	5/27/2003	8:56	D	5	0	8:51	9:01
R142	C142	L284	L287	5/27/2003	9:37	D	5	1	9:32	9:42
R143	C143	L285	L288	5/27/2003	9:16	P	5	1	9:10	9:20
R144	C144	L286	L289	5/27/2003	9:58	D	5	0	9:52	10:02
R145	C145	L290	L293	5/27/2003	10:42	D	5	1	10:37	10:47
R146	C146	L291	L294	5/27/2003	10:52	D	5	1	10:46	10:56
R147	C147	L292	L295	5/27/2003	11:07	D	5	0	11:02	11:12
R148	C148	L296	L299	5/27/2003	11:47	D	5	1	11:42	11:52
R149	C149	L297	L300	5/27/2003	11:32	P	5	1	11:27	11:37
R150	C150	L298	L301	5/27/2003	11:39	P	5	0	11:33	11:43

Appendix E

U150B Data Set

Table E.1: U150B Model Table

MRT	FIXED	60	TW	VARY	WAIT	30	SPEED	20	ROUTE	480

Table E.2: U150B Vehicle Table

ID	Capacity	Wheelchairs
V01	4	2
V02	4	2
V03	4	2
V04	4	2
V05	4	2
V06	4	2
V07	4	2
V08	4	2
V09	4	2
V10	4	2

Table E.3: U150B Client Table

ID	Name	Address	Wheelchair	ID	Name	Address	Wheelchair
C001	“name1”	null	n	C076	“name76”	null	n
C002	“name2”	null	n	C077	“name77”	null	n
C003	“name3”	null	y	C078	“name78”	null	y
C004	“name4”	null	n	C079	“name79”	null	n
C005	“name5”	null	n	C080	“name80”	null	n
C006	“name6”	null	y	C081	“name81”	null	y
C007	“name7”	null	n	C082	“name82”	null	n
C008	“name8”	null	n	C083	“name83”	null	n
C009	“name9”	null	y	C084	“name84”	null	y
C010	“name10”	null	n	C085	“name85”	null	n
C011	“name11”	null	n	C086	“name86”	null	n
C012	“name12”	null	y	C087	“name87”	null	y
C013	“name13”	null	n	C088	“name88”	null	n
C014	“name14”	null	n	C089	“name89”	null	n
C015	“name15”	null	y	C090	“name90”	null	y
C016	“name16”	null	n	C091	“name91”	null	n
C017	“name17”	null	n	C092	“name92”	null	n
C018	“name18”	null	y	C093	“name93”	null	y
C019	“name19”	null	n	C094	“name94”	null	n
C020	“name20”	null	n	C095	“name95”	null	n
C021	“name21”	null	y	C096	“name96”	null	y
C022	“name22”	null	n	C097	“name97”	null	n
C023	“name23”	null	n	C098	“name98”	null	n
C024	“name24”	null	y	C099	“name99”	null	y
C025	“name25”	null	n	C100	“name100”	null	n
C026	“name26”	null	n	C101	“name101”	null	n
C027	“name27”	null	y	C102	“name102”	null	y
C028	“name28”	null	n	C103	“name103”	null	n
C029	“name29”	null	n	C104	“name104”	null	n
C030	“name30”	null	y	C105	“name105”	null	y
C031	“name31”	null	n	C106	“name106”	null	n
C032	“name32”	null	n	C107	“name107”	null	n
C033	“name33”	null	y	C108	“name108”	null	y
C034	“name34”	null	n	C109	“name109”	null	n
C035	“name35”	null	n	C110	“name110”	null	n
C036	“name36”	null	y	C111	“name111”	null	y
C037	“name37”	null	n	C112	“name112”	null	n
C038	“name38”	null	n	C113	“name113”	null	n
C039	“name39”	null	y	C114	“name114”	null	y
C040	“name40”	null	n	C115	“name115”	null	n
C041	“name41”	null	n	C116	“name116”	null	n
C042	“name42”	null	y	C117	“name117”	null	y
C043	“name43”	null	n	C118	“name118”	null	n
C044	“name44”	null	n	C119	“name119”	null	n
C045	“name45”	null	y	C120	“name120”	null	y
C046	“name46”	null	n	C121	“name121”	null	n
C047	“name47”	null	n	C122	“name122”	null	n
C048	“name48”	null	y	C123	“name123”	null	y
C049	“name49”	null	n	C124	“name124”	null	n
C050	“name50”	null	n	C125	“name125”	null	n
C051	“name51”	null	y	C126	“name126”	null	y
C052	“name52”	null	n	C127	“name127”	null	n
C053	“name53”	null	n	C128	“name128”	null	n
C054	“name54”	null	y	C129	“name129”	null	y
C055	“name55”	null	n	C130	“name130”	null	n
C056	“name56”	null	n	C131	“name131”	null	n
C057	“name57”	null	y	C132	“name132”	null	y
C058	“name58”	null	n	C133	“name133”	null	n
C059	“name59”	null	n	C134	“name134”	null	n
C060	“name60”	null	y	C135	“name135”	null	y
C061	“name61”	null	n	C136	“name136”	null	n
C062	“name62”	null	n	C137	“name137”	null	n
C063	“name63”	null	y	C138	“name138”	null	y
C064	“name64”	null	n	C139	“name139”	null	n
C065	“name65”	null	n	C140	“name140”	null	n
C066	“name66”	null	y	C141	“name141”	null	y
C067	“name67”	null	n	C142	“name142”	null	n
C068	“name68”	null	n	C143	“name143”	null	n
C069	“name69”	null	y	C144	“name144”	null	y
C070	“name70”	null	n	C145	“name145”	null	n
C071	“name71”	null	n	C146	“name146”	null	n
C072	“name72”	null	y	C147	“name147”	null	y
C073	“name73”	null	n	C148	“name148”	null	n
C074	“name74”	null	n	C149	“name149”	null	n
C075	“name75”	null	y	C150	“name150”	null	y

Table E.4: U150B Location Table

ID	Address	x-coordinate	y-coordinate
L001	“depot”	0	0
L002	“addr1”	-1.45972328	-0.296940752
L003	“addr2”	-1.897003414	0.740732955
L004	“addr3”	0.309694716	1.547179731
L005	“addr4”	0.804828713	1.488904837
L006	“addr5”	-1.698943743	0.955019638
L007	“addr6”	1.755570238	1.720248611
L008	“addr7”	1.90149201	-1.588725706
L009	“addr8”	1.048930169	1.509285719
L010	“addr9”	0.743853581	0.300921028
L011	“addr10”	-0.227706456	-0.993722886
L012	“addr11”	-1.076630451	-1.951668675
L013	“addr12”	-0.509492665	-1.522486725
L014	“addr13”	0.121760874	-0.364691838
L015	“addr14”	-0.434346043	-0.74879721
L016	“addr15”	-1.85080782	-0.349973747
L017	“addr16”	-0.667046869	-1.753674088
L018	“addr17”	-1.909785413	0.975884201
L019	“addr18”	0.641215576	0.031354537
L020	“addr19”	0.540640335	1.811308205
L021	“addr20”	1.379552101	1.254266497
L022	“addr21”	1.348067121	-0.081421943
L023	“addr22”	0.350323731	0.80481935
L024	“addr23”	-0.18044449	1.831780279
L025	“addr24”	-1.788400347	-0.785425761
L026	“addr25”	1.535738209	-1.788981206
L027	“addr26”	-1.799329443	-1.430304791
L028	“addr27”	-0.28857119	1.612050868
L029	“addr28”	-0.036553811	1.606079643
L030	“addr29”	1.468571014	-0.352456196
L031	“addr30”	0.052847947	-0.324092431
L032	“addr31”	0.438156833	-1.151820607
L033	“addr32”	-0.477411365	1.919225387
L034	“addr33”	0.783519386	-1.511134628
L035	“addr34”	0.531010469	0.243716728
L036	“addr35”	-1.96181005	-0.772655821
L037	“addr36”	1.982937383	-1.38966072
L038	“addr37”	-0.922711357	-1.390480189
L039	“addr38”	-1.02422428	0.642038679
L040	“addr39”	1.092198613	-1.672144645
L041	“addr40”	0.179380663	-1.282714798
L042	“addr41”	-0.385941913	-1.42377451
L043	“addr42”	1.267485523	0.479705255
L044	“addr43”	-1.56761833	-1.073361743
L045	“addr44”	-0.86822634	1.08961461
L046	“addr45”	1.680907148	-1.139726641
L047	“addr46”	0.847953316	1.562789656
L048	“addr47”	0.41354284	-1.582686957
L049	“addr48”	1.226476031	1.186345727
L050	“addr49”	-1.330927233	0.38707632
L051	“addr50”	1.268823314	-0.728879408
L052	“addr51”	1.324059256	-1.958005318
L053	“addr52”	0.963563401	0.394494848
L054	“addr53”	-0.04083911	0.626001632
L055	“addr54”	-0.405334306	-0.192888859
L056	“addr55”	-0.485131768	-1.360061268
L057	“addr56”	-0.902162489	-0.178515921
L058	“addr57”	-0.311561145	-1.5763889
L059	“addr58”	-1.011067069	1.770027204
L060	“addr59”	0.824595546	0.616157941
L061	“addr60”	-0.998239372	-0.485448827
L062	“addr61”	0.478332146	-0.216565177
L063	“addr62”	-0.189096753	1.508820376
L064	“addr63”	-0.121047372	-1.166865833
L065	“addr64”	1.07843817	1.060286887
L066	“addr65”	1.200251947	1.795403526
L067	“addr66”	-0.448265955	-1.29334998
L068	“addr67”	-1.157474646	0.128751581
L069	“addr68”	0.052775634	0.517423345
L070	“addr69”	-1.142122767	0.918063187
L071	“addr70”	0.937753364	-0.349780004
L072	“addr71”	0.201080344	0.163575182
L073	“addr72”	-1.03106582	-0.363999885
L074	“addr73”	-0.335983128	0.81669856
L075	“addr74”	0.529111005	0.637054859

Table E.4:U150B Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L076	“addr75”	-0.847484889	1.300599913
L077	“addr76”	-1.2777899	0.316031862
L078	“addr77”	-0.609822554	-0.03951012
L079	“addr78”	-0.858082867	-1.026159596
L080	“addr79”	-1.815426006	-1.779404182
L081	“addr80”	-1.046537159	0.258803017
L082	“addr81”	-1.933039544	0.181827117
L083	“addr82”	0.421991798	0.018953949
L084	“addr83”	-0.582542	-1.023826354
L085	“addr84”	-0.865782499	0.763914603
L086	“addr85”	-0.886584921	-1.599091655
L087	“addr86”	0.898350678	1.722308488
L088	“addr87”	-0.710548071	-1.812303489
L089	“addr88”	1.67056531	-1.374792161
L090	“addr89”	-0.026529314	0.718745744
L091	“addr90”	-0.155447625	-0.884362981
L092	“addr91”	-0.836306519	0.084161661
L093	“addr92”	-0.146011363	-0.92131076
L094	“addr93”	-1.662425563	-0.918295066
L095	“addr94”	-1.723967996	0.206340684
L096	“addr95”	-1.565789317	-1.907821506
L097	“addr96”	-1.512765582	-0.244416008
L098	“addr97”	-0.97234043	-0.279297623
L099	“addr98”	1.411545773	1.728026755
L100	“addr99”	1.7093517	-1.323832918
L101	“addr100”	0.013361747	0.593639549
L102	“addr101”	-0.376432255	1.568186452
L103	“addr102”	-0.07314753	-1.799138924
L104	“addr103”	0.650731784	-1.867097728
L105	“addr104”	1.976087672	-0.3401923
L106	“addr105”	1.909052171	-0.143974154
L107	“addr106”	-1.671244099	-1.245140628
L108	“addr107”	0.04582361	-1.956471237
L109	“addr108”	0.639557807	-0.16257227
L110	“addr109”	1.215485872	-0.902567551
L111	“addr110”	-0.222366461	-0.83075574
L112	“addr111”	-1.307685714	-1.913828477
L113	“addr112”	0.595664868	0.33103646
L114	“addr113”	-1.258600436	0.656041095
L115	“addr114”	1.686576943	1.580990257
L116	“addr115”	0.672355677	1.912267441
L117	“addr116”	-0.812958638	-0.058068852
L118	“addr117”	-1.074025087	-0.894726673
L119	“addr118”	1.425268145	-0.684315686
L120	“addr119”	1.5948319	0.789741806
L121	“addr120”	-0.426148757	0.067608083
L122	“addr121”	-1.045382147	-1.418890721
L123	“addr122”	0.456717373	1.293824963
L124	“addr123”	-0.616230605	0.716156756
L125	“addr124”	-0.727494767	0.378204386
L126	“addr125”	-1.525263315	1.331392312
L127	“addr126”	0.234987464	-0.534313072
L128	“addr127”	0.19101514	-0.295045734
L129	“addr128”	1.994425067	1.123065513
L130	“addr129”	-1.400149009	1.664792572
L131	“addr130”	0.192029056	-0.272125618
L132	“addr131”	-0.988693705	-1.052079809
L133	“addr132”	1.61298141	-0.979434051
L134	“addr133”	0.280154721	-0.497446894
L135	“addr134”	1.141752482	1.774688586
L136	“addr135”	-0.42214128	0.556994566
L137	“addr136”	0.086730649	0.684305912
L138	“addr137”	1.117660015	0.736144561
L139	“addr138”	-1.570538816	-1.66256419
L140	“addr139”	-0.883402729	1.588551817
L141	“addr140”	-1.832924111	-1.461189487
L142	“addr141”	-1.295954386	0.59395279
L143	“addr142”	-1.845229223	1.264798857
L144	“addr143”	0.537153662	-0.946723066
L145	“addr144”	-1.281523921	-0.822144523
L146	“addr145”	1.559335852	-0.779875003
L147	“addr146”	-1.203860154	-1.102601515
L148	“addr147”	-0.36124241	1.614762711
L149	“addr148”	-0.520670436	-0.881454641
L150	“addr149”	1.749534485	-0.331301153

Table E.4:U150B Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L151	“addr150”	1.565137522	-1.810330863
L152	“addr151”	-0.090855454	0.679952247
L153	“addr152”	1.091656152	-0.095731722
L154	“addr153”	0.261338232	-0.483860113
L155	“addr154”	1.818571477	-1.96782576
L156	“addr155”	0.494606752	-1.415447634
L157	“addr156”	1.227703244	-0.490347237
L158	“addr157”	1.212056224	0.295412097
L159	“addr158”	-1.143397128	1.227485691
L160	“addr159”	-0.080688302	0.576542601
L161	“addr160”	-1.358399679	-1.999348691
L162	“addr161”	-0.027141101	-0.045293276
L163	“addr162”	-1.565027522	1.572753112
L164	“addr163”	-0.867987106	-1.699095109
L165	“addr164”	-0.02581198	0.40287651
L166	“addr165”	-0.7165182	-1.456045952
L167	“addr166”	-0.680536952	-0.49791258
L168	“addr167”	0.458278472	1.488699379
L169	“addr168”	-0.860854761	0.731313433
L170	“addr169”	0.436510266	0.332244046
L171	“addr170”	0.210723131	0.741884
L172	“addr171”	0.036362474	-0.446389685
L173	“addr172”	0.118540762	0.134534308
L174	“addr173”	1.609368559	0.559279442
L175	“addr174”	0.367577092	1.432209838
L176	“addr175”	0.314843524	-0.200912192
L177	“addr176”	-1.004067013	1.106540072
L178	“addr177”	0.422138842	-1.97601274
L179	“addr178”	0.256946669	0.167380582
L180	“addr179”	-1.809844372	1.939210802
L181	“addr180”	-1.944855755	0.71935409
L182	“addr181”	-1.200663509	-0.286586062
L183	“addr182”	1.558508442	-1.591147723
L184	“addr183”	1.362137817	1.63639501
L185	“addr184”	1.772833525	-1.170507614
L186	“addr185”	1.213763693	0.67502796
L187	“addr186”	-1.14778664	-0.276448379
L188	“addr187”	-1.514274993	0.759072096
L189	“addr188”	1.532992836	-1.788563733
L190	“addr189”	-1.609430453	-0.664443613
L191	“addr190”	0.316791648	-0.699228277
L192	“addr191”	1.696907227	-0.93753575
L193	“addr192”	0.891492955	-1.42876823
L194	“addr193”	1.549639734	1.469479163
L195	“addr194”	-0.232630252	0.200756773
L196	“addr195”	1.091081256	-1.004420318
L197	“addr196”	-1.781197498	-0.606659231
L198	“addr197”	1.264441146	-0.979693334
L199	“addr198”	-0.639676391	1.788100897
L200	“addr199”	-0.26808049	0.337553626
L201	“addr200”	0.54900613	-0.426130431
L202	“addr201”	-1.984553294	-1.022234423
L203	“addr202”	-0.706507215	-1.196912682
L204	“addr203”	-1.231117852	-1.633539811
L205	“addr204”	-0.536753616	-1.206701214
L206	“addr205”	0.516265327	-1.485981957
L207	“addr206”	-0.170119097	-1.475908891
L208	“addr207”	-0.758503723	0.263719766
L209	“addr208”	0.835714531	-1.418158972
L210	“addr209”	0.890602032	-0.417961526
L211	“addr210”	1.972968461	0.474171856
L212	“addr211”	0.148527815	-0.542772545
L213	“addr212”	1.270315105	-1.969450464
L214	“addr213”	0.778172739	1.179274564
L215	“addr214”	-1.878239491	0.124980742
L216	“addr215”	1.360170949	0.606326684
L217	“addr216”	1.056129146	0.101304942
L218	“addr217”	-0.552300226	0.479715611
L219	“addr218”	1.033696705	-0.948893195
L220	“addr219”	0.951380181	0.931753996
L221	“addr220”	0.469801868	1.488802498
L222	“addr221”	-0.004888311	-0.962505779
L223	“addr222”	0.423547034	0.987988138
L224	“addr223”	1.534470097	-1.406320478
L225	“addr224”	-0.747021735	-0.012331984

Table E.4:U150B Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L226	“addr225”	-1.294926166	-0.734853483
L227	“addr226”	1.526419585	-1.753487831
L228	“addr227”	1.169107712	-0.179032491
L229	“addr228”	0.092913375	-1.68217597
L230	“addr229”	-0.346026154	1.389655541
L231	“addr230”	0.953004051	1.937430179
L232	“addr231”	1.427821602	1.494783567
L233	“addr232”	1.934380572	0.361234328
L234	“addr233”	1.632486713	0.089402351
L235	“addr234”	0.165176378	-1.541287956
L236	“addr235”	-1.804844856	-1.110328363
L237	“addr236”	1.951004622	0.76314945
L238	“addr237”	1.465845596	1.015150956
L239	“addr238”	1.900090692	-0.301035141
L240	“addr239”	1.778586024	1.275227624
L241	“addr240”	-1.995857446	1.850979863
L242	“addr241”	-1.190057494	-1.601808984
L243	“addr242”	-0.12911836	-0.589959254
L244	“addr243”	1.909338586	-0.918507333
L245	“addr244”	0.417439346	0.823920082
L246	“addr245”	0.501383822	-0.051236038
L247	“addr246”	1.072103477	1.201960018
L248	“addr247”	-1.788785748	-0.668806801
L249	“addr248”	-1.104818252	-0.83354857
L250	“addr249”	-0.460374047	-1.752102865
L251	“addr250”	1.018030263	0.613272851
L252	“addr251”	1.715564412	-0.958049326
L253	“addr252”	-0.5874241	-0.761155226
L254	“addr253”	0.77931661	0.835771508
L255	“addr254”	1.241942692	0.709986186
L256	“addr255”	-1.999470089	-0.698412026
L257	“addr256”	-1.093925638	1.997791788
L258	“addr257”	-1.589034588	0.19974722
L259	“addr258”	-0.615474105	1.666179336
L260	“addr259”	-0.467325163	-0.28034419
L261	“addr260”	-1.408123309	0.800287142
L262	“addr261”	-0.691930067	1.865172193
L263	“addr262”	-0.800496122	1.779434767
L264	“addr263”	-0.210863913	1.958304915
L265	“addr264”	1.74072838	-0.756111445
L266	“addr265”	-0.681178234	1.792099478
L267	“addr266”	0.80463527	0.374207857
L268	“addr267”	-1.261582897	0.045629523
L269	“addr268”	-1.262855525	1.996477527
L270	“addr269”	0.580517782	1.281037797
L271	“addr270”	0.506194943	-1.953738606
L272	“addr271”	-1.742052062	0.174605736
L273	“addr272”	-0.748818787	0.568648608
L274	“addr273”	-1.065198494	0.590859335
L275	“addr274”	-1.474932488	-1.298711038
L276	“addr275”	-0.933461364	1.690751223
L277	“addr276”	0.350882035	0.981849432
L278	“addr277”	-0.478141087	1.471991801
L279	“addr278”	1.952577075	-1.159146212
L280	“addr279”	-0.250150139	-0.028369951
L281	“addr280”	0.383522361	1.285398002
L282	“addr281”	-0.368304502	1.818018642
L283	“addr282”	-0.735822026	-1.45123398
L284	“addr283”	-0.542993713	0.901328981
L285	“addr284”	1.453777772	-0.12449791
L286	“addr285”	-1.10938277	-1.361188628
L287	“addr286”	-0.280867951	1.631025219
L288	“addr287”	-1.153522401	1.643028637
L289	“addr288”	0.3269417	0.569405711
L290	“addr289”	1.861042341	0.558321564
L291	“addr290”	-0.025702703	1.362451617
L292	“addr291”	-1.140648516	1.412796204
L293	“addr292”	0.585316157	1.387786931
L294	“addr293”	-0.867231991	0.040154954
L295	“addr294”	1.240153103	1.293132518
L296	“addr295”	1.898125429	-1.042277114
L297	“addr296”	-0.134097888	-1.001466467
L298	“addr297”	1.983816945	0.544054259
L299	“addr298”	1.307185894	1.567807617
L300	“addr299”	0.989139655	-1.558466991
L301	“addr300”	-0.992487034	1.775019694

Table E.5: U150B Request Table

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R001	C001	L002	L005	5/27/2003	7:04	P	5	1	6:54	7:14
R002	C002	L003	L006	5/27/2003	7:44	D	5	1	7:34	7:54
R003	C003	L004	L007	5/27/2003	8:00	D	5	0	7:49	8:09
R004	C004	L008	L011	5/27/2003	8:15	P	5	1	8:04	8:24
R005	C005	L009	L012	5/27/2003	8:57	D	5	1	8:46	9:06
R006	C006	L010	L013	5/27/2003	8:38	P	5	0	8:28	8:48
R007	C007	L014	L017	5/27/2003	9:13	P	5	1	9:02	9:22
R008	C008	L015	L018	5/27/2003	9:54	D	5	1	9:43	10:03
R009	C009	L016	L019	5/27/2003	9:29	P	5	0	9:19	9:39
R010	C010	L020	L023	5/27/2003	10:17	P	5	1	10:07	10:27
R011	C011	L021	L024	5/27/2003	10:25	P	5	1	10:15	10:35
R012	C012	L022	L025	5/27/2003	11:06	D	5	0	10:55	11:15
R013	C013	L026	L029	5/27/2003	11:57	D	5	1	11:47	12:07
R014	C014	L027	L030	5/27/2003	11:36	P	5	1	11:26	11:46
R015	C015	L028	L031	5/27/2003	12:19	D	5	0	12:09	12:29
R016	C016	L032	L035	5/27/2003	7:45	D	5	1	7:34	7:54
R017	C017	L033	L036	5/27/2003	7:58	D	5	1	7:47	8:07
R018	C018	L034	L037	5/27/2003	8:15	D	5	0	8:04	8:24
R019	C019	L038	L041	5/27/2003	8:28	P	5	1	8:18	8:38
R020	C020	L039	L042	5/27/2003	8:39	P	5	1	8:29	8:49
R021	C021	L040	L043	5/27/2003	8:54	P	5	0	8:43	9:03
R022	C022	L044	L047	5/27/2003	10:16	D	5	1	10:06	10:26
R023	C023	L045	L048	5/27/2003	10:31	D	5	1	10:20	10:40
R024	C024	L046	L049	5/27/2003	10:03	P	5	0	9:52	10:12
R025	C025	L050	L053	5/27/2003	11:32	D	5	1	11:21	11:41
R026	C026	L051	L054	5/27/2003	11:11	P	5	1	11:01	11:21
R027	C027	L052	L055	5/27/2003	11:20	P	5	0	11:09	11:29
R028	C028	L056	L059	5/27/2003	12:30	D	5	1	12:19	12:39
R029	C029	L057	L060	5/27/2003	12:41	D	5	1	12:31	12:51
R030	C030	L058	L061	5/27/2003	12:53	D	5	0	12:42	13:02
R031	C031	L062	L065	5/27/2003	7:38	D	5	1	7:27	7:47
R032	C032	L063	L066	5/27/2003	7:45	D	5	1	7:34	7:54
R033	C033	L064	L067	5/27/2003	7:25	P	5	0	7:15	7:35
R034	C034	L068	L071	5/27/2003	8:10	P	5	1	8:00	8:20
R035	C035	L069	L072	5/27/2003	8:19	P	5	1	8:09	8:29
R036	C036	L070	L073	5/27/2003	8:28	P	5	0	8:17	8:37
R037	C037	L074	L077	5/27/2003	9:31	D	5	1	9:21	9:41
R038	C038	L075	L078	5/27/2003	9:39	D	5	1	9:28	9:48
R039	C039	L076	L079	5/27/2003	9:47	D	5	0	9:36	9:56
R040	C040	L080	L083	5/27/2003	10:27	D	5	1	10:16	10:36
R041	C041	L081	L084	5/27/2003	10:36	D	5	1	10:26	10:46
R042	C042	L082	L085	5/27/2003	10:15	P	5	0	10:04	10:24
R043	C043	L086	L089	5/27/2003	11:44	D	5	1	11:33	11:53
R044	C044	L087	L090	5/27/2003	11:15	P	5	1	11:04	11:24
R045	C045	L088	L091	5/27/2003	11:32	P	5	0	11:21	11:41
R046	C046	L092	L095	5/27/2003	7:03	P	5	1	6:52	7:12
R047	C047	L093	L096	5/27/2003	7:11	P	5	1	7:01	7:21
R048	C048	L094	L097	5/27/2003	7:50	D	5	0	7:40	8:00
R049	C049	L098	L101	5/27/2003	8:38	D	5	1	8:28	8:48
R050	C050	L099	L102	5/27/2003	8:46	D	5	1	8:36	8:56
R051	C051	L100	L103	5/27/2003	9:02	D	5	0	8:51	9:11
R052	C052	L104	L107	5/27/2003	9:09	P	5	1	8:58	9:18
R053	C053	L105	L108	5/27/2003	9:52	D	5	1	9:42	10:02
R054	C054	L106	L109	5/27/2003	10:03	D	5	0	9:52	10:12
R055	C055	L110	L113	5/27/2003	10:11	P	5	1	10:00	10:20
R056	C056	L111	L114	5/27/2003	10:54	D	5	1	10:44	11:04
R057	C057	L112	L115	5/27/2003	11:08	D	5	0	10:58	11:18
R058	C058	L116	L119	5/27/2003	11:17	P	5	1	11:06	11:26
R059	C059	L117	L120	5/27/2003	11:59	D	5	1	11:48	12:08
R060	C060	L118	L121	5/27/2003	11:37	P	5	0	11:26	11:46
R061	C061	L122	L125	5/27/2003	7:34	D	5	1	7:24	7:44
R062	C062	L123	L126	5/27/2003	7:20	P	5	1	7:09	7:29
R063	C063	L124	L127	5/27/2003	7:28	P	5	0	7:18	7:38
R064	C064	L128	L131	5/27/2003	8:41	D	5	1	8:31	8:51
R065	C065	L129	L132	5/27/2003	8:13	P	5	1	8:03	8:23
R066	C066	L130	L133	5/27/2003	8:29	P	5	0	8:18	8:38
R067	C067	L134	L137	5/27/2003	9:42	D	5	1	9:32	9:52
R068	C068	L135	L138	5/27/2003	9:50	D	5	1	9:40	10:00
R069	C069	L136	L139	5/27/2003	10:06	D	5	0	9:56	10:16
R070	C070	L140	L143	5/27/2003	10:55	D	5	1	10:44	11:04
R071	C071	L141	L144	5/27/2003	10:36	P	5	1	10:25	10:45
R072	C072	L142	L145	5/27/2003	10:47	P	5	0	10:37	10:57
R073	C073	L146	L149	5/27/2003	12:13	D	5	1	12:02	12:22
R074	C074	L147	L150	5/27/2003	11:47	P	5	1	11:36	11:56
R075	C075	L148	L151	5/27/2003	12:34	D	5	0	12:24	12:44

Table E.5: U150B Request Table *(continued)*

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R076	C076	L152	L155	5/27/2003	7:02	P	5	1	6:52	7:12
R077	C077	L153	L156	5/27/2003	7:40	D	5	1	7:29	7:49
R078	C078	L154	L157	5/27/2003	7:48	D	5	0	7:38	7:58
R079	C079	L158	L161	5/27/2003	7:56	P	5	1	7:45	8:05
R080	C080	L159	L162	5/27/2003	8:43	D	5	1	8:32	8:52
R081	C081	L160	L163	5/27/2003	8:54	D	5	0	8:44	9:04
R082	C082	L164	L167	5/27/2003	9:09	P	5	1	8:59	9:19
R083	C083	L165	L168	5/27/2003	9:52	D	5	1	9:41	10:01
R084	C084	L166	L169	5/27/2003	9:32	P	5	0	9:22	9:42
R085	C085	L170	L173	5/27/2003	10:11	P	5	1	10:00	10:20
R086	C086	L171	L174	5/27/2003	10:42	D	5	1	10:32	10:52
R087	C087	L172	L175	5/27/2003	10:52	D	5	0	10:41	11:01
R088	C088	L176	L179	5/27/2003	11:01	P	5	1	10:51	11:11
R089	C089	L177	L180	5/27/2003	11:52	D	5	1	11:41	12:01
R090	C090	L178	L181	5/27/2003	12:01	D	5	0	11:50	12:10
R091	C091	L182	L185	5/27/2003	7:46	D	5	1	7:36	7:56
R092	C092	L183	L186	5/27/2003	7:57	D	5	1	7:46	8:06
R093	C093	L184	L187	5/27/2003	8:09	D	5	0	7:59	8:19
R094	C094	L188	L191	5/27/2003	8:18	P	5	1	8:07	8:27
R095	C095	L189	L192	5/27/2003	8:35	P	5	1	8:24	8:44
R096	C096	L190	L193	5/27/2003	8:50	P	5	0	8:39	8:59
R097	C097	L194	L197	5/27/2003	10:07	D	5	1	9:57	10:17
R098	C098	L195	L198	5/27/2003	10:21	D	5	1	10:11	10:31
R099	C099	L196	L199	5/27/2003	9:53	P	5	0	9:43	10:03
R100	C100	L200	L203	5/27/2003	11:16	D	5	1	11:05	11:25
R101	C101	L201	L204	5/27/2003	10:54	P	5	1	10:44	11:04
R102	C102	L202	L205	5/27/2003	11:07	P	5	0	10:57	11:17
R103	C103	L206	L209	5/27/2003	11:39	P	5	1	11:28	11:48
R104	C104	L207	L210	5/27/2003	11:46	P	5	1	11:35	11:55
R105	C105	L208	L211	5/27/2003	12:25	D	5	0	12:15	12:35
R106	C106	L212	L215	5/27/2003	7:02	P	5	1	6:51	7:11
R107	C107	L213	L216	5/27/2003	7:55	D	5	1	7:45	8:05
R108	C108	L214	L217	5/27/2003	7:27	P	5	0	7:16	7:36
R109	C109	L218	L221	5/27/2003	8:41	D	5	1	8:31	8:51
R110	C110	L219	L222	5/27/2003	8:23	P	5	1	8:13	8:33
R111	C111	L220	L223	5/27/2003	9:05	D	5	0	8:54	9:14
R112	C112	L224	L227	5/27/2003	9:52	D	5	1	9:42	10:02
R113	C113	L225	L228	5/27/2003	10:02	D	5	1	9:52	10:12
R114	C114	L226	L229	5/27/2003	9:38	P	5	0	9:28	9:48
R115	C115	L230	L233	5/27/2003	10:27	P	5	1	10:16	10:36
R116	C116	L231	L234	5/27/2003	10:36	P	5	1	10:26	10:46
R117	C117	L232	L235	5/27/2003	11:10	D	5	0	10:59	11:19
R118	C118	L236	L239	5/27/2003	11:54	D	5	1	11:44	12:04
R119	C119	L237	L240	5/27/2003	12:04	D	5	1	11:53	12:13
R120	C120	L238	L241	5/27/2003	11:45	P	5	0	11:34	11:54
R121	C121	L242	L245	5/27/2003	7:06	P	5	1	6:55	7:15
R122	C122	L243	L246	5/27/2003	7:46	D	5	1	7:36	7:56
R123	C123	L244	L247	5/27/2003	7:55	D	5	0	7:45	8:05
R124	C124	L248	L251	5/27/2003	8:39	D	5	1	8:29	8:49
R125	C125	L249	L252	5/27/2003	8:18	P	5	1	8:07	8:27
R126	C126	L250	L253	5/27/2003	9:01	D	5	0	8:51	9:11
R127	C127	L254	L257	5/27/2003	9:48	D	5	1	9:38	9:58
R128	C128	L255	L258	5/27/2003	9:59	D	5	1	9:48	10:08
R129	C129	L256	L259	5/27/2003	9:35	P	5	0	9:24	9:44
R130	C130	L260	L263	5/27/2003	10:44	D	5	1	10:33	10:53
R131	C131	L261	L264	5/27/2003	10:29	P	5	1	10:19	10:39
R132	C132	L262	L265	5/27/2003	10:38	P	5	0	10:28	10:48
R133	C133	L266	L269	5/27/2003	11:21	P	5	1	11:11	11:31
R134	C134	L267	L270	5/27/2003	12:05	D	5	1	11:55	12:15
R135	C135	L268	L271	5/27/2003	11:43	P	5	0	11:33	11:53
R136	C136	L272	L275	5/27/2003	7:30	D	5	1	7:20	7:40
R137	C137	L273	L276	5/27/2003	7:44	D	5	1	7:34	7:54
R138	C138	L274	L277	5/27/2003	7:19	P	5	0	7:09	7:29
R139	C139	L278	L281	5/27/2003	8:39	D	5	1	8:29	8:49
R140	C140	L279	L282	5/27/2003	8:47	D	5	1	8:36	8:56
R141	C141	L280	L283	5/27/2003	9:02	D	5	0	8:51	9:11
R142	C142	L284	L287	5/27/2003	9:53	D	5	1	9:43	10:03
R143	C143	L285	L288	5/27/2003	9:26	P	5	1	9:15	9:35
R144	C144	L286	L289	5/27/2003	10:12	D	5	0	10:01	10:21
R145	C145	L290	L293	5/27/2003	10:21	P	5	1	10:11	10:31
R146	C146	L291	L294	5/27/2003	11:02	D	5	1	10:51	11:11
R147	C147	L292	L295	5/27/2003	11:14	D	5	0	11:04	11:24
R148	C148	L296	L299	5/27/2003	11:59	D	5	1	11:49	12:09
R149	C149	L297	L300	5/27/2003	11:38	P	5	1	11:27	11:47
R150	C150	L298	L301	5/27/2003	11:50	P	5	0	11:40	12:00

Appendix F

U150C Data Set

Table F.1: U150C Model Table

MRT	FIXED	60	TW	VARY	WAIT	30	SPEED	20	ROUTE	480

Table F.2: U150C Vehicle Table

ID	Capacity	Wheelchairs
V01	4	2
V02	4	2
V03	4	2
V04	4	2
V05	4	2
V06	4	2
V07	4	2
V08	4	2
V09	4	2
V10	4	2

Table F.3: U150C Client Table

ID	Name	Address	Wheelchair	ID	Name	Address	Wheelchair
C001	“name1”	null	n	C076	“name76”	null	n
C002	“name2”	null	n	C077	“name77”	null	n
C003	“name3”	null	y	C078	“name78”	null	y
C004	“name4”	null	n	C079	“name79”	null	n
C005	“name5”	null	n	C080	“name80”	null	n
C006	“name6”	null	y	C081	“name81”	null	y
C007	“name7”	null	n	C082	“name82”	null	n
C008	“name8”	null	n	C083	“name83”	null	n
C009	“name9”	null	y	C084	“name84”	null	y
C010	“name10”	null	n	C085	“name85”	null	n
C011	“name11”	null	n	C086	“name86”	null	n
C012	“name12”	null	y	C087	“name87”	null	y
C013	“name13”	null	n	C088	“name88”	null	n
C014	“name14”	null	n	C089	“name89”	null	n
C015	“name15”	null	y	C090	“name90”	null	y
C016	“name16”	null	n	C091	“name91”	null	n
C017	“name17”	null	n	C092	“name92”	null	n
C018	“name18”	null	y	C093	“name93”	null	y
C019	“name19”	null	n	C094	“name94”	null	n
C020	“name20”	null	n	C095	“name95”	null	n
C021	“name21”	null	y	C096	“name96”	null	y
C022	“name22”	null	n	C097	“name97”	null	n
C023	“name23”	null	n	C098	“name98”	null	n
C024	“name24”	null	y	C099	“name99”	null	y
C025	“name25”	null	n	C100	“name100”	null	n
C026	“name26”	null	n	C101	“name101”	null	n
C027	“name27”	null	y	C102	“name102”	null	y
C028	“name28”	null	n	C103	“name103”	null	n
C029	“name29”	null	n	C104	“name104”	null	n
C030	“name30”	null	y	C105	“name105”	null	y
C031	“name31”	null	n	C106	“name106”	null	n
C032	“name32”	null	n	C107	“name107”	null	n
C033	“name33”	null	y	C108	“name108”	null	y
C034	“name34”	null	n	C109	“name109”	null	n
C035	“name35”	null	n	C110	“name110”	null	n
C036	“name36”	null	y	C111	“name111”	null	y
C037	“name37”	null	n	C112	“name112”	null	n
C038	“name38”	null	n	C113	“name113”	null	n
C039	“name39”	null	y	C114	“name114”	null	y
C040	“name40”	null	n	C115	“name115”	null	n
C041	“name41”	null	n	C116	“name116”	null	n
C042	“name42”	null	y	C117	“name117”	null	y
C043	“name43”	null	n	C118	“name118”	null	n
C044	“name44”	null	n	C119	“name119”	null	n
C045	“name45”	null	y	C120	“name120”	null	y
C046	“name46”	null	n	C121	“name121”	null	n
C047	“name47”	null	n	C122	“name122”	null	n
C048	“name48”	null	y	C123	“name123”	null	y
C049	“name49”	null	n	C124	“name124”	null	n
C050	“name50”	null	n	C125	“name125”	null	n
C051	“name51”	null	y	C126	“name126”	null	y
C052	“name52”	null	n	C127	“name127”	null	n
C053	“name53”	null	n	C128	“name128”	null	n
C054	“name54”	null	y	C129	“name129”	null	y
C055	“name55”	null	n	C130	“name130”	null	n
C056	“name56”	null	n	C131	“name131”	null	n
C057	“name57”	null	y	C132	“name132”	null	y
C058	“name58”	null	n	C133	“name133”	null	n
C059	“name59”	null	n	C134	“name134”	null	n
C060	“name60”	null	y	C135	“name135”	null	y
C061	“name61”	null	n	C136	“name136”	null	n
C062	“name62”	null	n	C137	“name137”	null	n
C063	“name63”	null	y	C138	“name138”	null	y
C064	“name64”	null	n	C139	“name139”	null	n
C065	“name65”	null	n	C140	“name140”	null	n
C066	“name66”	null	y	C141	“name141”	null	y
C067	“name67”	null	n	C142	“name142”	null	n
C068	“name68”	null	n	C143	“name143”	null	n
C069	“name69”	null	y	C144	“name144”	null	y
C070	“name70”	null	n	C145	“name145”	null	n
C071	“name71”	null	n	C146	“name146”	null	n
C072	“name72”	null	y	C147	“name147”	null	y
C073	“name73”	null	n	C148	“name148”	null	n
C074	“name74”	null	n	C149	“name149”	null	n
C075	“name75”	null	y	C150	“name150”	null	y

Table F.4: U150C Location Table

ID	Address	x-coordinate	y-coordinate
L001	“depot”	0	0
L002	“addr1”	1.842273266	-0.710622429
L003	“addr2”	-1.219031388	-1.364926748
L004	“addr3”	-0.364378217	-0.948889261
L005	“addr4”	0.478238575	-1.239980474
L006	“addr5”	1.143153292	-1.581089811
L007	“addr6”	-0.51966714	-0.925793821
L008	“addr7”	-0.948449157	1.737916556
L009	“addr8”	1.607275624	1.944511416
L010	“addr9”	0.247565581	-0.44175238
L011	“addr10”	-1.59608252	-0.201077757
L012	“addr11”	-0.690204576	-1.025318231
L013	“addr12”	-0.853132174	0.351321797
L014	“addr13”	-0.684286299	1.108648991
L015	“addr14”	1.100332761	1.158585899
L016	“addr15”	-0.745890483	-1.988051673
L017	“addr16”	-1.327653411	1.728861367
L018	“addr17”	-0.662992499	-0.318234124
L019	“addr18”	-0.810705818	1.372554559
L020	“addr19”	-1.365960732	0.411726876
L021	“addr20”	1.198024375	1.085214604
L022	“addr21”	1.706147731	0.363867293
L023	“addr22”	0.521263533	-0.02584799
L024	“addr23”	0.294410478	1.031570483
L025	“addr24”	-0.266557423	-0.91171009
L026	“addr25”	0.62367282	0.173668417
L027	“addr26”	0.290459262	1.983401483
L028	“addr27”	-1.490408237	-1.164935952
L029	“addr28”	1.238300673	-0.047457724
L030	“addr29”	-0.78840633	0.229387463
L031	“addr30”	-1.758138866	-1.425855857
L032	“addr31”	0.853157558	-0.455644898
L033	“addr32”	-0.383307606	1.077892894
L034	“addr33”	1.234741675	-0.85934361
L035	“addr34”	0.112833955	1.001166657
L036	“addr35”	-0.006423507	-0.849649474
L037	“addr36”	-1.608726797	-0.293528428
L038	“addr37”	-0.6815457	1.140909744
L039	“addr38”	-1.550004817	1.552963768
L040	“addr39”	0.806551083	-1.703795769
L041	“addr40”	-1.302015261	-1.137194185
L042	“addr41”	1.553223794	1.475877549
L043	“addr42”	0.941952241	0.295760749
L044	“addr43”	1.194631348	-1.450279246
L045	“addr44”	1.241960438	-0.156917435
L046	“addr45”	-0.623066377	0.563806052
L047	“addr46”	-0.591466669	1.374420564
L048	“addr47”	-1.995269623	0.580300207
L049	“addr48”	-0.144761455	1.519107163
L050	“addr49”	0.625307598	-0.556967479
L051	“addr50”	1.587137635	-0.364139245
L052	“addr51”	-1.56583616	1.256048857
L053	“addr52”	1.296905861	-0.277415519
L054	“addr53”	0.641197563	-1.977461376
L055	“addr54”	0.85683002	-0.659394902
L056	“addr55”	-1.6841589	-0.323319081
L057	“addr56”	-1.063701227	0.975347036
L058	“addr57”	1.333333429	-1.348893401
L059	“addr58”	0.040804492	-0.271904982
L060	“addr59”	1.673230935	-0.842163553
L061	“addr60”	0.959885839	1.831156172
L062	“addr61”	1.075160698	-1.479295247
L063	“addr62”	-1.572158437	-0.798614303
L064	“addr63”	0.935727713	1.182439954
L065	“addr64”	-1.257543294	-1.905300314
L066	“addr65”	-0.665607652	-0.580274153
L067	“addr66”	0.153385403	-1.103147716
L068	“addr67”	-1.794956348	0.334725429
L069	“addr68”	-1.976757021	1.183412328
L070	“addr69”	0.923896622	0.351704778
L071	“addr70”	-0.919135346	-0.595469568
L072	“addr71”	0.963250559	-1.76744807
L073	“addr72”	0.527595562	-0.10225705
L074	“addr73”	1.764549757	-0.464861272
L075	“addr74”	0.323882465	1.404716782

Table F.4:U150C Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L076	“addr75”	-1.386902576	-0.526364521
L077	“addr76”	0.674843738	-1.134505684
L078	“addr77”	-0.949325306	1.3000368
L079	“addr78”	-0.450099098	1.986454316
L080	“addr79”	-0.758771472	1.611322212
L081	“addr80”	-0.98703908	1.92586054
L082	“addr81”	1.708694121	0.39682253
L083	“addr82”	-1.79395571	-0.586472673
L084	“addr83”	-1.005073419	0.175526826
L085	“addr84”	-0.979338244	1.718785681
L086	“addr85”	-1.960379235	1.233215195
L087	“addr86”	1.807646347	0.103591254
L088	“addr87”	1.463006522	-1.245537265
L089	“addr88”	-1.082443724	1.745270977
L090	“addr89”	-1.563989485	-1.445285352
L091	“addr90”	1.430167581	-1.41393778
L092	“addr91”	1.451484471	-0.847211997
L093	“addr92”	-0.877014502	-1.835746882
L094	“addr93”	0.064003704	-0.915685395
L095	“addr94”	-0.45489467	-1.795608739
L096	“addr95”	0.544647438	-0.358208042
L097	“addr96”	-0.995836464	1.591680809
L098	“addr97”	1.401079032	0.637069329
L099	“addr98”	-1.659475894	1.328348583
L100	“addr99”	-1.15785188	-1.682751465
L101	“addr100”	1.55425163	0.167412465
L102	“addr101”	-0.479854036	-1.774860906
L103	“addr102”	1.66682976	-1.158057617
L104	“addr103”	0.262750017	-0.419440146
L105	“addr104”	1.640988636	-1.844437363
L106	“addr105”	0.010598914	-0.331061087
L107	“addr106”	-1.251587636	1.672597256
L108	“addr107”	1.80534829	-1.241183514
L109	“addr108”	-1.533497068	1.343919197
L110	“addr109”	-1.573039527	-1.769984173
L111	“addr110”	-1.19821625	-1.872782123
L112	“addr111”	-1.842109145	0.879398344
L113	“addr112”	1.878931745	1.508058788
L114	“addr113”	1.155736816	-1.218719831
L115	“addr114”	-1.299165852	1.800761945
L116	“addr115”	-1.95517556	-1.385258577
L117	“addr116”	-0.823528891	1.205784496
L118	“addr117”	-1.408888784	-0.969785608
L119	“addr118”	-1.170167482	-0.177051726
L120	“addr119”	1.703691986	-0.587103713
L121	“addr120”	-1.955516871	1.167509616
L122	“addr121”	0.510100447	0.310686221
L123	“addr122”	-0.735203341	0.208724175
L124	“addr123”	1.881504302	0.829501751
L125	“addr124”	-1.733715013	1.548215874
L126	“addr125”	0.537799248	-0.54100989
L127	“addr126”	1.423856837	-1.921881711
L128	“addr127”	-0.258486847	-1.923199898
L129	“addr128”	1.848243536	1.306606624
L130	“addr129”	1.593337361	-1.710257763
L131	“addr130”	1.32723897	1.511102554
L132	“addr131”	0.071009018	0.805317827
L133	“addr132”	0.199803471	0.977083484
L134	“addr133”	-0.792825464	0.187253762
L135	“addr134”	1.607491984	-1.196910913
L136	“addr135”	-0.819230517	1.133838332
L137	“addr136”	-1.990930344	-0.899829377
L138	“addr137”	0.600358512	0.328007734
L139	“addr138”	-1.589416025	1.886687478
L140	“addr139”	-0.113217041	-0.056710667
L141	“addr140”	-1.545426514	1.577010166
L142	“addr141”	-1.583628784	1.049864885
L143	“addr142”	0.328492599	1.685345453
L144	“addr143”	0.964873312	-0.549827602
L145	“addr144”	-0.606155616	0.213555728
L146	“addr145”	-1.003813085	0.883622223
L147	“addr146”	1.503915063	0.16450087
L148	“addr147”	1.468848618	-0.924842057
L149	“addr148”	-0.830586675	1.720087824
L150	“addr149”	-1.912830557	-1.23744905

Table F.4:U150C Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L151	“addr150”	0.04772133	0.845867432
L152	“addr151”	1.302802002	0.62247146
L153	“addr152”	-0.327127054	1.431676897
L154	“addr153”	1.876431184	-0.074462827
L155	“addr154”	1.909777723	0.550480462
L156	“addr155”	0.438186368	1.78569544
L157	“addr156”	-0.692380035	1.015218075
L158	“addr157”	0.239281083	-0.970548338
L159	“addr158”	-0.656236559	-1.86387224
L160	“addr159”	0.166290457	1.995612869
L161	“addr160”	1.490902933	-1.571336263
L162	“addr161”	0.093384354	0.986372511
L163	“addr162”	-1.893393221	0.994789471
L164	“addr163”	-1.223260349	1.625716002
L165	“addr164”	1.335802818	-0.6019219
L166	“addr165”	-0.250738899	1.041287166
L167	“addr166”	0.533165808	-0.924286238
L168	“addr167”	-0.947352053	0.571007041
L169	“addr168”	-1.84623023	0.67813891
L170	“addr169”	-1.852252773	-1.394102594
L171	“addr170”	0.626777417	1.199751288
L172	“addr171”	1.973044441	-0.784286069
L173	“addr172”	-1.581658681	0.361711487
L174	“addr173”	1.875043558	0.364152697
L175	“addr174”	0.306301458	1.430574756
L176	“addr175”	-1.986799103	1.805547905
L177	“addr176”	-1.169332343	-0.495879271
L178	“addr177”	1.845977414	1.165137506
L179	“addr178”	0.950282741	0.268408797
L180	“addr179”	0.433484427	-0.759873489
L181	“addr180”	0.12746799	0.327743966
L182	“addr181”	0.54610266	0.218937009
L183	“addr182”	1.378858732	-1.568106384
L184	“addr183”	1.947360334	1.301188016
L185	“addr184”	-0.727955496	-0.727447562
L186	“addr185”	-0.276737576	-0.888893965
L187	“addr186”	-1.585965118	1.993709233
L188	“addr187”	1.159170834	1.332221648
L189	“addr188”	0.66897655	-1.841775737
L190	“addr189”	-0.580760816	1.151366716
L191	“addr190”	-0.841044595	-1.364415921
L192	“addr191”	-1.837686075	1.385138833
L193	“addr192”	0.043122996	1.723454908
L194	“addr193”	0.232118814	0.572160033
L195	“addr194”	-0.863207848	1.788313501
L196	“addr195”	-0.311552695	-1.15038948
L197	“addr196”	-0.376445228	-1.19191003
L198	“addr197”	1.699991304	-0.907324994
L199	“addr198”	1.429878252	0.422412988
L200	“addr199”	-0.801656418	0.58069571
L201	“addr200”	-1.726954344	1.443705122
L202	“addr201”	-1.303384444	0.363982309
L203	“addr202”	0.959710598	1.255031025
L204	“addr203”	-1.694980218	-0.459614689
L205	“addr204”	1.203742688	1.118258341
L206	“addr205”	-1.270329029	1.810895612
L207	“addr206”	-0.9515532	-0.861252912
L208	“addr207”	0.919767996	1.968688237
L209	“addr208”	-1.080354442	-1.519560558
L210	“addr209”	0.032222754	0.386187277
L211	“addr210”	-1.597479122	0.39787894
L212	“addr211”	0.511711211	-1.900118048
L213	“addr212”	-1.973484575	-0.250657944
L214	“addr213”	-0.661975255	1.75278175
L215	“addr214”	1.659003248	0.877925179
L216	“addr215”	1.871509555	-1.560444527
L217	“addr216”	-0.450157947	-1.904917796
L218	“addr217”	-0.768714765	-0.590446954
L219	“addr218”	1.635329948	1.473767004
L220	“addr219”	-0.230158169	-1.381932408
L221	“addr220”	1.401491323	-0.744009853
L222	“addr221”	1.961520319	-0.723649682
L223	“addr222”	0.423025968	0.902255348
L224	“addr223”	1.901287329	0.836046169
L225	“addr224”	-1.36433941	-0.342001462

Table F.4:U150C Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L226	“addr225”	1.614121127	-1.746122667
L227	“addr226”	-0.533952213	-1.515192924
L228	“addr227”	0.504233584	0.917553721
L229	“addr228”	0.134764752	-1.698146288
L230	“addr229”	-0.472742641	-1.725949714
L231	“addr230”	-0.235630404	0.70563679
L232	“addr231”	-1.952566162	-1.013456296
L233	“addr232”	1.365879166	0.683773243
L234	“addr233”	0.84574556	-1.643540965
L235	“addr234”	0.343439285	-1.688752452
L236	“addr235”	1.373840344	0.782233204
L237	“addr236”	1.84836444	-1.984823209
L238	“addr237”	-0.89749638	0.186557117
L239	“addr238”	0.130128467	1.71680701
L240	“addr239”	-0.30766212	1.629016432
L241	“addr240”	0.973297854	-1.992886469
L242	“addr241”	0.718219396	-0.268795005
L243	“addr242”	1.375800983	-0.514966942
L244	“addr243”	-1.191997258	-1.923456088
L245	“addr244”	-1.184663791	0.485012204
L246	“addr245”	-1.690143714	-1.592808136
L247	“addr246”	1.767126513	1.643214017
L248	“addr247”	0.981511808	1.2255931
L249	“addr248”	1.577635201	0.099377998
L250	“addr249”	-1.301377378	0.318361108
L251	“addr250”	-0.634737432	-1.396010969
L252	“addr251”	0.808327737	-1.244644634
L253	“addr252”	0.310451041	-0.601937613
L254	“addr253”	0.508027972	-0.923035617
L255	“addr254”	-0.48596877	-0.544487035
L256	“addr255”	0.508283793	1.66776283
L257	“addr256”	0.544048121	0.525729408
L258	“addr257”	1.242653305	-1.275131188
L259	“addr258”	1.857714113	-1.613368788
L260	“addr259”	1.016194095	0.764482344
L261	“addr260”	1.476538325	0.816145054
L262	“addr261”	-1.261742037	-1.729756988
L263	“addr262”	-1.390907209	-0.987311157
L264	“addr263”	-1.735884815	-1.127799928
L265	“addr264”	0.357277578	-1.457086891
L266	“addr265”	0.873850844	-1.64465343
L267	“addr266”	0.20835723	-0.632399733
L268	“addr267”	-0.954118338	1.831821977
L269	“addr268”	-1.293556517	0.0942841
L270	“addr269”	-1.563651891	1.571048077
L271	“addr270”	1.513508838	1.327802958
L272	“addr271”	-0.145456776	0.26721536
L273	“addr272”	0.011419382	0.837660631
L274	“addr273”	1.028444909	-1.066303568
L275	“addr274”	-1.337117548	1.233721476
L276	“addr275”	1.083326604	-0.850883333
L277	“addr276”	1.540611077	0.93942576
L278	“addr277”	0.910324252	0.039971016
L279	“addr278”	-0.249586949	-0.525634518
L280	“addr279”	1.980748008	1.169829965
L281	“addr280”	-0.084680953	-1.594282159
L282	“addr281”	-1.334170962	1.415335241
L283	“addr282”	1.454270591	-0.199311078
L284	“addr283”	-1.249498789	-1.055658186
L285	“addr284”	-0.919706938	-1.390221321
L286	“addr285”	1.583722144	0.571718407
L287	“addr286”	0.759489929	-1.866667871
L288	“addr287”	1.186745238	0.053476544
L289	“addr288”	-1.686406931	0.571497987
L290	“addr289”	-1.35534098	0.29005145
L291	“addr290”	-0.084780521	1.630375418
L292	“addr291”	0.235418663	1.944718412
L293	“addr292”	-1.083618249	1.254780614
L294	“addr293”	1.004498641	0.643280727
L295	“addr294”	-0.23278623	-0.21927357
L296	“addr295”	-0.127796785	-1.293676036
L297	“addr296”	1.223273843	0.148360668
L298	“addr297”	1.864576834	1.275486042
L299	“addr298”	1.606195945	-1.711232782
L300	“addr299”	-0.312236752	-1.269649199
L301	“addr300”	0.573867272	-0.604311477

Table F.5: U150C Request Table

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R001	C001	L002	L005	5/27/2003	7:36	D	5	1	7:20	7:50
R002	C002	L003	L006	5/27/2003	7:20	P	5	1	7:05	7:35
R003	C003	L004	L007	5/27/2003	7:53	D	5	0	7:38	8:08
R004	C004	L008	L011	5/27/2003	8:07	P	5	1	7:51	8:21
R005	C005	L009	L012	5/27/2003	8:52	D	5	1	8:36	9:06
R006	C006	L010	L013	5/27/2003	9:01	D	5	0	8:45	9:15
R007	C007	L014	L017	5/27/2003	9:51	D	5	1	9:35	10:05
R008	C008	L015	L018	5/27/2003	9:19	P	5	1	9:03	9:33
R009	C009	L016	L019	5/27/2003	9:35	P	5	0	9:19	9:49
R010	C010	L020	L023	5/27/2003	10:50	D	5	1	10:35	11:05
R011	C011	L021	L024	5/27/2003	10:34	P	5	1	10:18	10:48
R012	C012	L022	L025	5/27/2003	11:09	D	5	0	10:54	11:24
R013	C013	L026	L029	5/27/2003	11:19	P	5	1	11:03	11:33
R014	C014	L027	L030	5/27/2003	11:29	P	5	1	11:14	11:44
R015	C015	L028	L031	5/27/2003	11:45	P	5	0	11:29	11:59
R016	C016	L032	L035	5/27/2003	7:03	P	5	1	6:47	7:17
R017	C017	L033	L036	5/27/2003	7:48	D	5	1	7:33	8:03
R018	C018	L034	L037	5/27/2003	7:26	P	5	0	7:11	7:41
R019	C019	L038	L041	5/27/2003	8:45	D	5	1	8:30	9:00
R020	C020	L039	L042	5/27/2003	9:02	D	5	1	8:46	9:16
R021	C021	L040	L043	5/27/2003	8:34	P	5	0	8:18	8:48
R022	C022	L044	L047	5/27/2003	9:21	P	5	1	9:06	9:36
R023	C023	L045	L048	5/27/2003	9:30	P	5	1	9:14	9:44
R024	C024	L046	L049	5/27/2003	9:41	P	5	0	9:25	9:55
R025	C025	L050	L053	5/27/2003	10:59	D	5	1	10:44	11:14
R026	C026	L051	L054	5/27/2003	10:29	P	5	1	10:14	10:44
R027	C027	L052	L055	5/27/2003	10:45	P	5	0	10:29	10:59
R028	C028	L056	L059	5/27/2003	11:32	P	5	1	11:16	11:46
R029	C029	L057	L060	5/27/2003	12:16	D	5	1	12:01	12:31
R030	C030	L058	L061	5/27/2003	12:29	D	5	0	12:14	12:44
R031	C031	L062	L065	5/27/2003	7:05	P	5	1	6:50	7:20
R032	C032	L063	L066	5/27/2003	7:19	P	5	1	7:03	7:33
R033	C033	L064	L067	5/27/2003	7:33	P	5	0	7:18	7:48
R034	C034	L068	L071	5/27/2003	8:19	P	5	1	8:04	8:34
R035	C035	L069	L072	5/27/2003	8:27	P	5	1	8:11	8:41
R036	C036	L070	L073	5/27/2003	8:41	P	5	0	8:25	8:55
R037	C037	L074	L077	5/27/2003	9:23	P	5	1	9:07	9:37
R038	C038	L075	L078	5/27/2003	10:13	D	5	1	9:57	10:27
R039	C039	L076	L079	5/27/2003	10:20	D	5	0	10:05	10:35
R040	C040	L080	L083	5/27/2003	11:03	D	5	1	10:48	11:18
R041	C041	L081	L084	5/27/2003	10:33	P	5	1	10:17	10:47
R042	C042	L082	L085	5/27/2003	11:21	D	5	0	11:06	11:36
R043	C043	L086	L089	5/27/2003	11:29	P	5	1	11:14	11:44
R044	C044	L087	L090	5/27/2003	12:27	D	5	1	12:11	12:41
R045	C045	L088	L091	5/27/2003	12:41	D	5	0	12:25	12:55
R046	C046	L092	L095	5/27/2003	7:35	D	5	1	7:19	7:49
R047	C047	L093	L096	5/27/2003	7:45	D	5	1	7:29	7:59
R048	C048	L094	L097	5/27/2003	7:57	D	5	0	7:42	8:12
R049	C049	L098	L101	5/27/2003	8:10	P	5	1	7:55	8:25
R050	C050	L099	L102	5/27/2003	8:25	P	5	1	8:09	8:39
R051	C051	L100	L103	5/27/2003	9:19	D	5	0	9:03	9:33
R052	C052	L104	L107	5/27/2003	10:03	D	5	1	9:48	10:18
R053	C053	L105	L108	5/27/2003	9:39	P	5	1	9:24	9:54
R054	C054	L106	L109	5/27/2003	10:38	D	5	0	10:23	10:53
R055	C055	L110	L113	5/27/2003	11:29	D	5	1	11:13	11:43
R056	C056	L111	L114	5/27/2003	11:42	D	5	1	11:27	11:57
R057	C057	L112	L115	5/27/2003	11:59	D	5	0	11:43	12:13
R058	C058	L116	L119	5/27/2003	12:14	P	5	1	11:58	12:28
R059	C059	L117	L120	5/27/2003	13:00	D	5	1	12:45	13:15
R060	C060	L118	L121	5/27/2003	12:39	P	5	0	12:23	12:53
R061	C061	L122	L125	5/27/2003	7:02	P	5	1	6:46	7:16
R062	C062	L123	L126	5/27/2003	7:54	D	5	1	7:38	8:08
R063	C063	L124	L127	5/27/2003	8:04	D	5	0	7:48	8:18
R064	C064	L128	L131	5/27/2003	8:14	P	5	1	7:58	8:28
R065	C065	L129	L132	5/27/2003	8:30	P	5	1	8:15	8:45
R066	C066	L130	L133	5/27/2003	9:14	D	5	0	8:59	9:29
R067	C067	L134	L137	5/27/2003	9:23	P	5	1	9:08	9:38
R068	C068	L135	L138	5/27/2003	9:36	P	5	1	9:21	9:51
R069	C069	L136	L139	5/27/2003	10:30	D	5	0	10:15	10:45
R070	C070	L140	L143	5/27/2003	11:12	D	5	1	10:56	11:26
R071	C071	L141	L144	5/27/2003	10:54	P	5	1	10:38	11:08
R072	C072	L142	L145	5/27/2003	11:01	P	5	0	10:45	11:15
R073	C073	L146	L149	5/27/2003	11:41	P	5	1	11:26	11:56
R074	C074	L147	L150	5/27/2003	12:32	D	5	1	12:17	12:47
R075	C075	L148	L151	5/27/2003	12:46	D	5	0	12:30	13:00

Table F.5: U150C Request Table *(continued)*

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R076	C076	L152	L155	5/27/2003	7:04	P	5	1	6:49	7:19
R077	C077	L153	L156	5/27/2003	7:45	D	5	1	7:30	8:00
R078	C078	L154	L157	5/27/2003	7:28	P	5	0	7:12	7:42
R079	C079	L158	L161	5/27/2003	8:48	D	5	1	8:33	9:03
R080	C080	L159	L162	5/27/2003	8:15	P	5	1	7:59	8:29
R081	C081	L160	L163	5/27/2003	9:13	D	5	0	8:57	9:27
R082	C082	L164	L167	5/27/2003	9:21	P	5	1	9:05	9:35
R083	C083	L165	L168	5/27/2003	10:10	D	5	1	9:55	10:25
R084	C084	L166	L169	5/27/2003	10:18	D	5	0	10:03	10:33
R085	C085	L170	L173	5/27/2003	10:29	P	5	1	10:14	10:44
R086	C086	L171	L174	5/27/2003	11:29	D	5	1	11:13	11:43
R087	C087	L172	L175	5/27/2003	11:39	D	5	0	11:24	11:54
R088	C088	L176	L179	5/27/2003	11:51	P	5	1	11:36	12:06
R089	C089	L177	L180	5/27/2003	12:36	D	5	1	12:21	12:51
R090	C090	L178	L181	5/27/2003	12:45	D	5	0	12:29	12:59
R091	C091	L182	L185	5/27/2003	7:42	D	5	1	7:26	7:56
R092	C092	L183	L186	5/27/2003	7:13	P	5	1	6:57	7:27
R093	C093	L184	L187	5/27/2003	7:26	P	5	0	7:11	7:41
R094	C094	L188	L191	5/27/2003	8:58	D	5	1	8:42	9:12
R095	C095	L189	L192	5/27/2003	8:31	P	5	1	8:15	8:45
R096	C096	L190	L193	5/27/2003	8:45	P	5	0	8:30	9:00
R097	C097	L194	L197	5/27/2003	10:00	D	5	1	9:45	10:15
R098	C098	L195	L198	5/27/2003	10:11	D	5	1	9:56	10:26
R099	C099	L196	L199	5/27/2003	10:20	D	5	0	10:05	10:35
R100	C100	L200	L203	5/27/2003	10:32	P	5	1	10:17	10:47
R101	C101	L201	L204	5/27/2003	10:41	P	5	1	10:25	10:55
R102	C102	L202	L205	5/27/2003	10:49	P	5	0	10:34	11:04
R103	C103	L206	L209	5/27/2003	11:44	P	5	1	11:28	11:58
R104	C104	L207	L210	5/27/2003	12:41	D	5	1	12:25	12:55
R105	C105	L208	L211	5/27/2003	12:51	D	5	0	12:35	13:05
R106	C106	L212	L215	5/27/2003	7:44	D	5	1	7:29	7:59
R107	C107	L213	L216	5/27/2003	7:57	D	5	1	7:41	8:11
R108	C108	L214	L217	5/27/2003	8:09	D	5	0	7:53	8:23
R109	C109	L218	L221	5/27/2003	8:58	D	5	1	8:42	9:12
R110	C110	L219	L222	5/27/2003	9:05	D	5	1	8:49	9:19
R111	C111	L220	L223	5/27/2003	9:16	D	5	0	9:01	9:31
R112	C112	L224	L227	5/27/2003	9:26	P	5	1	9:10	9:40
R113	C113	L225	L228	5/27/2003	10:20	D	5	1	10:05	10:35
R114	C114	L226	L229	5/27/2003	9:56	P	5	0	9:41	10:11
R115	C115	L230	L233	5/27/2003	11:21	D	5	1	11:06	11:36
R116	C116	L231	L234	5/27/2003	10:53	P	5	1	10:37	11:07
R117	C117	L232	L235	5/27/2003	11:05	P	5	0	10:49	11:19
R118	C118	L236	L239	5/27/2003	11:53	P	5	1	11:37	12:07
R119	C119	L237	L240	5/27/2003	12:06	P	5	1	11:51	12:21
R120	C120	L238	L241	5/27/2003	12:55	D	5	0	12:40	13:10
R121	C121	L242	L245	5/27/2003	7:35	D	5	1	7:20	7:50
R122	C122	L243	L246	5/27/2003	7:09	P	5	1	6:54	7:24
R123	C123	L244	L247	5/27/2003	7:23	P	5	0	7:08	7:38
R124	C124	L248	L251	5/27/2003	8:47	D	5	1	8:31	9:01
R125	C125	L249	L252	5/27/2003	8:56	D	5	1	8:41	9:11
R126	C126	L250	L253	5/27/2003	8:36	P	5	0	8:21	8:51
R127	C127	L254	L257	5/27/2003	9:10	P	5	1	8:54	9:24
R128	C128	L255	L258	5/27/2003	9:18	P	5	1	9:02	9:32
R129	C129	L256	L259	5/27/2003	9:30	P	5	0	9:15	9:45
R130	C130	L260	L263	5/27/2003	10:39	D	5	1	10:23	10:53
R131	C131	L261	L264	5/27/2003	10:45	D	5	1	10:29	10:59
R132	C132	L262	L265	5/27/2003	10:56	D	5	0	10:41	11:11
R133	C133	L266	L269	5/27/2003	11:35	D	5	1	11:20	11:50
R134	C134	L267	L270	5/27/2003	11:12	P	5	1	10:56	11:26
R135	C135	L268	L271	5/27/2003	11:59	D	5	0	11:43	12:13
R136	C136	L272	L275	5/27/2003	7:34	D	5	1	7:19	7:49
R137	C137	L273	L276	5/27/2003	7:49	D	5	1	7:33	8:03
R138	C138	L274	L277	5/27/2003	7:59	D	5	0	7:44	8:14
R139	C139	L278	L281	5/27/2003	8:07	P	5	1	7:52	8:22
R140	C140	L279	L282	5/27/2003	8:16	P	5	1	8:01	8:31
R141	C141	L280	L283	5/27/2003	8:30	P	5	0	8:14	8:44
R142	C142	L284	L287	5/27/2003	9:28	P	5	1	9:13	9:43
R143	C143	L285	L288	5/27/2003	9:34	P	5	1	9:19	9:49
R144	C144	L286	L289	5/27/2003	9:49	P	5	0	9:34	10:04
R145	C145	L290	L293	5/27/2003	10:33	P	5	1	10:17	10:47
R146	C146	L291	L294	5/27/2003	10:43	P	5	1	10:28	10:58
R147	C147	L292	L295	5/27/2003	11:20	D	5	0	11:05	11:35
R148	C148	L296	L299	5/27/2003	11:28	P	5	1	11:13	11:43
R149	C149	L297	L300	5/27/2003	11:39	P	5	1	11:24	11:54
R150	C150	L298	L301	5/27/2003	12:21	D	5	0	12:06	12:36

Appendix G

U250 Data Set

Table G.1: U250 Model Table

MRT	FIXED	60	TW	VARY	WAIT	30	SPEED	20	ROUTE	480

Table G.2: U250 Vehicle Table

ID	Capacity	Wheelchairs
V01	4	2
V02	4	2
V03	4	2
V04	4	2
V05	4	2
V06	4	2
V07	4	2
V08	4	2
V09	4	2
V10	4	2
V11	4	2
V12	4	2
V13	4	2

Table G.3: U250 Client Table

ID	Name	Address	Wheelchair	ID	Name	Address	Wheelchair
C001	“name1”	null	n	C066	“name66”	null	y
C002	“name2”	null	n	C067	“name67”	null	n
C003	“name3”	null	y	C068	“name68”	null	n
C004	“name4”	null	n	C069	“name69”	null	y
C005	“name5”	null	n	C070	“name70”	null	n
C006	“name6”	null	y	C071	“name71”	null	n
C007	“name7”	null	n	C072	“name72”	null	y
C008	“name8”	null	n	C073	“name73”	null	n
C009	“name9”	null	y	C074	“name74”	null	n
C010	“name10”	null	n	C075	“name75”	null	y
C011	“name11”	null	n	C076	“name76”	null	n
C012	“name12”	null	y	C077	“name77”	null	n
C013	“name13”	null	n	C078	“name78”	null	y
C014	“name14”	null	n	C079	“name79”	null	n
C015	“name15”	null	y	C080	“name80”	null	n
C016	“name16”	null	n	C081	“name81”	null	y
C017	“name17”	null	n	C082	“name82”	null	n
C018	“name18”	null	y	C083	“name83”	null	n
C019	“name19”	null	n	C084	“name84”	null	y
C020	“name20”	null	n	C085	“name85”	null	n
C021	“name21”	null	y	C086	“name86”	null	n
C022	“name22”	null	n	C087	“name87”	null	y
C023	“name23”	null	n	C088	“name88”	null	n
C024	“name24”	null	y	C089	“name89”	null	n
C025	“name25”	null	n	C090	“name90”	null	y
C026	“name26”	null	n	C091	“name91”	null	n
C027	“name27”	null	y	C092	“name92”	null	n
C028	“name28”	null	n	C093	“name93”	null	y
C029	“name29”	null	n	C094	“name94”	null	n
C030	“name30”	null	y	C095	“name95”	null	n
C031	“name31”	null	n	C096	“name96”	null	n
C032	“name32”	null	n	C097	“name97”	null	y
C033	“name33”	null	y	C098	“name98”	null	n
C034	“name34”	null	n	C099	“name99”	null	n
C035	“name35”	null	n	C100	“name100”	null	y
C036	“name36”	null	y	C101	“name101”	null	n
C037	“name37”	null	n	C102	“name102”	null	n
C038	“name38”	null	n	C103	“name103”	null	y
C039	“name39”	null	n	C104	“name104”	null	n
C040	“name40”	null	y	C105	“name105”	null	n
C041	“name41”	null	n	C106	“name106”	null	y
C042	“name42”	null	n	C107	“name107”	null	n
C043	“name43”	null	y	C108	“name108”	null	n
C044	“name44”	null	n	C109	“name109”	null	y
C045	“name45”	null	n	C110	“name110”	null	n
C046	“name46”	null	y	C111	“name111”	null	n
C047	“name47”	null	n	C112	“name112”	null	y
C048	“name48”	null	n	C113	“name113”	null	n
C049	“name49”	null	y	C114	“name114”	null	n
C050	“name50”	null	n	C115	“name115”	null	n
C051	“name51”	null	n	C116	“name116”	null	n
C052	“name52”	null	y	C117	“name117”	null	y
C053	“name53”	null	n	C118	“name118”	null	n
C054	“name54”	null	n	C119	“name119”	null	n
C055	“name55”	null	y	C120	“name120”	null	y
C056	“name56”	null	n	C121	“name121”	null	n
C057	“name57”	null	n	C122	“name122”	null	n
C058	“name58”	null	n	C123	“name123”	null	y
C059	“name59”	null	n	C124	“name124”	null	n
C060	“name60”	null	y	C125	“name125”	null	n
C061	“name61”	null	n	C126	“name126”	null	y
C062	“name62”	null	n	C127	“name127”	null	n
C063	“name63”	null	y	C128	“name128”	null	n
C064	“name64”	null	n	C129	“name129”	null	y
C065	“name65”	null	n	C130	“name130”	null	n

Table G.3:U250 Client Table *(continued)*

ID	Name	Address	Wheelchair	ID	Name	Address	Wheelchair
C131	“name131”	null	n	C191	“name191”	null	y
C132	“name132”	null	y	C192	“name192”	null	n
C133	“name133”	null	n	C193	“name193”	null	n
C134	“name134”	null	n	C194	“name194”	null	y
C135	“name135”	null	y	C195	“name195”	null	n
C136	“name136”	null	n	C196	“name196”	null	n
C137	“name137”	null	n	C197	“name197”	null	y
C138	“name138”	null	y	C198	“name198”	null	n
C139	“name139”	null	n	C199	“name199”	null	n
C140	“name140”	null	n	C200	“name200”	null	y
C141	“name141”	null	y	C201	“name201”	null	n
C142	“name142”	null	n	C202	“name202”	null	n
C143	“name143”	null	n	C203	“name203”	null	y
C144	“name144”	null	y	C204	“name204”	null	n
C145	“name145”	null	n	C205	“name205”	null	n
C146	“name146”	null	n	C206	“name206”	null	y
C147	“name147”	null	y	C207	“name207”	null	n
C148	“name148”	null	n	C208	“name208”	null	n
C149	“name149”	null	n	C209	“name209”	null	n
C150	“name150”	null	y	C210	“name210”	null	y
C151	“name151”	null	n	C211	“name211”	null	n
C152	“name152”	null	n	C212	“name212”	null	n
C153	“name153”	null	n	C213	“name213”	null	y
C154	“name154”	null	y	C214	“name214”	null	n
C155	“name155”	null	n	C215	“name215”	null	n
C156	“name156”	null	n	C216	“name216”	null	y
C157	“name157”	null	y	C217	“name217”	null	n
C158	“name158”	null	n	C218	“name218”	null	n
C159	“name159”	null	n	C219	“name219”	null	y
C160	“name160”	null	y	C220	“name220”	null	n
C161	“name161”	null	n	C221	“name221”	null	n
C162	“name162”	null	n	C222	“name222”	null	y
C163	“name163”	null	y	C223	“name223”	null	n
C164	“name164”	null	n	C224	“name224”	null	n
C165	“name165”	null	n	C225	“name225”	null	y
C166	“name166”	null	y	C226	“name226”	null	n
C167	“name167”	null	n	C227	“name227”	null	n
C168	“name168”	null	n	C228	“name228”	null	n
C169	“name169”	null	y	C229	“name229”	null	n
C170	“name170”	null	n	C230	“name230”	null	y
C171	“name171”	null	n	C231	“name231”	null	n
C172	“name172”	null	n	C232	“name232”	null	n
C173	“name173”	null	n	C233	“name233”	null	y
C174	“name174”	null	y	C234	“name234”	null	n
C175	“name175”	null	n	C235	“name235”	null	n
C176	“name176”	null	n	C236	“name236”	null	y
C177	“name177”	null	y	C237	“name237”	null	n
C178	“name178”	null	n	C238	“name238”	null	n
C179	“name179”	null	n	C239	“name239”	null	y
C180	“name180”	null	y	C240	“name240”	null	n
C181	“name181”	null	n	C241	“name241”	null	n
C182	“name182”	null	n	C242	“name242”	null	y
C183	“name183”	null	y	C243	“name243”	null	n
C184	“name184”	null	n	C244	“name244”	null	n
C185	“name185”	null	n	C245	“name245”	null	y
C186	“name186”	null	y	C246	“name246”	null	n
C187	“name187”	null	n	C247	“name247”	null	n
C188	“name188”	null	n	C248	“name248”	null	y
C189	“name189”	null	y	C249	“name249”	null	n
C190	“name190”	null	n	C250	“name250”	null	n

Table G.4: U250 Location Table

ID	Address	x-coordinate	y-coordinate
L001	“depot”	0	0
L002	“addr1”	-1.495914137	0.198995339
L003	“addr2”	0.50315484	0.781868548
L004	“addr3”	1.874786555	1.398289229
L005	“addr4”	-1.474858978	-0.681457236
L006	“addr5”	1.778064222	-0.72846352
L007	“addr6”	-0.867402745	-1.956911835
L008	“addr7”	1.47346635	1.642453541
L009	“addr8”	-0.983226564	-0.846906925
L010	“addr9”	1.126271065	-0.932471505
L011	“addr10”	-1.395749025	1.840237717
L012	“addr11”	-0.569649378	-1.481715028
L013	“addr12”	0.039152951	1.63068653
L014	“addr13”	-1.496529913	-1.899219548
L015	“addr14”	1.397322233	-0.079996545
L016	“addr15”	1.186000748	0.87789248
L017	“addr16”	-0.172097634	1.035719299
L018	“addr17”	1.534187267	0.176349036
L019	“addr18”	1.540411399	0.539588885
L020	“addr19”	1.08426268	0.201868917
L021	“addr20”	-0.096943513	1.985478928
L022	“addr21”	-0.626140296	0.352181444
L023	“addr22”	0.390691726	-0.022264538
L024	“addr23”	-0.757672806	1.309103081
L025	“addr24”	-1.281520166	-0.871212832
L026	“addr25”	-0.408287342	1.807253062
L027	“addr26”	0.219972554	1.393643343
L028	“addr27”	0.575330473	-1.08160148
L029	“addr28”	0.3759822	0.262855943
L030	“addr29”	-1.409009202	1.756823525
L031	“addr30”	0.373272831	1.424418124
L032	“addr31”	0.991121136	-1.297475892
L033	“addr32”	-0.33024452	1.75898016
L034	“addr33”	-0.917832681	1.555437787
L035	“addr34”	1.765528359	1.911667542
L036	“addr35”	-1.229181406	-1.524679036
L037	“addr36”	1.726758733	-1.812305148
L038	“addr37”	1.124757028	1.128295706
L039	“addr38”	-1.517939758	1.147873745
L040	“addr39”	-1.849213222	-1.847508575
L041	“addr40”	-0.491324915	1.034019058
L042	“addr41”	-1.746642455	-1.128384007
L043	“addr42”	0.248426753	0.033165857
L044	“addr43”	1.121649233	0.126694073
L045	“addr44”	1.441065808	-1.606587111
L046	“addr45”	-0.796455413	-0.837380822
L047	“addr46”	-1.464714001	-0.466881998
L048	“addr47”	0.45997025	-1.302428977
L049	“addr48”	-1.564653117	0.539073138
L050	“addr49”	0.753244857	-1.380244901
L051	“addr50”	-1.936107675	-1.171802232
L052	“addr51”	0.008226859	-1.802668587
L053	“addr52”	0.569212625	1.251190511
L054	“addr53”	-1.804786462	-0.746201091
L055	“addr54”	0.477853708	0.545489771
L056	“addr55”	1.218696922	0.452516827
L057	“addr56”	-1.308867977	1.798796198
L058	“addr57”	0.956239478	1.650441719
L059	“addr58”	-0.053227171	-1.536679765
L060	“addr59”	-0.02311157	-0.777426958
L061	“addr60”	-0.785458444	1.46482017
L062	“addr61”	1.878704365	0.325000252
L063	“addr62”	-0.532601419	0.079406135
L064	“addr63”	-1.279865514	0.255853999
L065	“addr64”	1.367360944	-0.495012003
L066	“addr65”	1.680119926	0.389870486
L067	“addr66”	1.627911831	1.433155268
L068	“addr67”	-0.133020839	-1.396318673
L069	“addr68”	-0.83754419	-1.568819904
L070	“addr69”	-0.893730727	-0.324649206
L071	“addr70”	0.6009716	0.572741964
L072	“addr71”	-0.427019754	1.833086844
L073	“addr72”	-0.645326745	-0.490120242
L074	“addr73”	-1.706423758	1.444458723
L075	“addr74”	-0.868854747	1.057582147

Table G.4:U250 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L076	“addr75”	1.828551894	0.778860684
L077	“addr76”	-0.349926591	-1.633667185
L078	“addr77”	1.856328752	-0.157632713
L079	“addr78”	-0.717650689	1.886835109
L080	“addr79”	1.914474228	-1.524468395
L081	“addr80”	0.432984534	1.585046102
L082	“addr81”	-0.785620191	1.221085315
L083	“addr82”	0.710923922	-0.755445271
L084	“addr83”	1.230693047	0.909903647
L085	“addr84”	0.276504946	-1.802751379
L086	“addr85”	0.676953134	1.48000704
L087	“addr86”	1.229968274	-1.547883727
L088	“addr87”	0.514036563	-1.76522523
L089	“addr88”	-1.497923306	0.945940445
L090	“addr89”	0.798176384	0.562955431
L091	“addr90”	-0.569439917	-0.481456013
L092	“addr91”	-0.539527773	-1.915538498
L093	“addr92”	1.053757488	1.219154609
L094	“addr93”	-0.480031421	0.252345108
L095	“addr94”	0.125484861	0.679363666
L096	“addr95”	-0.232469666	1.632755957
L097	“addr96”	-1.047186276	-1.457943181
L098	“addr97”	-1.382094388	0.401089329
L099	“addr98”	0.381100092	1.314605466
L100	“addr99”	1.612517883	-1.966060829
L101	“addr100”	1.598262189	-1.481491928
L102	“addr101”	-0.034310872	1.742875768
L103	“addr102”	-1.097040603	-1.36341712
L104	“addr103”	0.582149163	1.887475841
L105	“addr104”	0.178982598	-1.887284707
L106	“addr105”	1.133844758	0.421436948
L107	“addr106”	0.540821007	1.538071323
L108	“addr107”	1.069217041	-0.632983785
L109	“addr108”	1.463028355	-1.838405191
L110	“addr109”	0.838272759	0.379439252
L111	“addr110”	0.678666192	1.991512251
L112	“addr111”	-0.848069904	1.654329256
L113	“addr112”	1.229759203	0.008473683
L114	“addr113”	-0.245722092	-1.252383116
L115	“addr114”	-0.724647283	1.389868097
L116	“addr115”	0.971541078	1.31879243
L117	“addr116”	-0.389578844	0.14875363
L118	“addr117”	-0.788297133	0.374773715
L119	“addr118”	-0.6784722	-1.780247915
L120	“addr119”	-0.376108895	0.283063334
L121	“addr120”	1.602799019	1.778163402
L122	“addr121”	1.286789576	-1.360719146
L123	“addr122”	1.478425567	0.457458903
L124	“addr123”	-1.419504779	-0.984442151
L125	“addr124”	-0.97443457	-0.865776126
L126	“addr125”	-1.193605244	0.682691758
L127	“addr126”	-0.335825197	-0.921479927
L128	“addr127”	-1.34520636	1.590360049
L129	“addr128”	-0.822046877	-0.478449409
L130	“addr129”	0.780183707	0.206102961
L131	“addr130”	-0.646758501	0.071410753
L132	“addr131”	-1.47080568	0.255188645
L133	“addr132”	-1.529626381	0.911619902
L134	“addr133”	-0.223572693	1.969882472
L135	“addr134”	1.354630244	1.346030812
L136	“addr135”	-0.141469	0.148044926
L137	“addr136”	0.291608694	0.007622456
L138	“addr137”	0.757027869	-0.862606209
L139	“addr138”	-1.165528235	1.927587134
L140	“addr139”	0.965562893	1.803167665
L141	“addr140”	1.378199187	-1.334970449
L142	“addr141”	-0.11260116	1.603337861
L143	“addr142”	0.348619111	-0.173499135
L144	“addr143”	0.14414015	0.086919381
L145	“addr144”	-0.048344738	1.558498119
L146	“addr145”	-1.163177931	-1.55266021
L147	“addr146”	0.650879038	-0.025034815
L148	“addr147”	-1.414339819	-0.795770426
L149	“addr148”	-0.110884859	-1.014262525
L150	“addr149”	1.084909391	-1.346595265

Table G.4:U250 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L151	“addr150”	-1.957496878	-1.379221423
L152	“addr151”	1.944440052	0.416219056
L153	“addr152”	-1.806924887	1.698964964
L154	“addr153”	1.749792569	-0.764132469
L155	“addr154”	-1.971435976	-0.430921706
L156	“addr155”	0.557630313	0.10963175
L157	“addr156”	0.025817125	-0.345426055
L158	“addr157”	-0.546333647	-1.087538254
L159	“addr158”	0.046840319	-1.337400863
L160	“addr159”	1.335294061	0.858696964
L161	“addr160”	-1.425840756	1.571639743
L162	“addr161”	-0.258689189	0.465132899
L163	“addr162”	1.034837898	1.89108313
L164	“addr163”	1.336277726	-0.879950448
L165	“addr164”	-1.989848019	1.189888646
L166	“addr165”	0.567216355	1.929220644
L167	“addr166”	-1.399430745	-0.479535914
L168	“addr167”	-1.954502314	-1.798668979
L169	“addr168”	-1.186651627	1.117754577
L170	“addr169”	-0.633062242	1.751635354
L171	“addr170”	1.562719228	-0.367547433
L172	“addr171”	0.368267388	0.1440971
L173	“addr172”	1.720270529	-1.838851015
L174	“addr173”	-1.771698248	0.058653457
L175	“addr174”	0.625819121	1.465146625
L176	“addr175”	1.357291377	-0.838342404
L177	“addr176”	-0.433301392	-0.976431809
L178	“addr177”	0.678092446	-0.826042715
L179	“addr178”	0.956056327	1.945292562
L180	“addr179”	-1.312050432	0.272472373
L181	“addr180”	-0.711593242	0.268163996
L182	“addr181”	-1.400158311	-0.472456116
L183	“addr182”	-0.576656256	1.802821176
L184	“addr183”	1.897498024	-0.230270811
L185	“addr184”	1.34597211	-0.567597944
L186	“addr185”	0.798346018	0.096128084
L187	“addr186”	-1.724726339	-1.341715304
L188	“addr187”	-1.596189524	1.16772806
L189	“addr188”	0.839280232	1.185932922
L190	“addr189”	-0.273461036	1.51566444
L191	“addr190”	0.008581594	-0.526181489
L192	“addr191”	-0.680714113	1.265697422
L193	“addr192”	-0.422646754	-0.577698549
L194	“addr193”	1.890340737	-0.99338627
L195	“addr194”	1.203744703	1.758253731
L196	“addr195”	-0.924452728	1.969666889
L197	“addr196”	-0.630874858	1.175118631
L198	“addr197”	0.447111469	1.252577987
L199	“addr198”	0.088340194	1.917005255
L200	“addr199”	-0.125443921	0.513021355
L201	“addr200”	-0.199210797	-0.75067826
L202	“addr201”	1.51333383	0.372930557
L203	“addr202”	1.594788484	-1.86264024
L204	“addr203”	0.981768295	-0.283637745
L205	“addr204”	0.765343109	1.45814672
L206	“addr205”	0.697148115	-0.869935579
L207	“addr206”	0.678105703	0.199789366
L208	“addr207”	1.876955458	0.927300087
L209	“addr208”	-0.990044207	1.35643933
L210	“addr209”	-0.62863345	-1.488114934
L211	“addr210”	1.659291756	-0.39408426
L212	“addr211”	1.278318194	-0.719196322
L213	“addr212”	0.966781855	0.360928492
L214	“addr213”	-0.797409232	1.830172647
L215	“addr214”	0.35997738	-1.586250539
L216	“addr215”	-0.944485855	1.920731054
L217	“addr216”	-0.867151624	0.1480879
L218	“addr217”	-1.547941878	0.475437805
L219	“addr218”	0.793956346	0.170829788
L220	“addr219”	1.047228274	-1.176832812
L221	“addr220”	0.264638347	1.220210168
L222	“addr221”	-1.436159214	-0.531102778
L223	“addr222”	-0.818772599	-0.664110063
L224	“addr223”	-0.201091724	0.862192904
L225	“addr224”	-1.347002563	0.530319495

Table G.4:U250 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L226	“addr225”	-1.443975503	1.952192637
L227	“addr226”	1.050170865	-1.420327323
L228	“addr227”	1.236291573	0.758711092
L229	“addr228”	0.743987031	-1.943453737
L230	“addr229”	-0.269510101	0.755479933
L231	“addr230”	-1.806514861	0.355521011
L232	“addr231”	0.29034778	1.552953646
L233	“addr232”	0.371309992	-0.783409416
L234	“addr233”	0.216922587	-1.81548874
L235	“addr234”	1.243507906	0.130369927
L236	“addr235”	1.489262971	-1.75870387
L237	“addr236”	0.865933489	1.012898184
L238	“addr237”	-0.133322816	0.167337064
L239	“addr238”	0.632546917	1.666763061
L240	“addr239”	1.533088993	-0.781996609
L241	“addr240”	1.308534642	-1.200826294
L242	“addr241”	1.012119	-0.098747883
L243	“addr242”	1.703846134	-0.253480433
L244	“addr243”	-0.133315313	-0.837468458
L245	“addr244”	1.190392479	-0.441142051
L246	“addr245”	-0.392929685	-0.859389427
L247	“addr246”	0.738810642	0.726677147
L248	“addr247”	0.266572404	0.013012593
L249	“addr248”	-1.251647072	0.586192534
L250	“addr249”	-1.736016034	0.266135023
L251	“addr250”	1.666117398	0.318334432
L252	“addr251”	-0.542663906	1.666544813
L253	“addr252”	-0.760508908	1.460496949
L254	“addr253”	1.703471904	-0.723654863
L255	“addr254”	-1.781664928	-1.020980354
L256	“addr255”	-0.226814819	1.04691506
L257	“addr256”	-1.578905011	-1.795715819
L258	“addr257”	0.131810601	-1.158269937
L259	“addr258”	-0.698321224	0.309209241
L260	“addr259”	-0.119690728	-0.819622134
L261	“addr260”	-0.498402395	1.515839113
L262	“addr261”	-0.937948817	1.562289956
L263	“addr262”	0.344524744	-1.156378006
L264	“addr263”	-1.424350807	0.920115878
L265	“addr264”	-1.171919607	1.966884582
L266	“addr265”	-0.905641841	0.569072157
L267	“addr266”	0.390309598	0.627913396
L268	“addr267”	-0.322291168	0.960007232
L269	“addr268”	0.575974444	-1.944039752
L270	“addr269”	-1.828799638	1.610325514
L271	“addr270”	1.700439358	0.971395625
L272	“addr271”	-0.974966678	-1.221675722
L273	“addr272”	1.706583805	0.008139858
L274	“addr273”	1.82302194	-0.237099938
L275	“addr274”	1.632833456	0.76640797
L276	“addr275”	-0.402022473	-1.848010528
L277	“addr276”	1.458056357	-0.78828899
L278	“addr277”	-0.079984208	-1.090485013
L279	“addr278”	-0.438271659	-0.23503782
L280	“addr279”	1.779391585	-0.117987988
L281	“addr280”	-1.36457382	-1.817557336
L282	“addr281”	1.445626353	-0.877479128
L283	“addr282”	-1.395634454	-1.796711465
L284	“addr283”	0.769846201	1.370732128
L285	“addr284”	0.965354194	0.946065871
L286	“addr285”	-0.192595123	-1.905218554
L287	“addr286”	-1.07980966	-0.835092325
L288	“addr287”	0.173025295	1.691495962
L289	“addr288”	-0.218809789	0.729592675
L290	“addr289”	-1.799155548	1.758299011
L291	“addr290”	0.34818435	1.072852059
L292	“addr291”	-0.233091258	-0.208611447
L293	“addr292”	0.303982758	-1.14790321
L294	“addr293”	1.541817258	0.231209461
L295	“addr294”	1.147958267	-1.298336696
L296	“addr295”	0.476371331	-1.532353516
L297	“addr296”	-1.21493883	1.705583539
L298	“addr297”	-1.516758416	-1.766229157
L299	“addr298”	0.610229179	0.099789706
L300	“addr299”	-1.43717707	-0.241620826

Table G.4:U250 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L301	“addr300”	-1.224185488	-0.283428377
L302	“addr301”	-1.968398742	-1.886028211
L303	“addr302”	-1.697691933	-0.186990457
L304	“addr303”	-1.953485603	-1.611779761
L305	“addr304”	-0.209706866	1.179013383
L306	“addr305”	-1.721055626	-1.376616523
L307	“addr306”	-0.285778421	1.799503933
L308	“addr307”	-0.622864964	0.90953638
L309	“addr308”	0.626257584	-1.1508221
L310	“addr309”	1.765985325	1.583881122
L311	“addr310”	-0.928906129	-1.57279364
L312	“addr311”	-0.631674976	-0.820182024
L313	“addr312”	0.61029931	0.193121475
L314	“addr313”	0.991934838	-0.414140634
L315	“addr314”	-0.9225868	0.944051203
L316	“addr315”	-1.588269687	-0.184124568
L317	“addr316”	-0.967340726	1.177371519
L318	“addr317”	-1.413315443	-1.14143297
L319	“addr318”	0.358807723	1.811748269
L320	“addr319”	-0.013778021	0.265468227
L321	“addr320”	-1.96263179	0.493448569
L322	“addr321”	-0.339030884	0.453185645
L323	“addr322”	-1.633042829	-1.706291362
L324	“addr323”	0.477908074	-0.863010276
L325	“addr324”	-0.766665598	0.106850322
L326	“addr325”	-1.149867783	-0.120722508
L327	“addr326”	-1.486736707	-0.312256058
L328	“addr327”	0.326560179	1.42783448
L329	“addr328”	1.292004097	-0.978875885
L330	“addr329”	1.567121338	-1.664772116
L331	“addr330”	1.296905688	1.594167587
L332	“addr331”	0.514081947	0.292071332
L333	“addr332”	-1.202781508	0.147652479
L334	“addr333”	-0.0196642	-0.326694197
L335	“addr334”	-1.80709339	-0.390079037
L336	“addr335”	1.410643196	1.988048783
L337	“addr336”	-0.387106707	-0.828223738
L338	“addr337”	1.231953117	0.239969833
L339	“addr338”	0.065831652	-0.160650224
L340	“addr339”	1.994105075	1.646556645
L341	“addr340”	-0.965853956	1.305880584
L342	“addr341”	-1.587640894	-0.267957758
L343	“addr342”	0.874749416	0.612256096
L344	“addr343”	-1.541638741	-1.499830448
L345	“addr344”	1.987494639	0.820068901
L346	“addr345”	-0.193693319	-1.790622359
L347	“addr346”	0.435076111	0.799850547
L348	“addr347”	-1.773779805	0.214396857
L349	“addr348”	1.545754169	-0.758950101
L350	“addr349”	0.926428019	1.404650823
L351	“addr350”	-0.99707017	-0.88077002
L352	“addr351”	-1.617546445	-1.038735505
L353	“addr352”	-0.976744668	-0.890137667
L354	“addr353”	0.849724602	1.470888704
L355	“addr354”	1.447620037	1.817005866
L356	“addr355”	-1.457364863	-1.124046636
L357	“addr356”	-0.08298951	-0.810937619
L358	“addr357”	1.372662361	1.307371224
L359	“addr358”	1.777896236	1.122707093
L360	“addr359”	-0.826603738	-0.462881128
L361	“addr360”	0.95945193	-1.9900005
L362	“addr361”	0.500493211	-0.395473471
L363	“addr362”	-0.212418272	-0.137415238
L364	“addr363”	1.379722429	1.111518148
L365	“addr364”	0.548366996	1.650804668
L366	“addr365”	0.786432229	-1.251554564
L367	“addr366”	-1.41378295	-1.375559613
L368	“addr367”	1.174915177	1.8035681
L369	“addr368”	1.819656081	0.63571702
L370	“addr369”	-1.963688288	-1.274360489
L371	“addr370”	0.780338382	-1.135893342
L372	“addr371”	0.248845398	-0.749540107
L373	“addr372”	-1.371462325	1.846372505
L374	“addr373”	0.894339563	1.168669237
L375	“addr374”	-1.928984686	0.344201551

Table G.4:U250 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L376	“addr375”	0.250123072	0.503456107
L377	“addr376”	-0.295174639	-1.664318393
L378	“addr377”	0.825586185	0.683265312
L379	“addr378”	-1.966726305	0.295784076
L380	“addr379”	0.597807291	-1.7640064
L381	“addr380”	-0.752097583	0.934918775
L382	“addr381”	1.010824383	1.59756958
L383	“addr382”	0.843028221	-1.306464655
L384	“addr383”	0.838282887	-0.180781671
L385	“addr384”	-0.772981109	0.544650823
L386	“addr385”	1.669348452	-0.725879834
L387	“addr386”	-0.282850421	-0.964910535
L388	“addr387”	1.562927449	0.905576767
L389	“addr388”	1.652535102	-0.999872567
L390	“addr389”	0.325396258	-1.874794331
L391	“addr390”	-0.167708234	-0.984859719
L392	“addr391”	-1.097804819	0.161089385
L393	“addr392”	-1.916261515	-0.598240424
L394	“addr393”	-1.466199879	-0.527786747
L395	“addr394”	-0.511896569	-1.515051895
L396	“addr395”	0.611740841	0.786929787
L397	“addr396”	0.638856883	-0.198688123
L398	“addr397”	-0.419509029	1.389596246
L399	“addr398”	-1.719549861	1.087872651
L400	“addr399”	0.766210694	1.470936592
L401	“addr400”	-0.227105667	-0.865568939
L402	“addr401”	-1.326518676	1.674085749
L403	“addr402”	-1.949729949	-1.03844479
L404	“addr403”	-0.923676476	0.409681179
L405	“addr404”	-0.911085444	1.290138985
L406	“addr405”	0.139951592	0.07977111
L407	“addr406”	-0.62630944	-1.102622903
L408	“addr407”	-0.155131627	1.041717489
L409	“addr408”	-1.551604198	0.207912731
L410	“addr409”	1.145064928	0.959675136
L411	“addr410”	1.359028819	1.347818262
L412	“addr411”	-1.422757096	1.132638293
L413	“addr412”	1.646283941	1.892977471
L414	“addr413”	-1.55186454	-1.545552172
L415	“addr414”	1.51630266	0.672341047
L416	“addr415”	1.262157849	-0.147436908
L417	“addr416”	1.916367133	0.240437733
L418	“addr417”	-0.288224744	-0.631195395
L419	“addr418”	1.464875755	0.012424673
L420	“addr419”	-1.661867351	-1.513317158
L421	“addr420”	-0.467401889	0.558871365
L422	“addr421”	1.87964503	-0.123666372
L423	“addr422”	1.244863014	-0.81362519
L424	“addr423”	0.361406949	-0.580514004
L425	“addr424”	-1.228011453	1.684011014
L426	“addr425”	0.138432289	1.193297111
L427	“addr426”	1.997920481	-1.005466142
L428	“addr427”	-0.609106802	1.033291495
L429	“addr428”	0.504700816	0.573139075
L430	“addr429”	-0.020315303	-1.637947925
L431	“addr430”	1.783044878	-1.67448363
L432	“addr431”	1.105794991	-0.838286818
L433	“addr432”	-0.323038112	0.27082023
L434	“addr433”	1.877120255	1.28471471
L435	“addr434”	-1.706991512	1.45409492
L436	“addr435”	1.671516856	-1.261319003
L437	“addr436”	1.95242463	-0.863403214
L438	“addr437”	1.095042198	0.147694747
L439	“addr438”	1.23705271	1.921361414
L440	“addr439”	0.369489845	0.283643861
L441	“addr440”	-1.222398731	0.22527204
L442	“addr441”	-1.856518183	0.377722905
L443	“addr442”	-1.844532054	-1.612211246
L444	“addr443”	0.006537825	-1.136420934
L445	“addr444”	1.136994413	0.710406848
L446	“addr445”	0.39569503	1.269998292
L447	“addr446”	-1.56831839	-1.150182046
L448	“addr447”	0.119410782	-1.827566909
L449	“addr448”	1.263549437	-1.758346781
L450	“addr449”	0.383637129	1.429079734

Table G.4:U250 Location Table *(continued)*

ID	Address	x-coordinate	y-coordinate
L451	“addr450”	1.267482045	-1.38940442
L452	“addr451”	0.990107661	-0.631380238
L453	“addr452”	0.570055564	0.126551882
L454	“addr453”	1.287041635	-0.836854062
L455	“addr454”	-1.612998106	-1.604513535
L456	“addr455”	1.487616645	1.311199235
L457	“addr456”	-0.93288205	-1.772583162
L458	“addr457”	1.131727292	0.97225017
L459	“addr458”	-0.764913982	-1.997012807
L460	“addr459”	-1.248304634	1.835364025
L461	“addr460”	-1.466672426	1.463933205
L462	“addr461”	0.212758374	0.27119416
L463	“addr462”	-0.564330934	1.881664569
L464	“addr463”	-1.265682105	-1.292143033
L465	“addr464”	0.703253435	-1.187906849
L466	“addr465”	1.258084055	1.487645435
L467	“addr466”	-1.72421121	-1.97625535
L468	“addr467”	-0.385587688	1.902945172
L469	“addr468”	-1.261118155	-1.309905676
L470	“addr469”	0.848280664	1.676124747
L471	“addr470”	-0.019142154	0.565977175
L472	“addr471”	-0.898485428	1.839575831
L473	“addr472”	1.250128191	-0.378173788
L474	“addr473”	-0.391976977	-1.211513087
L475	“addr474”	-1.453179144	-0.511599698
L476	“addr475”	1.643962929	-0.779660733
L477	“addr476”	1.130045521	0.068569704
L478	“addr477”	-0.315090921	0.741521308
L479	“addr478”	-1.105178098	1.801631036
L480	“addr479”	-1.442740913	-0.887137369
L481	“addr480”	-1.529643837	0.319534353
L482	“addr481”	-0.968737017	-1.577704063
L483	“addr482”	0.114005542	1.613305397
L484	“addr483”	-0.911780304	0.914260976
L485	“addr484”	-0.177070471	-1.668141636
L486	“addr485”	1.004541577	-1.228128814
L487	“addr486”	-0.441921291	1.464294888
L488	“addr487”	-0.569456073	0.839269668
L489	“addr488”	1.072833485	-1.589303122
L490	“addr489”	-1.198030062	-0.911992548
L491	“addr490”	1.561256607	0.325322432
L492	“addr491”	1.70742293	0.572683586
L493	“addr492”	-1.811418093	0.620022689
L494	“addr493”	-0.754970564	0.687492042
L495	“addr494”	-0.460781203	0.64460425
L496	“addr495”	-0.063185948	0.726727819
L497	“addr496”	-0.884795109	-0.185234852
L498	“addr497”	-0.57057442	1.06870985
L499	“addr498”	1.240700004	-0.162857746
L500	“addr499”	-1.004993202	-1.256014952
L501	“addr500”	-0.924233637	0.184911774

Table G.5: U250 Request Table

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R001	C001	L002	L005	5/27/2003	7:05	P	5	1	6:59	7:09
R002	C002	L003	L006	5/27/2003	7:16	P	5	1	7:10	7:20
R003	C003	L004	L007	5/27/2003	7:25	P	5	0	7:20	7:30
R004	C004	L008	L011	5/27/2003	9:12	D	5	1	9:06	9:16
R005	C005	L009	L012	5/27/2003	9:27	D	5	1	9:21	9:31
R006	C006	L010	L013	5/27/2003	8:55	P	5	0	8:50	9:00
R007	C007	L014	L017	5/27/2003	10:30	D	5	1	10:25	10:35
R008	C008	L015	L018	5/27/2003	10:41	D	5	1	10:35	10:45
R009	C009	L016	L019	5/27/2003	10:21	P	5	0	10:16	10:26
R010	C010	L020	L023	5/27/2003	11:24	D	5	1	11:18	11:28
R011	C011	L021	L024	5/27/2003	11:05	P	5	1	11:00	11:10
R012	C012	L022	L025	5/27/2003	11:46	D	5	0	11:40	11:50
R013	C013	L026	L029	5/27/2003	12:28	D	5	1	12:22	12:32
R014	C014	L027	L030	5/27/2003	12:40	D	5	1	12:34	12:44
R015	C015	L028	L031	5/27/2003	12:19	P	5	0	12:13	12:23
R016	C016	L032	L035	5/27/2003	13:39	D	5	1	13:33	13:43
R017	C017	L033	L036	5/27/2003	13:57	D	5	1	13:52	14:02
R018	C018	L034	L037	5/27/2003	14:11	D	5	0	14:06	14:16
R019	C019	L038	L039	5/27/2003	14:25	P	5	1	14:20	14:30
R020	C020	L040	L043	5/27/2003	7:08	P	5	1	7:02	7:12
R021	C021	L041	L044	5/27/2003	7:54	D	5	0	7:49	7:59
R022	C022	L042	L045	5/27/2003	7:35	P	5	1	7:29	7:39
R023	C023	L046	L049	5/27/2003	8:49	D	5	1	8:43	8:53
R024	C024	L047	L050	5/27/2003	9:03	D	5	0	8:57	9:07
R025	C025	L048	L051	5/27/2003	8:35	P	5	1	8:30	8:40
R026	C026	L052	L055	5/27/2003	9:27	P	5	1	9:21	9:31
R027	C027	L053	L056	5/27/2003	9:41	P	5	0	9:36	9:46
R028	C028	L054	L057	5/27/2003	9:56	P	5	1	9:50	10:00
R029	C029	L058	L061	5/27/2003	10:41	P	5	1	10:36	10:46
R030	C030	L059	L062	5/27/2003	10:56	P	5	0	10:51	11:01
R031	C031	L060	L063	5/27/2003	11:03	P	5	1	10:58	11:08
R032	C032	L064	L067	5/27/2003	11:49	P	5	1	11:43	11:53
R033	C033	L065	L068	5/27/2003	12:02	P	5	0	11:56	12:06
R034	C034	L066	L069	5/27/2003	12:10	P	5	1	12:04	12:14
R035	C035	L070	L073	5/27/2003	12:49	P	5	1	12:43	12:53
R036	C036	L071	L074	5/27/2003	13:33	D	5	0	13:27	13:37
R037	C037	L072	L075	5/27/2003	13:09	P	5	1	13:03	13:13
R038	C038	L076	L077	5/27/2003	14:08	D	5	1	14:03	14:13
R039	C039	L078	L081	5/27/2003	7:54	D	5	1	7:48	7:58
R040	C040	L079	L082	5/27/2003	7:20	P	5	0	7:15	7:25
R041	C041	L080	L083	5/27/2003	7:38	P	5	1	7:33	7:43
R042	C042	L084	L087	5/27/2003	8:25	P	5	1	8:20	8:30
R043	C043	L085	L088	5/27/2003	9:15	D	5	0	9:10	9:20
R044	C044	L086	L089	5/27/2003	8:54	P	5	1	8:48	8:58
R045	C045	L090	L093	5/27/2003	9:42	P	5	1	9:37	9:47
R046	C046	L091	L094	5/27/2003	10:28	D	5	0	10:22	10:32
R047	C047	L092	L095	5/27/2003	10:02	P	5	1	9:56	10:06
R048	C048	L096	L099	5/27/2003	11:19	D	5	1	11:14	11:24
R049	C049	L097	L100	5/27/2003	11:35	D	5	0	11:29	11:39
R050	C050	L098	L101	5/27/2003	11:08	P	5	1	11:03	11:13
R051	C051	L102	L105	5/27/2003	12:44	D	5	1	12:39	12:49
R052	C052	L103	L106	5/27/2003	12:12	P	5	0	12:06	12:16
R053	C053	L104	L107	5/27/2003	12:28	P	5	1	12:22	12:32
R054	C054	L108	L111	5/27/2003	13:48	D	5	1	13:42	13:52
R055	C055	L109	L112	5/27/2003	13:26	P	5	0	13:21	13:31
R056	C056	L110	L113	5/27/2003	14:11	D	5	1	14:05	14:15
R057	C057	L114	L115	5/27/2003	14:34	D	5	1	14:29	14:39
R058	C058	L116	L119	5/27/2003	7:33	D	5	1	7:28	7:38
R059	C059	L117	L120	5/27/2003	7:44	D	5	1	7:39	7:49
R060	C060	L118	L121	5/27/2003	7:22	P	5	0	7:16	7:26
R061	C061	L122	L125	5/27/2003	8:43	D	5	1	8:37	8:47
R062	C062	L123	L126	5/27/2003	8:53	D	5	1	8:47	8:57
R063	C063	L124	L127	5/27/2003	8:37	P	5	0	8:31	8:41
R064	C064	L128	L131	5/27/2003	9:47	D	5	1	9:42	9:52
R065	C065	L129	L132	5/27/2003	9:28	P	5	1	9:22	9:32

Table G.5: U250 Request Table *(continued)*

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R066	C066	L130	L133	5/27/2003	9:38	P	5	0	9:32	9:42
R067	C067	L134	L137	5/27/2003	10:39	D	5	1	10:33	10:43
R068	C068	L135	L138	5/27/2003	10:22	P	5	1	10:16	10:26
R069	C069	L136	L139	5/27/2003	10:32	P	5	0	10:27	10:37
R070	C070	L140	L143	5/27/2003	11:53	D	5	1	11:48	11:58
R071	C071	L141	L144	5/27/2003	11:28	P	5	1	11:22	11:32
R072	C072	L142	L145	5/27/2003	11:43	P	5	0	11:37	11:47
R073	C073	L146	L149	5/27/2003	12:56	D	5	1	12:51	13:01
R074	C074	L147	L150	5/27/2003	13:05	D	5	1	13:00	13:10
R075	C075	L148	L151	5/27/2003	12:47	P	5	0	12:42	12:52
R076	C076	L152	L153	5/27/2003	13:37	P	5	1	13:32	13:42
R077	C077	L154	L157	5/27/2003	7:42	D	5	1	7:36	7:46
R078	C078	L155	L158	5/27/2003	7:50	D	5	0	7:44	7:54
R079	C079	L156	L159	5/27/2003	7:57	D	5	1	7:51	8:01
R080	C080	L160	L163	5/27/2003	8:43	D	5	1	8:38	8:48
R081	C081	L161	L164	5/27/2003	8:57	D	5	0	8:51	9:01
R082	C082	L162	L165	5/27/2003	8:33	P	5	1	8:27	8:37
R083	C083	L166	L169	5/27/2003	9:26	P	5	1	9:21	9:31
R084	C084	L167	L170	5/27/2003	9:41	P	5	0	9:35	9:45
R085	C085	L168	L171	5/27/2003	10:26	D	5	1	10:20	10:30
R086	C086	L172	L175	5/27/2003	10:35	P	5	1	10:29	10:39
R087	C087	L173	L176	5/27/2003	11:29	D	5	0	11:24	11:34
R088	C088	L174	L177	5/27/2003	11:04	P	5	1	10:58	11:08
R089	C089	L178	L181	5/27/2003	12:22	D	5	1	12:16	12:26
R090	C090	L179	L182	5/27/2003	12:01	P	5	0	11:56	12:06
R091	C091	L180	L183	5/27/2003	12:42	D	5	1	12:37	12:47
R092	C092	L184	L187	5/27/2003	12:57	P	5	1	12:51	13:01
R093	C093	L185	L188	5/27/2003	13:37	D	5	0	13:32	13:42
R094	C094	L186	L189	5/27/2003	13:50	D	5	1	13:44	13:54
R095	C095	L190	L191	5/27/2003	14:09	D	5	1	14:04	14:14
R096	C096	L192	L195	5/27/2003	7:40	D	5	1	7:35	7:45
R097	C097	L193	L196	5/27/2003	7:15	P	5	0	7:09	7:19
R098	C098	L194	L197	5/27/2003	7:59	D	5	1	7:54	8:04
R099	C099	L198	L201	5/27/2003	8:33	D	5	1	8:27	8:37
R100	C100	L199	L202	5/27/2003	8:44	D	5	0	8:39	8:49
R101	C101	L200	L203	5/27/2003	8:24	P	5	1	8:19	8:29
R102	C102	L204	L207	5/27/2003	9:06	P	5	1	9:00	9:10
R103	C103	L205	L208	5/27/2003	9:46	D	5	0	9:40	9:50
R104	C104	L206	L209	5/27/2003	9:59	D	5	1	9:54	10:04
R105	C105	L210	L213	5/27/2003	10:40	D	5	1	10:35	10:45
R106	C106	L211	L214	5/27/2003	10:25	P	5	0	10:20	10:30
R107	C107	L212	L215	5/27/2003	11:08	D	5	1	11:03	11:13
R108	C108	L216	L219	5/27/2003	11:24	P	5	1	11:19	11:29
R109	C109	L217	L220	5/27/2003	12:03	D	5	0	11:58	12:08
R110	C110	L218	L221	5/27/2003	11:42	P	5	1	11:36	11:46
R111	C111	L222	L225	5/27/2003	12:53	D	5	1	12:48	12:58
R112	C112	L223	L226	5/27/2003	12:35	P	5	0	12:29	12:39
R113	C113	L224	L227	5/27/2003	13:20	D	5	1	13:15	13:25
R114	C114	L228	L229	5/27/2003	13:32	P	5	1	13:26	13:36
R115	C115	L230	L233	5/27/2003	7:36	D	5	1	7:31	7:41
R116	C116	L231	L234	5/27/2003	7:45	D	5	1	7:39	7:49
R117	C117	L232	L235	5/27/2003	7:24	P	5	0	7:19	7:29
R118	C118	L236	L239	5/27/2003	8:07	P	5	1	8:01	8:11
R119	C119	L237	L240	5/27/2003	8:52	D	5	1	8:47	8:57
R120	C120	L238	L241	5/27/2003	8:29	P	5	0	8:24	8:34
R121	C121	L242	L245	5/27/2003	9:34	D	5	1	9:29	9:39
R122	C122	L243	L246	5/27/2003	9:14	P	5	1	9:09	9:19
R123	C123	L244	L247	5/27/2003	9:25	P	5	0	9:19	9:29
R124	C124	L248	L251	5/27/2003	10:34	D	5	1	10:29	10:39
R125	C125	L249	L252	5/27/2003	10:47	D	5	1	10:42	10:52
R126	C126	L250	L253	5/27/2003	10:53	D	5	0	10:47	10:57
R127	C127	L254	L257	5/27/2003	11:50	D	5	1	11:45	11:55
R128	C128	L255	L258	5/27/2003	12:01	D	5	1	11:55	12:05
R129	C129	L256	L259	5/27/2003	12:11	D	5	0	12:06	12:16
R130	C130	L260	L263	5/27/2003	12:20	P	5	1	12:14	12:24

Table G.5: U250 Request Table *(continued)*

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R131	C131	L261	L264	5/27/2003	13:05	D	5	1	13:00	13:10
R132	C132	L262	L265	5/27/2003	12:38	P	5	0	12:33	12:43
R133	C133	L266	L267	5/27/2003	13:32	D	5	1	13:26	13:36
R134	C134	L268	L271	5/27/2003	7:51	D	5	1	7:45	7:55
R135	C135	L269	L272	5/27/2003	7:17	P	5	0	7:12	7:22
R136	C136	L270	L273	5/27/2003	7:35	P	5	1	7:30	7:40
R137	C137	L274	L277	5/27/2003	9:00	D	5	1	8:55	9:05
R138	C138	L275	L278	5/27/2003	9:10	D	5	0	9:04	9:14
R139	C139	L276	L279	5/27/2003	9:18	D	5	1	9:12	9:22
R140	C140	L280	L283	5/27/2003	9:29	P	5	1	9:24	9:34
R141	C141	L281	L284	5/27/2003	9:45	P	5	0	9:40	9:50
R142	C142	L282	L285	5/27/2003	10:36	D	5	1	10:30	10:40
R143	C143	L286	L289	5/27/2003	11:21	D	5	1	11:15	11:25
R144	C144	L287	L290	5/27/2003	11:32	D	5	0	11:26	11:36
R145	C145	L288	L291	5/27/2003	11:13	P	5	1	11:07	11:17
R146	C146	L292	L295	5/27/2003	12:21	D	5	1	12:16	12:26
R147	C147	L293	L296	5/27/2003	12:01	P	5	0	11:55	12:05
R148	C148	L294	L297	5/27/2003	12:44	D	5	1	12:39	12:49
R149	C149	L298	L301	5/27/2003	13:30	D	5	1	13:24	13:34
R150	C150	L299	L302	5/27/2003	13:13	P	5	0	13:08	13:18
R151	C151	L300	L303	5/27/2003	13:24	P	5	1	13:19	13:29
R152	C152	L304	L305	5/27/2003	14:15	D	5	1	14:09	14:19
R153	C153	L306	L309	5/27/2003	7:42	D	5	1	7:37	7:47
R154	C154	L307	L310	5/27/2003	7:22	P	5	0	7:17	7:27
R155	C155	L308	L311	5/27/2003	8:13	D	5	1	8:08	8:18
R156	C156	L312	L315	5/27/2003	8:50	D	5	1	8:44	8:54
R157	C157	L313	L316	5/27/2003	8:31	P	5	0	8:25	8:35
R158	C158	L314	L317	5/27/2003	8:38	P	5	1	8:32	8:42
R159	C159	L318	L321	5/27/2003	9:56	D	5	1	9:51	10:01
R160	C160	L319	L322	5/27/2003	10:06	D	5	0	10:01	10:11
R161	C161	L320	L323	5/27/2003	10:19	D	5	1	10:13	10:23
R162	C162	L324	L327	5/27/2003	10:31	P	5	1	10:25	10:35
R163	C163	L325	L328	5/27/2003	10:40	P	5	0	10:35	10:45
R164	C164	L326	L329	5/27/2003	11:18	D	5	1	11:13	11:23
R165	C165	L330	L333	5/27/2003	11:25	P	5	1	11:20	11:30
R166	C166	L331	L334	5/27/2003	11:40	P	5	0	11:35	11:45
R167	C167	L332	L335	5/27/2003	12:19	D	5	1	12:14	12:24
R168	C168	L336	L339	5/27/2003	12:36	P	5	1	12:31	12:41
R169	C169	L337	L340	5/27/2003	13:24	D	5	0	13:18	13:28
R170	C170	L338	L341	5/27/2003	13:02	P	5	1	12:56	13:06
R171	C171	L342	L343	5/27/2003	14:00	D	5	1	13:55	14:05
R172	C172	L344	L347	5/27/2003	7:06	P	5	1	7:01	7:11
R173	C173	L345	L348	5/27/2003	7:24	P	5	1	7:19	7:29
R174	C174	L346	L349	5/27/2003	7:39	P	5	0	7:34	7:44
R175	C175	L350	L353	5/27/2003	8:31	P	5	1	8:26	8:36
R176	C176	L351	L354	5/27/2003	8:45	P	5	1	8:40	8:50
R177	C177	L352	L355	5/27/2003	9:20	D	5	0	9:15	9:25
R178	C178	L356	L359	5/27/2003	9:38	P	5	1	9:32	9:42
R179	C179	L357	L360	5/27/2003	9:47	P	5	1	9:41	9:51
R180	C180	L358	L361	5/27/2003	10:32	D	5	0	10:27	10:37
R181	C181	L362	L365	5/27/2003	11:08	D	5	1	11:03	11:13
R182	C182	L363	L366	5/27/2003	10:49	P	5	1	10:44	10:54
R183	C183	L364	L367	5/27/2003	11:00	P	5	0	10:55	11:05
R184	C184	L368	L371	5/27/2003	11:51	P	5	1	11:46	11:56
R185	C185	L369	L372	5/27/2003	12:00	P	5	1	11:55	12:05
R186	C186	L370	L373	5/27/2003	12:52	D	5	0	12:47	12:57
R187	C187	L374	L377	5/27/2003	13:41	D	5	1	13:36	13:46
R188	C188	L375	L378	5/27/2003	13:18	P	5	1	13:13	13:23
R189	C189	L376	L379	5/27/2003	14:08	D	5	0	14:02	14:12
R190	C190	L380	L381	5/27/2003	14:36	D	5	1	14:31	14:41
R191	C191	L382	L385	5/27/2003	7:38	D	5	1	7:33	7:43
R192	C192	L383	L386	5/27/2003	7:51	D	5	0	7:46	7:56
R193	C193	L384	L387	5/27/2003	7:28	P	5	1	7:22	7:32
R194	C194	L388	L391	5/27/2003	8:44	D	5	1	8:38	8:48
R195	C195	L389	L392	5/27/2003	8:53	D	5	0	8:48	8:58

Table G.5: U250 Request Table *(continued)*

ID	Passenger	From	To	Date	Time	Type	Loading Time	Load	L-TW	R-TW
R196	C196	L390	L393	5/27/2003	9:01	D	5	1	8:56	9:06
R197	C197	L394	L397	5/27/2003	9:08	P	5	1	9:02	9:12
R198	C198	L395	L398	5/27/2003	9:17	P	5	0	9:11	9:21
R199	C199	L396	L399	5/27/2003	9:30	P	5	1	9:24	9:34
R200	C200	L400	L403	5/27/2003	10:10	P	5	1	10:04	10:14
R201	C201	L401	L404	5/27/2003	10:22	P	5	0	10:17	10:27
R202	C202	L402	L405	5/27/2003	11:07	D	5	1	11:02	11:12
R203	C203	L406	L409	5/27/2003	11:17	P	5	1	11:11	11:21
R204	C204	L407	L410	5/27/2003	12:01	D	5	0	11:55	12:05
R205	C205	L408	L411	5/27/2003	12:07	D	5	1	12:02	12:12
R206	C206	L412	L415	5/27/2003	12:21	P	5	1	12:15	12:25
R207	C207	L413	L416	5/27/2003	13:18	D	5	0	13:13	13:23
R208	C208	L414	L417	5/27/2003	12:54	P	5	1	12:49	12:59
R209	C209	L418	L419	5/27/2003	13:38	P	5	1	13:32	13:42
R210	C210	L420	L423	5/27/2003	7:07	P	5	1	7:01	7:11
R211	C211	L421	L424	5/27/2003	7:47	D	5	0	7:41	7:51
R212	C212	L422	L425	5/27/2003	7:31	P	5	1	7:26	7:36
R213	C213	L426	L429	5/27/2003	8:47	D	5	1	8:41	8:51
R214	C214	L427	L430	5/27/2003	8:58	D	5	0	8:53	9:03
R215	C215	L428	L431	5/27/2003	9:09	D	5	1	9:03	9:13
R216	C216	L432	L435	5/27/2003	9:56	D	5	1	9:50	10:00
R217	C217	L433	L436	5/27/2003	9:28	P	5	0	9:22	9:32
R218	C218	L434	L437	5/27/2003	9:40	P	5	1	9:34	9:44
R219	C219	L438	L441	5/27/2003	11:00	D	5	1	10:54	11:04
R220	C220	L439	L442	5/27/2003	10:39	P	5	0	10:34	10:44
R221	C221	L440	L443	5/27/2003	11:18	D	5	1	11:12	11:22
R222	C222	L444	L447	5/27/2003	11:28	P	5	1	11:23	11:33
R223	C223	L445	L448	5/27/2003	11:40	P	5	0	11:34	11:44
R224	C224	L446	L449	5/27/2003	12:21	D	5	1	12:15	12:25
R225	C225	L450	L453	5/27/2003	13:05	D	5	1	12:59	13:09
R226	C226	L451	L454	5/27/2003	12:50	P	5	0	12:44	12:54
R227	C227	L452	L455	5/27/2003	13:27	D	5	1	13:22	13:32
R228	C228	L456	L457	5/27/2003	13:45	P	5	1	13:40	13:50
R229	C229	L458	L461	5/27/2003	7:43	D	5	1	7:37	7:47
R230	C230	L459	L462	5/27/2003	7:20	P	5	1	7:15	7:25
R231	C231	L460	L463	5/27/2003	7:37	P	5	0	7:31	7:41
R232	C232	L464	L467	5/27/2003	8:19	P	5	1	8:14	8:24
R233	C233	L465	L468	5/27/2003	8:30	P	5	1	8:25	8:35
R234	C234	L466	L469	5/27/2003	8:43	P	5	0	8:38	8:48
R235	C235	L470	L473	5/27/2003	10:23	D	5	1	10:18	10:28
R236	C236	L471	L474	5/27/2003	10:00	P	5	1	9:54	10:04
R237	C237	L472	L475	5/27/2003	10:43	D	5	0	10:37	10:47
R238	C238	L476	L479	5/27/2003	11:24	D	5	1	11:18	11:28
R239	C239	L477	L480	5/27/2003	11:37	D	5	1	11:31	11:41
R240	C240	L478	L481	5/27/2003	11:15	P	5	0	11:09	11:19
R241	C241	L482	L485	5/27/2003	12:33	D	5	1	12:28	12:38
R242	C242	L483	L486	5/27/2003	12:12	P	5	1	12:06	12:16
R243	C243	L484	L487	5/27/2003	12:20	P	5	0	12:15	12:25
R244	C244	L488	L491	5/27/2003	13:03	P	5	1	12:58	13:08
R245	C245	L489	L492	5/27/2003	13:17	P	5	1	13:12	13:22
R246	C246	L490	L493	5/27/2003	13:29	P	5	0	13:24	13:34
R247	C247	L494	L497	5/27/2003	14:13	P	5	1	14:07	14:17
R248	C248	L495	L498	5/27/2003	14:19	P	5	1	14:13	14:23
R249	C249	L496	L499	5/27/2003	14:54	D	5	0	14:49	14:59
R250	C250	L500	L501	5/27/2003	15:16	D	5	1	15:10	15:20