SOLVING THE DIAL-A-RIDE PROBLEM (DARP) USING AN AGENT BASED SIMULATION APPROACH AND HEURISTIC METHODS

Marc Lucien Fienberg

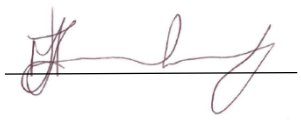
A Dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree Master of Science in Engineering.

Johannesburg, 2016

Declaration

I declare that this dissertation is my own unaided work where otherwise acknowledged. It is being submitted for the Degree Masters in Science in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before, for any other degree or examination at any other University.

Signed on this 16th day of February 2016.



Marc Lucien Fienberg

Student number: 351240

Abstract

The Dial-a-Ride Problem (DARP) requires a set of customers to be transported by a limited fleet of vehicles between unique origins and destinations under several service constraints, most notably, within defined time windows. The problem is considered NP-hard and has typically been solved using metaheuristics methods. An agent based simulation (ABS) model was developed, where each vehicle bids to service customers based on a weighted objective function that considers the cost to service the customer, and time quality of the service that would be achieved. The approach applied a pre-processing technique to reduce the search space given the service time window constraints. Tests of the model show significantly better customer transit and waiting times than the benchmark datasets. The ABS was able to obtain solutions for much larger problem sizes than the benchmark solutions, with this work being the first known application of ABS to the DARP.

Table of Contents

List of Figures

List of Tables

Nomenclature

ABS	Agent Based Simulation
CPU	Central Processing Unit
CVRP	Capacitated Vehicle Routing Problem
CVRPTW	Capacitated Vehicle Routing Problem with Time Windows
DARP	Dial-a-Ride Problem
FFD	First Fit Decreasing
GA	Genetic Algorithm
PDP	Pick and Delivery Problem
SA	Simulated Annealing
SE	String Exchange
SVP	Single Vehicle Problem
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
VRPPD	Vehicle Routing Problem with Pick and Delivery
VRPPDTW	Vehicle Routing Problem with Pick, Delivery and Time Windows

CHAPTER 1: INTRODUCTION

1.1 The Problem And Its Background

Cordeau and Laporte (2007) point out that "there is a genuine need for reliable cost effective" public transport systems in developed countries. Public transport systems are facing increased demand and costs, while forming part of complex transportation networks which need to service a broad demographic of customers. Public transport systems have a direct impact on the economic development and quality of life offered in urban areas, and must provide services for a wide-ranging age demographic. In particular, special transportation services are needed for the elderly, sick, children and the disabled. Without vehicles to provide a door-to-door transportation service for these groups of people, they would otherwise have to rely on family or friends to meet their transportation needs (Hansen, 2010). This paper explores the application of a purpose built agent based model, which must determine how to provide door-to-door transportation of multiple customer requests using a limited fleet of vehicles.

The door-to-door transportation of people or goods forms part of an extensively studied set of vehicle routing problems (VRPs) involving the use of combinatorial optimisation, a sub-field of Operations Research. Most vehicle scheduling and routing problems are a variation of the capacitated vehicle routing problem (CVRP), which generalises the well-known Travelling Salesman Problem (TSP). The TSP aims to determine the shortest route to travel between a number of different locations, required to be visited only once, and returning to the starting point.

The CVRP uses identical vehicles from a single depot, with only vehicle capacity restrictions imposed to minimise a total cost or weighted objective function. A simple variation of the CVRP is the capacitated vehicle routing problem with time windows (CVRPTW). In this case n customers must be serviced within specified time windows $[a_n, b_n]$, which includes vehicle loading times while stopped at a customer location. In the event of an early arrival, a vehicle would have to wait.

Customers may define either soft or hard service time windows. Soft time windows allow for a degree of violation to balance customer service and cost requirements, while hard time windows do not (Toth and Vigo, 2002). The presence of time window constraints make the solution space of a CVRPTW highly non-convex, making it very easy to move from a feasible to a non-feasible solution space.

A further variation of the CVRPTW is the vehicle routing problem with pick-up and deliveries (VRPPD) which allows for mixed servicing of customer requests for the delivery of goods.

Vehicle routes are determined to service specific customer collection and drop-off requests, while considering the customers' goods load sizes and the pick-up vehicles' load capacity. Each transport request is between a single origin and destination, with no transhipment of goods taking place. The same vehicle has to perform both the pick-up and delivery of customer goods. The Dial-a-Ride Problem (DARP) is an extension of the VRPPD, where the loads to be transported are represented by people and is considered as one of the most complex VRPs (Savelsbergh & Sol, 1995). The relationship between the different VRPs is shown in Figure 1.1. All the problem types shown in Figure 1.1 can either be for a single or multi-vehicle scenario, where different vehicle capacities are considered in the problem.

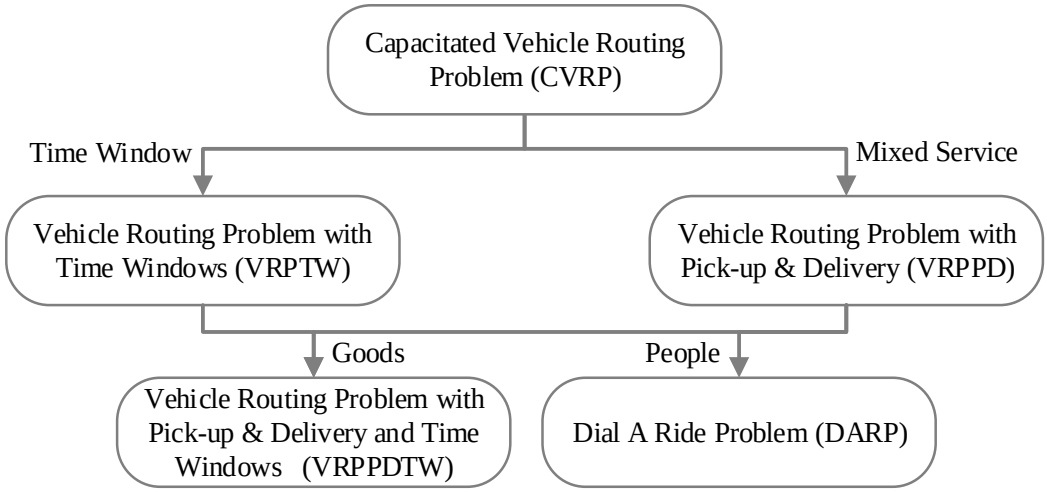


Figure 1.1: The DARP in the VRP structure

Specifically, the DARP occurs when customers phone to request a transport service, where the customers are typically the elderly, sick or the disabled. Like a VRP, the DARP services *n* number of customers with a fleet of *m* vehicles. However, DARP requires vehicle routes and schedules that accommodate requests from customers which have specified where they must be collected from, where they must be dropped off, and times at which it is convenient for them to be picked-up and dropped off. This makes DARP a many-to-many type VRP, where each customer has its own starting and ending route nodes. In some cases, customers may require a return trip between their travel points later in the day (Cordeau and Laporte, 2007).

The DARP is characterised by conflicting objectives. It must minimise transportation costs, like the VRP, while also having to consider minimising user time inconvenience but should also maximise the number of customers serviced per vehicle. For a DARP, requests are typically collected and vehicle routes scheduled by a central planning unit which must determine how to allocate customers to vehicles. The problem is more complicated than just partitioning customers

into clusters, as consideration of the time sequence in which customer requests occur must be accounted for. The DARP is considered to be NP-hard, thus one cannot expect to find a global optimal solution in polynomial time. Rather a robust algorithm that is able to provide a well optimised solution should be sought (Hansen, 2010).

When a DARP is applied to a transportation system, all vehicles start and end at a common depot, with customers typically sharing rides. Three variations of the DARP exist, namely the group taxi system, the static problem and the dynamic problem.

For the group taxi system, customer requests are made directly to the driver, who is on the road. The driver may pick-up other passengers before and during the servicing of a specific customer. This type of problem enables the driver to make his own decisions regarding his route, and customer servicing schedule, with no central planning unit used.

For the static DARP all customer pick-up points, drop-off points and time windows for transportation are known beforehand and do not change. This enables a central planning unit to carefully allocate drivers, routes and schedules prior to the vehicles leaving the depot. A precedence constraint exists, requiring the same vehicle that picks up a customer at their origin to drop-off the customer at their destination. Given that all customer transportation information is known in advance, a well optimised schedule can be obtained, as several hours can be spent on optimising the vehicle route configurations. However, this type of problem does not allow for the inclusion of uncertain events like traffic jams and changes in customer transport requests.

The dynamic DARP receives customer requests throughout the day from a central planning unit, where vehicle routes' are updated to accommodate new requests. The dynamic DARP requires current routes to be only partially optimised, with knowledge of current vehicle loads, locations and planned routes having to be monitored. As customer requests are made, the DARP is re-solved in a short period of time to reschedule customers and vehicles. In some cases, the customer transportation request may be rejected, and the customer needs to be re-scheduled for an alternative day (Colorini and Righini, 2001).

In solving a DARP two decisions primarily need to be made, namely the allocation of customers to vehicles, and the routing of each vehicle. These decisions are typically handled as two sub-problems. The type of algorithm that can be used to solve DARP depends on both the number of customers, and whether the problem is either static or dynamic. The DARP has been widely studied in literature, with typical optimisation techniques used to solve the DARP involving dynamic programming, and custom metaheuristics methods based on tabu search, simulated annealing and genetic algorithm methods.

The applicability of the DARP in daily life is seen by the rise of on demand transport services like Uber, which must provide personalised and efficient transport services. The Finland National Public Health Institute demonstrated how a DARP system in Helsinki could reduce traffic volumes by 50-70%, with 50% of customers seeing it as an attractive transportation option without the use of state subsidies (Tuomisto and Tainio, 2005).This research intends to investigate the use of an alternative optimisation technique known as Agent Based Simulation to solve the DARP.

1.2 Agent Based Simulation

A relatively new and unused optimisation technique in the field of Operations Research, is the use of Agent Based Simulation (ABS). ABS allows for increasingly complex systems to be modelled and requires fewer simplifying assumptions, making the simulation model more representative of reality. ABS decentralises decision making to the model's respective agents, requiring the agents to be able to communicate with one-another to reach a decision. Control is shared amongst the agents with an integrated negotiation process needed to make a final decision (Zeddini *et al*., 2008).

ABS is more flexible than equation based models, and allows for coordination and strategic interactions between agents. An agent in an ABS may have some or all of the following characteristics:

1. An agent can act independently from other agents in its environment,
2. An agent is a discrete individual that may have its own set of behaviours, attributes and decision making ability,
3. An agent has a social ability to interact with other agents. Its social interactions are determined by mechanisms,
4. Each agent has its own local goal that it is trying to achieve, and
5. An agent has the ability to learn and adapt from its experiences (Macal and North, 2009).

The specific role that an agent performs in a simulation depends on the model's objectives. Agents can be either self-goal-directed or reactive to their environment, or both. Based on the defined agent environment, behaviours and attributes will determine inter-agent interactions. Figure 1.2 shows a simple model of how 2 agents would reason and interact with one-another. Depending on the model's application, the links shown in Figure 1.2 between the 2 agents and the environment can be observed (Janssen, 2005).

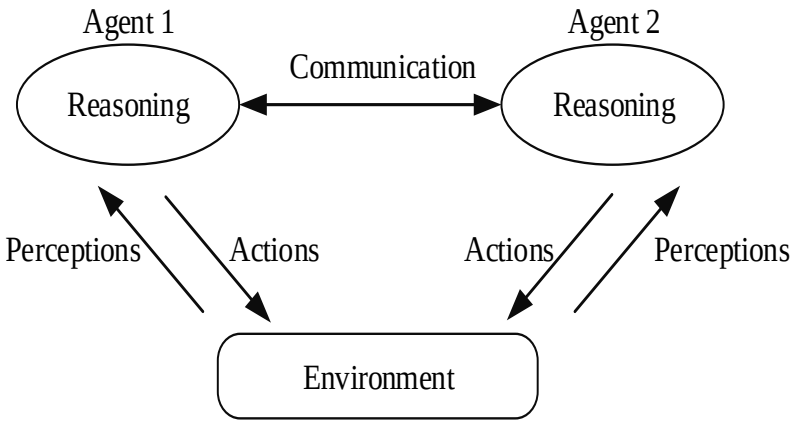


Figure 1.2: Reasoning interactions between two agents and their environment (Janssen, 2005)

ABS has been applied to a broad range of areas from energy analysis, social interactions and crime analysis. As far as the author can ascertain, there are no academic works that have used an agent based simulation approach to solve the DARP. ABS has been used to solve the CVRP (see Zeddini *et al*., *2008* and Vokrinek *et al., 2010*).

1.3 Research Objectives

The objectives of this study are:

1. To develop an agent-based simulation approach to solve the static Dial-a-Ride problem.
2. Evaluate the results generated by the proposed method against conventional algorithms used to solve the Dial-a-Ride problem.

1.4 Data And Assumptions

- This research will only focus on the static DARP, where customers have specified either a desired pick-up or a delivery time.
- The proposed ABS model will be applied to test instances developed by Cordeau and Laporte (2003). The data for the test scenarios are based on realistic assumptions, which use input data regarding time window widths, vehicle capacities, route durations and maximum ride durations gathered from the Montreal Transit Commission.

1.5 Dissertation Layout

The remainder of this dissertation is structured as follows:

Chapter Two presents a literature review which aims to:

- Explore the nature of the DARP within the field of VRP's,
- Investigate different types of heuristic and metaheuristic methods that can be used for vehicle scheduling,
- Investigate other research works conducted on the DARP, and
- Explore the field of agent based simulation.

Chapter Three presents the specific methodology and method used for this research.

The methodology explores:

- The use of simulation as an analysis tool,
- The components needed to develop an ABS framework,
- The types of simulation software available, and
- How the results obtained can be validated.

This chapter goes on to outline the particular method used by explaining:

- How the agent based simulation model was developed,
- The assumptions made and the data used to test the model,
- How the performance of the model will be evaluated, and
- The experimental procedure used.

Chapter Four presents the results obtained from the proposed simulation approach and provides a comparison to other results obtained using the same data sets to solve the DARP.

This chapter concludes by interpreting the results obtained.

Chapter Five presents the conclusions, major findings and recommendations for future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Operation research techniques have been widely applied to numerous types of VRPs which have been extensively studied for over 40 years, resulting in various algorithms being developed which have yielded good solutions to VRPs. However as the problem size and number of variables increases in a VRP so does the computational complexity, giving it a NP-hard classification (Toth and Vigo, 2002).

To construct an agent based simulation model of the DARP, an in-depth understanding of the problem's characteristics and its applications within the field of VRPs is needed. Common methods for solving VRPs and specifically the DARP need to be determined.

2.2 The Dial-A-Ride Problem

There are two typical approaches applied when solving a DARP. The first approach determines an optimal fleet size and capacity configuration to satisfy all customer requests. The second approach aims to determine how many customer requests can be serviced using a fixed fleet size (Cordeau and Laporte, 2007). The DARP forms part of a sub-class of the pick-up and delivery problems (PDPs), which itself comprises of *m* routes that should satisfy the following conditions:

1. All pick-up and delivery requests are fulfilled,
2. Transhipments of goods or people may not occur on the route,
3. The load of a vehicle cannot exceed its capacity, and
4. The objective is to minimise a cost function (Berbeglia, 2007).

DARP is different to the Pick-up and Delivery Problem (PDP) in that it includes time constrained intervals in which the servicing of customers should take place. Thus the objective function of a DARP usually considers a user inconvenience factor to measure the quality of the service offered. This makes the DARP harder to solve than typical transportation problems, as a weighted objective function between cost and time is needed. In the event that strict time windows are used, transportation costs would be high. Similarly, while having well optimised vehicle routes can result in customers having long travelling and waiting times. Therefore a balance between service quality and routing costs must be sought **(**Cordeau and Laporte, 2002).

The cost elements used in the objective function commonly relate to the fixed vehicle costs for each kilometre travelled, and the cost per minute for servicing customers linked to the drivers wages (Hansen, 2010). Service quality can be determined by making a comparison between the

direct and excess ride times a customer spends travelling to his destination, and the time spent waiting before departure (Cordeau and Laporte, 2007). Typically an upper bound on the maximum customer transportation time constraint will exist. Figure 2.1 illustrates the differences in direct and excess ride times, given the time window of $[a_{n+i}, b_{n+i}]$ for the drop-off location, and the time window of $[a_i, b_i]$ for the pick-up location. The direct transportation time is defined as $t_{i, n+i}$ from i to $n + i$. The service time is defined as s_i at i, with an upper limit E, existing on the amount of excess ride time allowed. Ideally a customer would want to be serviced in the minimal time window of $[b_i, a_{n+i}]$, with the maximum permissible service limits being between $[a_i, b_{n+i}]$.

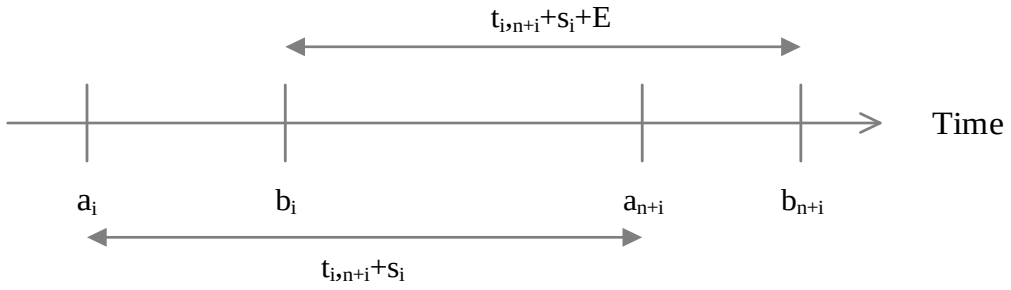


Figure 2.1: Direct and excess customer ride time windows (Jorgensen *et al*., 2007)

The DARP can be further classified into either a single capacity or multi-capacity vehicle problem. The fleet of vehicles can either have homogeneous or heterogeneous features. Features can relate to the vehicles' capacities or possible seating configurations. While most applications of DARP consider a homogenous vehicle fleet, Madsen *et al*. (1995) considered a multidimensional vehicle capacity requirement. Vehicles had a combination of ordinary seats, lying seats, children seats, wheel chair places and a number of bed places. The use of a heterogeneous fleet of vehicles increased the complexity of the DARP, as the seating requirements of customers form an additional strict customer service constraint.

Industry applications of the DARP have been used in Copenhagen and Belgium (see Madsen *et al*., 1995, and Rekiek *et al*., 2006.) to transport the handicapped. In the USA there are approximately 1500 demand response transit systems to service the needs of customers in rural areas, with at least 400 such services existing in urban districts (Transit Cooperative Research Programme, 2009).

Generally in solving a static DARP three key decision are required, namely:

1. Determining how to cluster the customers to be serviced by the same vehicle,
2. Sequencing the servicing of customers into a vehicle route within their assigned cluster, and

3. Scheduling the customer pick-up and drop-off events for each vehicle (Cordeau and Laporte, 2003).

To make these decisions in the DARP, two groups of operations research methods are used, namely:

1. Insertion heuristics, whereby customers are optimally inserted into an initial route, making the solution quality reliant on the sequence in which customers are visited.
2. Cluster-first and route-second heuristics. This approach partitions all the customers into a number of subsets, equivalent to the number of vehicles available to service the customers. Optimal routes are then determined for each specific vehicle to service its customer subset. The quality of the solution obtained relies on the partitioning of customers into vehicle subsets (Baugh *et al*., 1998).

These scheduling methods are not unique to the DARP, with variations of these methods applicable to a number of VRPs. The specific scheduling algorithm used to route vehicles and minimise user inconvenience in a DARP will directly influence the solution obtained.

2.2.1 Vehicle scheduling

The static DARP can have difficulty adhering to pre-established schedules due to time varying factors, such as irregular customer demand or traffic congestion which can affect the execution of a set schedule. In general, given a sequence of $i_1, i_2, \ldots, i_q$ nodes to be visited, the vehicle scheduling problem can be formulated as:

$$Minimise \sum_{i=1}^{q} g_i B_i \qquad (2.1)$$

Such that:

$$B_i + d_i + t_{i,i+1} \leq B_{i+1}, \quad \forall i = 1, \ldots, q-1 \qquad (2.2)$$

$$e_i \leq B_i \leq l_i, \qquad \forall i = 1, \ldots, q \qquad (2.3)$$

Where $g_i B_i$ is a convex function that defines a time window $[e_i, l_i]$ for each customer node $i \in N$. There exists a non-negative load q_i and service duration d_i. Travel time t_{ij} is associated with each arc $(i, j) \in A$ between customer nodes (Cordeau *et al*., 2007).

Within a DARP of *m* vehicles, there exists an equivalent amount of *m* single vehicle routing sub-problems. Each individual vehicle is concerned about how best to service its own customers in its designated customer cluster.

Sexton and Bodin (1985) propose a two-step scheduling method that uses Benders Decomposition to solve a DARP. The first step involves partitioning customers into vehicle clusters. The cluster

are then each solved as a single vehicle routing problem (SVRP), which aims to minimise the total customer schedule inconvenience defined by the function:

Total inconvenience of the schedule = $\sum$ total customer inconvenience (2.4)

Where:

The total customer inconvenience = $\sum$ (excess ride time + delivery time deviation) (2.5)

And:

Excess ride time = (actual ride time – direct ride time) (2.6)

Delivery time deviation = (customer desired delivery time – customer actual delivery time) (2.7)

Using a weighted objective function that minimises customer inconvenience, Sexton and Bodin (1985) found the dual of the scheduling problem can be interpreted as a Network Flow Problem. This type of problem could then be solved very efficiently as a linear programming problem to determine an optimal vehicle schedule.

Once the scheduling sequence has been determined, the actual arrival and departure times of vehicles at customer locations must be established. This is to ensure time window constraints are adhered to and route durations are minimised.

Cordeau and Laporte (2003) present a scheduling programming heuristic that determines the required departure time at which vehicle *k* must leave the depot, to minimise the total route duration and unnecessary waiting time. The heuristic is defined by the variables:

$v_1, \ldots, v_{q-1}$	Each customer vertex, where v_o and v_q represent depots;
T_k	Maximum duration of vehicle *k*;
$[e_i, l_i]$	Time window at the start of service for vertex v_i;
t_{ijk}	Travel time of vehicle *k* from v_i to v_j;
d_i	Service duration at *vi* vehicle *k* ;
T_{aik}	Arrival time of vehicle *k* for customer request *i;*
T_{sik}	Start time of service for customer request *i* by vehicle *k;*
T_{dik}	Departure time from vertex v_i in vehicle *k;*
W_{ik}	Waiting time at v_i for vehicle *k*.[1]

[1] This notation will be used throughout the Literature Review

Where:

$T_{sik} \geq max\{e_i, T_{aik}\}$ (2.8)

$T_{dik} = T_{sik} + d_{ik}$ (2.9)

The arrival of vehicle k should occur at v_i or before e_i, resulting in a waiting time of $W_{ik} = T_{sik} - T_{aik}$ (2.10). If $T_{sik} > l_i$, then the time window for vertex v_i is violated. For a feasible scheduling problem, the earliest time at which a vehicle k leaves the depot is $T_{sok} = e_{oi}$ and $T_{sik} = max\{ei, T_{aik}\}$ for $i = 1,....,q$.

Thus the maximum time delay that can occur before the servicing of a customer can be determined by calculating the forward time slack, F_i of v_i, defined for vehicle k by:

$$\boldsymbol{F_{ik} = min_{i \leq j \leq q}\{l_i - (T_{sik} + \sum_{i \leq j \leq q} W_{ik})\}} \quad (2.11)$$

2.2.2 Mathematical formulation of the DARP

The DARP has been modelled mathematically in numerous works, with each work proposing slight variations of the algorithm and specific objective function used. Typically a DARP consists of n customer requests, where each customer specifies a pick-up location i and drop-off location $n+i$. Customers indicate a preferred earliest pick-up time a_i and drop-off time b_{n+i}. There exists a fleet of k vehicles of constant capacity C, which start at an origin depot $o(k)$ and must end at a destination depot $d(k)$.The following variables and constant sets must also be defined:

$P= \{1,...,n\}$	Set of pick up locations;
$D=\{n+1,....,2n\}$	Set of delivery locations;
M	Set of depots;
$N= P \cup D$	Set of pick-up and delivery locations;
$k \subset K$	Set of vehicles;
$A = N \cup \{o(k), d(k)\}$	Set of all possible stopping locations for all vehicles $k \epsilon K$;

Where:

A_i	Earliest time that a customer may be serviced at location i;
B_i	Latest time that a customer may be serviced at location i;
L_{ik}	Load of vehicle k after visiting customer location i;
q_{ik}	Number of customers loaded onto vehicle k at customer location i;
Q_k	Capacity of vehicle k;
S_i	Service time required at customer location i;
R_k	Maximum route duration for vehicle k;
U_i	Maximum ride time for customer from pick up location i;
$C_{ijk} = C_{ij}C_k$	Cost to travel between point i and j such that C_k is the cost of using vehicle k; and
X_{ijk}	Decision variable for the problem.

Thus the DARP can be defined mathematically as:

Minimise F (X_{ijk}) (2.12)

Such that:

$$\sum_{k\in K}\sum_{j\in N} X_{ijk} = 1 \qquad \forall i \in D \qquad (2.13)$$

$$\sum_{j\cup D\cup A} X_{ijk} - \sum_{j\cup D\cup A} X_{j,n+i,k} = 0 \qquad \forall k \in K, \forall i \in D \qquad (2.14)$$

$$\sum_{i\in N} X_{ijk} - \sum_{i\in N} X_{jik} = 0 \qquad \forall j \in A \cup D, \forall k \in K \qquad (2.15)$$

$$X_{ijk}\left(T_{sik} + T_{ijk} - T_{sjk}\right) \leq 0 \qquad \forall k \in K, (i,j) \in N \qquad (2.16)$$

$$a_i \leq T_{sik} \leq b_i \qquad \forall i \in N, \forall k \in K \qquad (2.17)$$

$$a_{i+n} \leq T_{aik} \leq b_{i+n} \qquad \forall i \in N, \forall k \in K \qquad (2.18)$$

$$X_{ijk}\left(L_{ik} + q_{jk} - L_{jk}\right) \leq 0 \qquad \forall k \in K, (i,j) \in N \qquad (2.19)$$

$$q_{ik} \leq L_{ik} \leq Q_k \qquad \forall i \in D, \forall k \in K \qquad (2.20)$$

$$L_{mk} = 0 \qquad \forall m \in M, \forall k \in K \qquad (2.21)$$

$$X_{ijk} \in \{0; 1\} \qquad (2.22)$$

The objective function (2.12) is usually a multi-criteria objective function, minimising the total transportation costs and customer transportation time inconvenience. Constraints (2.13) and (2.14) ensure that each customer is only serviced by a single vehicle and the delivery of a customer must take place on the same route as the customer pick-up. Constraint (2.15) prevents a flow conflict, by maintaining all customers that are picked up are dropped off at a location. Equation (2.16) is a time precedence constraint, to ensure the sum of the departure and travel times from location *i* takes place before the time of an arrival event at location *j*. The time window restrictions for a vehicle pick-up and delivery are described by constraints (2.17) and (2.18). The vehicle capacity constraint is described by (2.20), while equation (2.19) ensures the number of passengers transported by a vehicle *k* is conserved. Constraint (2.21) prevents customers from being transported to the depot, with constraint (2.22) guaranteeing only binary decision variables (Zidi *et al*., 2012).

2.3 Heuristic And Metaheuristic Methods

Given that the DARP is a NP-hard combinatorial problem, few exact algorithms exist that can solve it, with exact algorithms only able to solve small problem sizes. Psaraftis (1980) present an exact dynamic programming algorithm that can service a maximum of 9 customers. Heuristic and

metaheuristic methods have been widely applied to solve larger DARPs, which have enabled solutions to be determined for increasingly large and complex problems (see Fu, 2002).

Heuristics differ from metaheuristics as they perform a focused search and are able to produce good solutions in reasonable computing time. Conversely, metaheuristics involve intricate neighbourhood search procedures, rule based operating systems and some form of memory function. Metaheuristics tend to yield better solutions to heuristic methods but require longer computing times.

2.3.1 Classical heuristics

Classical heuristics applied to the CVRP, and its variants can be classified into the categories of route construction methods, two-phased methods, and route improvement methods. These methods include a population mechanism, which uses elements of the current solution for the next solution. A new approach of classical heuristic methods is the inclusion of learning mechanisms to improve on current solutions or to restart solving the problem with a different set of rules or parameters (Cordeau *et al*., 2004).

Route construction methods iteratively insert customers into vehicle tours, until all customers are assigned. These methods can operate sequentially or in parallel, whereby routes are expanded on one by one, or routes are built up for different vehicles concurrently. Route construction heuristics require an initialisation and selection criteria to determine which customer should be inserted into a specific route. The selection criteria determines the sequence in which a vehicle tour will service a specific customer.

The Clark and Wright Savings Algorithm is one of the most widely known routing heuristics. It aims to make route cost savings by combining customers into the same route, rather than servicing them separately to reduce the total vehicle distance travelled. The cost savings are defined by $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ (2.23), where i is a customer on one route, and j another customer on a second route. Routes are merged together if s_{ij} is positive, with the heuristic able to operate in both parallel or sequentially. The algorithm is time consuming to implement as distances, costs and savings need to be calculated between all customers. The Mole and Jameson Sequential Insertion heuristic is similar to Clark and Wright Savings Algorithm, whereby routes are expanded by evaluating the extra distance an un-routed customer would add to a current route (Cordeau *et al*., 2007).

Typically two phased heuristic approaches are used and can be classified into two groups, namely cluster-first route-second methods or route-first cluster-second methods. In the first instance, customers are grouped into feasible clusters and then vehicle routes are constructed for each cluster. The route-first cluster-second method, builds a vehicle tour as a large TSP, which may be

broken into segments to form feasible routes. However, Toth and Vigo (2002) found that the route-first cluster-second method is unable to yield a better solution than any heuristic method used for the cluster-first route-second approach.

Well known clustering methods are the Sweep algorithm, Fisher and Jaikumar algorithm and the Truncated Branch-and-Bound method. The Sweep algorithm applies to planar VRPs, and starts by selecting a random customer for the start of a vehicle's route. The algorithm then sequentially allocates the remaining customers to the current vehicle's route according to the increasing polar angle of the other customers' locations to the initial customer and the vehicle depot. If a customer cannot be allocated to the current vehicle's route, a new route is created. Each vehicle route can then be solved for individually as a TSP.

The Fisher and Jaikumar algorithm is very similar to the Sweep Algorithm, in that it only differs by forming the customer clusters by solving a generalised Assignment problem.

The Truncated Branch-and-Bound method uses a decision tree structure, where the number of levels on the tree is equivalent to the size of the vehicle fleet. Each level of the tree is restricted to represent a single route, with tree nodes representing some completed routes. Descendent tree nodes represent possible routes and unallocated customers to a vehicle's tour (Cordeau *et al*., 2007).

Improvement heuristic methods aim to find better solutions while maintaining solution feasibility. These methods typically stop when no further route exchanges can be made, without deteriorating from the current solution. Depending on the improvement method used, vehicle routes can be improved upon separately or concurrently (Cordeau *et al*., 2007).

Lin's (1965) λ-opt method is widely known for single route improvements. It operates by removing λ-edges from a tour and then reconnecting all the remaining vehicle tour segments, whereby the route improvements are calculated for each combination. If an improvement is found, it is implemented. The method stops in the event of no improvement or a local minimum being found. Checking of route optimality can be achieved in O (n^{λ}) time, with method variations existing, whereby 2 or 3 route vertices are removed simultaneously, to create 2-opt and 3-opt route interchanges.

Van Breedam (1994), proposes various route recombination methods to improve routes concurrently, with the String Exchange method found to be particularly effective. String Exchange involves switching two customer stops of different routes to generate a new neighbourhood solution. The method is displayed graphically in Figure 2.2.

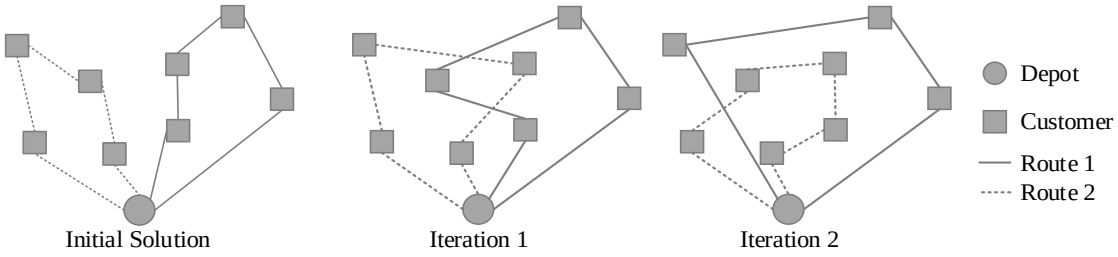


Figure 2.2: An example of a route String Exchange (Van Breedam, 2001)

2.3.2 Metaheuristics

A difference of metaheuristics to heuristic methods is that they allow solution deterioration, while performing a much larger general search of the solution space. In some cases infeasible solutions may be temporarily accepted, with metaheuristics having the limitation of not having a stopping criteria. Despite having longer computing times than heuristic methods, metaheuristics are often able to outperform heuristic methods. Metaheuristics are capable of providing a thorough search of the solution space, seeking a solution near the global optimum (see Van Breedam, 2001).

Metaheuristics can be classified into three broad classes, namely:

1. Local search methods, which move iteratively through the solution space from solution x_i at iteration i to the next solution at x_{i+1}, until some stopping criteria is met.
2. Population search methods, which generate new solutions by recombining the results from a pool of previous good parent solutions.
3. Learning search mechanisms that gradually construct a good solution overtime, using feedback mechanisms which gathers learnt information from previous iterations.

Table 2.1 provides a brief description of the common metaheuristic techniques and their algorithm classes to which they belong.

Table 2.1: Common metaheuristic techniques

Metaheuristic	Algorithm Class	Description
Simulated Annealing (SA)	Local Search	Given a current solution x, SA considers moving to a neighbouring solution x_{t+1} based on a probabilistic decision of staying at x or moving to x_{t+1}. This process is continued until a good solution is found or defined stopping criteria met.
Tabu Search (TS)		x iterates to x_{t+1} until a defined stopping criteria has been reached. When a solution becomes stuck in a local optimum, it is added to a list of forbidden moves, to allow x to move to another search neighbourhood.
Genetic Algorithms (GA)	Population Search	Each iteration consists of a pool of parent solutions where each solution is composed of a set of chromosomes. Offspring's chromosomes are generated from 2 parents, where a random mutation based on probability is applied to parent chromosomes when generating off-spring.
Ant Systems (AS)	Learning Mechanism	AS is similar to an ant colony searching for food. Artificial ants construct n tours using a probabilistic nearest neighbour heuristic at each iteration. The more times the same path is used, the 'stronger' the tour becomes, while other less used tours 'evaporate' over time.
Neural Networks (NN)		NN consists of a pool of processing units which are able to communicate with one another based on numerous interconnected links between the processing units. The links are weighted and have a transformation function between the processing units. The processing units are self-learning, like neurons in the human brain, with the NN stopping when an output processing unit is activated.

2.4 Seminal Papers

The type of algorithm that can be used to solve a DARP depends on both the number of customers, and whether the problem classification is either static or dynamic. Typical techniques used to solve DARP include custom heuristics and metaheuristics methods based on dynamic programming, tabu search (TS), simulated annealing (SA) and genetic algorithm (GA) methods. Henceforth, the application of these various methods in seminal papers will be explored for the static DARP.

Cordeau and Laporte's (2003) TS algorithm constructs an initial solution by randomly assigning customer requests' to vehicles, with an infeasible solution permissible. Their algorithm follows a typically TS method, iterating from solution t to s_t in neighbourhood $N(s_t)$. At each iteration the best non-tabu move is selected, after sequentially calculating route durations, customer waiting

times and ride times before evaluating a neighbourhood solution. Each route move is measured against the objective function, which aims to minimise route duration and ride times. When a customer request *i* is removed from a route and inserted into another route, it is added to a tabu list for θ iterations.

To further optimise routes Cordeau and Laporte's (2003) algorithm performs an intra-route interchange. Every vertex $v_1 \ldots v_{2n}$ is sequentially removed from its current position and reinserted into the best route position for the specific vehicle to minimise the objective function. Cordeau and Laporte (2003) applied their tabu search algorithm to data sets varying in size, the largest being 144 customer requests with 13 vehicles. Their algorithm first used 10^4 iterations, with only marginal improvements in the solutions being found when 10^5 iterations were performed.

Fabri and Recht (2007) propose the use of a dynamic programming algorithm that incorporates two dimensional status vector graphs and the Dijkstra algorithm. Status vectors are used to describe the demand for each customer *n*. The vectors also represent whether the customer is waiting to be serviced, has been picked up or delivered to their destination. A single vehicle's tour is determined by finding the shortest path from the source to the sink of a vertex, subject to vehicle capacity and customer time window constraints. Each vector is used to describe a single status change of a pick-up, drop-off or waiting event. Fabri and Recht (2007) were able to solve a DARP with 100 different customers, which produced thousands of nodes that needed to be explored to find a solution.

Zidi *et al.* (2012) proposed an approach that uses a multi-objective simulated annealing algorithm that minimises both cost and a service quality condition. Service quality (SQ) was defined as a function of both customer ride time (RT) and the number of customer stations visited (NSV), where SQ = RT+NSV (2.24).

The algorithm used the two phased approach of cluster-first, route-second. The initial route constructing was done by a random assignment of vehicles to the transportation requests. This method may have resulted in a violation of vehicle capacity constraints and customer time windows. However, simulated annealing allows for solution degradation as it seeks for a solution in a neighbourhood search area. The algorithm found a solution using the concept of Pareto optimality, where through repeated trials a solution by convergence was found. Furthermore, Zidi *et al.* (2012) used a 7 step neighbourhood improvement structure in the second phase of the algorithm to improve their solution. The steps involved were:

1. Selecting two customer requests *i* and *j;*
2. Removing request *j* from route $r_{j;}$

3. Inserting the pick-up point of customer request *j* in the best position of route $r_{i;}$
4. Insert the delivery point of customer request *j* directly after the best position of route r_i; the new pick up point of customer *j*;
5. Removing request *i* from route $r_{i;}$
6. Inserting the pick-up point of customer request *i* in the best position of route $r_{j;}$
7. Insert the delivery point of customer request *i* directly after the best position of route r_j, the new pick-up point of customer *i*.

Zidi *et al.* (2012) further supplement their approach with a programming algorithm to determine the times at which vehicles should arrive and depart from customer locations. Their algorithm is a variation of the Cordeau and Laporte (2003) programming heuristic. The algorithm sequentially moves through the sequence of minimising route durations, while avoiding violation of time window constraints and attempting to delay the required serving of customers as much as possible.

Zidi *et al.* (2012) developed an approach which was able to outperform a genetic algorithm approach used by Cubillos *et al.* (2009), where time windows were considered as hard constraints. The proposed genetic algorithm approach was however unable to improve on a solution obtained by Cordeau and Laporte (2003), who used a tabu search heuristic on the same data set. When Zidi *et al.* (2012) considered soft time windows, their vehicle ride times were largely similar to those obtained by Cubillos *et al.* (2009), and significantly outperformed Cordeau and Laporte's (2003) customer ride times.

Both Jorgensen *et al.* (2007) and Cubillos *et al.* (2009) apply their own version of a genetic algorithm to the same dataset of the DARP, using a cluster-first route-second approach. In both cases a genetic algorithm is used for the clustering, with Cubillos *et al.* (2009) using a pre-processing step before commencing with the clustering.

Cubillos *et al.* (2009) construct a preliminary precedence table to reduce the search space of feasible solutions. Their table considers each time event associated with a customer pick-up or delivery, and whether other events have to occur before or after the specific event. In Figure 2.3 an example of a preliminary precedence table for 3 customers (*I, II* and *III*) is shown, where (+) indicates a pick-up and (-) a delivery event. Understandably event *I*+ must occur before event *I*-, and generally can be stated as $A_x + DT_{xy} > B_y$ (2.25), where *DT* is the direct ride time from location *X* to location *Y*. Given the required precedence, an incompatible list of customers that can be serviced by the same vehicle can be determined by the list of events that precede each other simultaneously. E.g. if X is preliminary to *Y* and *Y* preliminary to *X* in that they occur at the same time, it is impossible for both *X* and *Y* to be serviced by the same vehicle. However, this condition

only holds if customer locations are either too far or too close to another, as waiting time for the commencement of a pick-up event can be incurred by a customer.

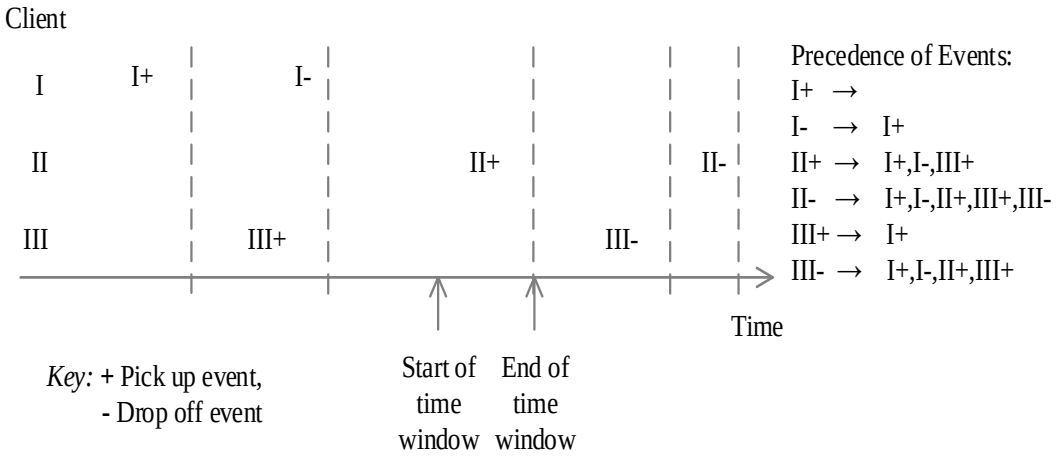


Figure 2.3: An example of the preliminary precedence of events[2]

Pre-processing of time events allows the genetic algorithm to search a smaller solution space, with less chance of moving into infeasible solution spaces (Cubillos *et al.*, 2009). The construction of a precedence table is similar to the tabu search metaheuristic where a 'memory' property of forbidden moves exists.

A comparison between the genetic algorithm approaches used by Jorgensen. (2007) *et al.* and Cubillos *et al.* (2009) is provided in Table 2.2. Both authors use a string of chromosomes to represent a vehicle route, with the order of the chromosomes representing the sequence in which customers are to be serviced. Cubillos *et al*. (2009) found that their method was able to outperform that of Jorgensen *et al.* (2007). Cordeau and Laporte's (2003) tabu search algorithm was able to yield better vehicle route durations but was unable to provide better customer ride time durations than the methods used by to Cubillos *et al*. (2009) and Jorgensen *et al.* (2007).

[2] Modified from Cubillos *et al*. (2009).

Table 2.2: A comparison of genetic algorithm heuristic approaches applied to the DARP

Author	**Jorgensen *et al.* (2007)**	**Cubillos *et al.* (2009)**
Objective function	Weighted function of transport cost & cost of inadequate service	Weighted function of vehicle travel time, waiting time and excess travel time
Time windows	Soft constraint	Hard constraint
Clustering	Genetic algorithm	Genetic algorithm
Routing	Space-time nearest neighbour heuristic	Greedy insertion heuristic

Other commonly applied algorithms used to solve the static DARP are summarised in Table 2.3.

Table 2.3: A summary of common algorithms used to solve the static DARP

Reference	**Objectives Function**	**Key Constraints**	**Time Windows**	**Algorithm Used**	**No. of Customers**
Xiang *et al.* (2006)	Minimise sum of costs of all trips	Time windows Vehicle capacity Trip duration Driver qualifications	Hard time windows	Custom heuristic with simple local search strategy for clustering & routing	$n \leq 2000$
Jaws *et al.* (1986)	Identify feasible vehicles schedule that satisfies customer pick-up & delivery times	Time windows Customer ride time	Hard time windows	Custom sequential route insertion heuristic	$n \leq 2617$
Cordeau (2003)	Minimise total routing costs	Time window tightening Vehicle capacity Precedence of events	Hard time windows	Mixed-integer branch-and-bound	$n \leq 32$

2.5 Agent Based Simulation

Agent based simulation has been gaining increasing importance as a method to use in simulating complex systems, with limited coherence existing in literature on how an ABS should be applied. According to Segfried *et al.* (2009) an ABS is generally characterised by:

1. Agents operating in an environment that is beyond their control;
2. A simulated environment forms part of the model by providing the required infrastructure. The simulated environment may perform auxiliary tasks or assist with the interaction between agents.
3. A step-wise execution approach where all agent actions' occur sequentially, with no event occurring in between the discrete event steps.

The major elements that constitute an ABS, shown in Figure 2.4, are considered to be: the model world, events and entities. The model world is common to all agents and defines the simulation specific characterises related to time, environment and operational rules that govern agent behaviours or interactions.

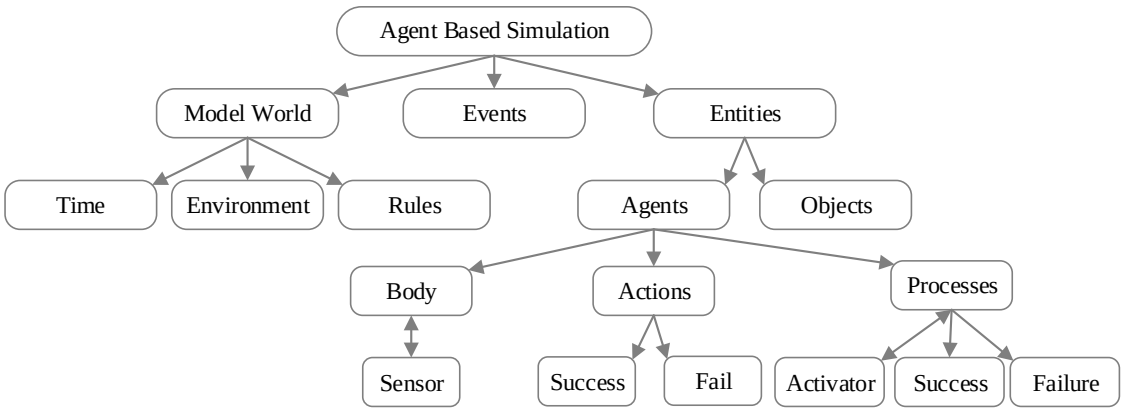


Figure 2.4: The basic structure of an ABS[3]

During the running of time in simulations, various time subsets, specific to agents' may exist, provided that simulation time remains totally ordered. The model world environment denotes the common space that agents will function in. The model world may be a bounded or unbound space, which typically defines agent or event locations.

[3] Modified from Segfried *et al.* (2009).

Events in an ABS are the set of all possible actions that may occur (e.g. customer pick-up or drop-off). Depending on the definition of the ABS, events may be either dependent or independent of time and may occur concurrently to one another, or may trigger another action.

Entities in ABS's can play either an active role (being considered an agent) or a passive role (being considered an object in an ABS). An agent is differentiated from an object in that:

1. It has a physical dimension which forms its body, and is able to sense and interpret its environment;
2. It is capable of performing multiple actions that consume a specific amount of time to be performed. Actions can be triggered by a sensor, with the outcome of an action deemed to be either a success or failure.
3. An agent is closely interlinked with action processes. One action may activate other agent actions that form a process, with the possibility of actions occurring concurrently. A process would continue to run until it reaches a successful outcome or it is terminated due to a failure (Segfried *et al.* 2009).

The use of ABS's in literature is limited, with applications in the field of transportation focusing on the general CVRP. Zeddini *et al*. (2008) solve a dynamic VRP using a multi-agent system consisting of vehicle, bidder and client agents. The authors used an objective function to minimise the number of vehicles used and the distance travelled. Client requests are managed by the bidder agent which co-ordinates with the vehicle agents. Zeddini *et al*. (2008) developed an approach involving each customer request being broadcasted to all vehicles that bid to determine the cost of adding the client to their respective routes. The bidder agent then collects all bids and assigns the client to the lowest bid vehicle. The negotiation process of this ABS involves deciding which vehicle to allocate to each customer. The ABS was unable to provide new optimal solutions to the VRP, with solutions resulting in an average of 16% more distance being travelled by the vehicles (Zeddini *et al*., 2008).

Similarly Vokrinek *et al*. (2010) use a multi-agent approach for a VRP that must minimise route costs and the number of vehicles used. Making use of 3 types of agents which have defined purposes and sub-algorithms, the VRP is able to be solved in polynomial time. The solutions from each sub-algorithm are aggregated to determine the minimum cost solution. The agents and sub-algorithms used were:

1. A single task agent, which sequences incoming customer requests using a First Fit Decreasing heuristic (FFD). FFD is typically a simple greedy approximation algorithm

used in packaging, but is applied by Vokrinek *et al*. (2010) to allocate customers to the first available vehicle that can accommodate them.

2. A single allocation agent, which reviews the customers allocated to a specific vehicle and conducts an improvement phase of the customer allocations. The improvement strategy involves reallocating the sequence in which customers for a specific vehicle are serviced, and can move a customer to another vehicle agent.
3. A set of vehicle agents, where each agent represents its own TSP using a cheapest route insertion heuristic.

This approach is not able to outperform the benchmark results, but does yield relatively good solutions in short computing times (Vokrinek *et al*., 2010). The limitation of their approach is the route improvement strategy, where each vehicle agent successively re-allocates all its assigned customers, or may assign a customer to another vehicle agent for each improvement phase. While this approach leads to extensive searching of the solution space, the lack of a memory property to maintain good solutions may result in a less optimal final solution.

Barbucha and Jedrzejowicz (2007) propose a two agent parallel model to solve a VRP that must service up to 199 customers. Both the solution manager agent and optimisation agents are permitted to improve and exchange solutions, which are then stored in a common memory bank. Barbucha and Jedrzejowicz (2007) make use of the sweep algorithm, following the conversion of customer locations from Cartesian to Polar co-ordinates. This approach allows for easy customer cluster allocation, where the customer servicing sequence is determined using the cheapest insertion method.

Barbucha and Jedrzejowicz (2007) allow the optimisation agents to use four local route improvement methods, summarised in Table 2.4. Various strategies are tested by the authors to exchange and store improved solutions, with this approach unable to yield better results obtained for a tabu search method applied to the same data set.

Table 2.4: Local route improvement methods for a two-agent VRP

Route Improvement Method	Description of Method
2-Opt Local Search	2 route edges are removed, with the entire vehicle route reconnected until a new feasible tour is obtained.
String Cross	2 route edges are exchanged by crossing edges of 2 different routes.
λ-opt Interchange	According to the value chosen for λ, λ parts are exchanged /moved in a route.
2-Lamda Algorithm	Customers located far from the centroid of the current route are removed. The removed customer is inserted into the position closest to the centroid of another route.

2.5.1 Limitations of ABS

ABS is not well known outside the narrow ABS community, with limited application of the method beyond the purposes of scientific, educational and experimental use. ABS is still considered to be in the early stages of development, with no widely accepted and proven methodology existing for the development of an ABS. The complexity required to construct an ABS is considered vast, given that it is not based on traditional mathematical computations which can be performed relatively quickly using computers. In particular the computational requirements of an ABS can be extremely intensive as the number of agents increase. This is due to the design of an ABS which usually imposes communication requirement between agents which is not scaled linearly, drastically increasing the computations that are required to solve a problem (Salamon, 2011). This is also evident from the literature reviewed, where the number of customer requests modelled did not exceed 200.

2.6 Conclusion of Literature Review

It is clear that there exists a substantial amount of literature on solving VRP's, which have focused on the use of heuristic and metaheuristic methods. Metaheuristic methods have been proven to yield better solutions to larger VRPs' than heuristics methods. The use of metaheuristics methods has also dominated the solving of the DARP, with no application found in literature of agent based simulation to solve the DARP. The field of agent based simulation is relatively new, with a limited amount of literature being found applied to VRPs. Furthermore, the approaches has been inconsistently applied to solve different types of VRP's, yielding mixed results in comparison to metaheuristic methods.

CHAPTER 3: METHOD

3.1 Overview

The main contribution of this research is to investigate if an agent based simulation approach with the use of heuristic methods can be applied to solve the Dial-a-Ride Problem. The specific methodology (sub-section 3.2) and method (sub-section 3.2) used for this research are described henceforth. The methodology will explore the elements that make up this scientific enquiry, and will establish the rules for competent and acceptable methods for using simulation as a research tool. The method will describe the way in which the research was conducted, to ensure repeatability and examination of the actual research tool.

3.2 Methodology

This section will explore the general requirements to construct a simulation, followed by investigating the components needed to develop a framework to build an agent based model. This section will also investigate the requirements needed for the outputs of the simulation to be deemed valid.

3.2.1 Simulation as an analysis tool

Computer simulation typically involves the modelling of a system through a representative mathematical model that has been programmed to follow a set logic within predefined parameters and system conditions. In general a computer simulation will always attempt to create a series of representative sample results for a specific model, which would otherwise be impossible to do in practice.

There are various benefits of using computer based simulation, namely: flexibility, the study of transient state behaviours, and communication. Flexibility refers to the ability of a user to change a model's inputs to experiment with multiple iterations of a specific model. Inputs can often be changed with ease, and are not costly or time consuming to change once the simulation model has been constructed. In terms of transient state behaviours, simulations allow for the study of random events in dynamic systems that would otherwise require too many simplifying assumptions to model using other methods. Additional benefits of computer simulation include the evaluation of the compression or expansion of time, determining sensitivity analysis of parameters, allowing for the study of non-existent processes and the evaluation of multiple performance indicators.

There are two types of modelling that can be used in simulation, namely static and dynamic modelling. Static models are independent of time, with the model equations being solved only once and the model conditions staying constant, whereas dynamic models are dependent on time.

Simulations can be further classified as continuous or discrete. Continuous models change from state to state, producing different output results as the input data varies with each small increment in time. Discrete models are evaluated at either the occurrence of specific events or time intervals, thus the model is not driven by the continuous passing of time but rather whether a binary event has occurred or not.

An important aspect of computer simulation is the quality of the input data provided, which can be time consuming to collect and inaccurate, bringing the model's validity into question.

Depending on the input data used in a simulation model, it can be classified as either deterministic or stochastic. Deterministic input data is fixed, with no random inputs. It is used in simulations with well-defined conditions, with it being used to extrapolate results or evaluate the differences between multiple deterministic models. Stochastic input data has at least one random input variable that would be from a common statistical distribution. Thus simulations that use stochastic data produce results that are one of many possible solutions. This form of data is particularly useful for modelling real world simulations, given that in reality the random occurrence of events can be represented by a statistical distribution (Hillier & Lieberman, 2010).

The procedure that is typically followed in building a simulation model is outlined in Figure 3.1.

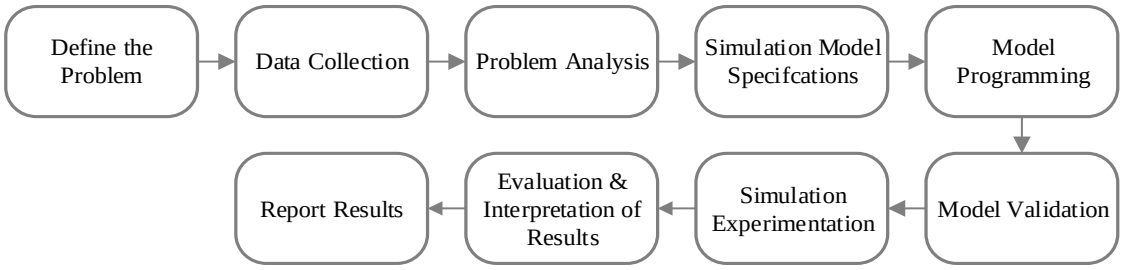


Figure 3.1: A high-level procedure to construct a computer simulation (Hillier & Lieberman, 2010)

3.2.2 ABS development framework

ABS differs from discrete or continuous simulation approaches, by operating with defined characteristics and behaviours where agents typically work according to a goal orientated approach. Salamon (2011) prescribes a methodology known as Agentology for the specification and design of agent systems. The methodology consists of a four phased approach, outlined in Figure 3.2.

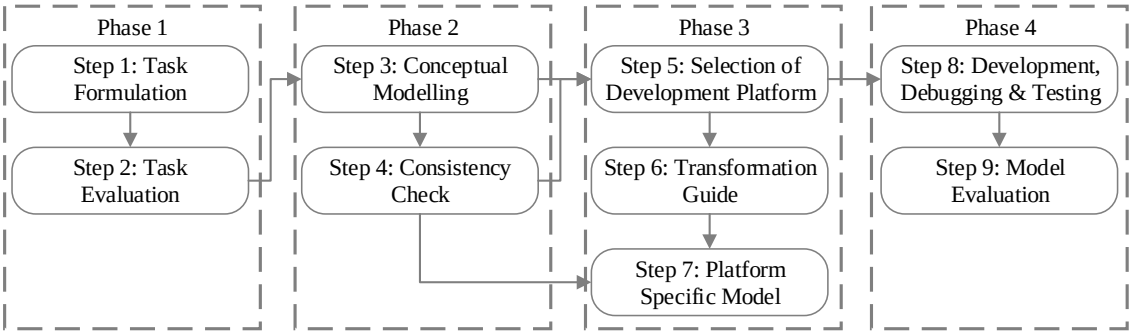


Figure 3.2: The Agentology methodology (Salamon, 2011)

Similarly to a generic simulation approach as described by Hillier & Lieberman (2010), the first phase of Agentology formulates and evaluates the required tasks that must be performed by the agent based model. The second phase involves conceptual modelling of the problem, by representing it as a diagram to depict the various aspects that the model must encompass. In the third phase an appropriate programming language is selected, with the conceptual model being transformed into the software specific programming code. In the final fourth phase, the model is developed and evaluated to determine its conformance to reality (Darragi *et al.*, 2013).

The Agentology framework will be used as a foundation in this research to develop an agent based simulation approach for the DARP[4].

3.2.3 Simulation software

ABS is possible on multiple software packages, which are either licensed products or open source. The software packages considered for use in this research were:

1. The Repast Suite, an open source agent based modelling platform which uses Java and C++ computer language. Repast has built-in libraries with common algorithms, and requires the use of a supercomputer for large models (Repast, 2015).
2. NetLogo, an agent based modelling environment that enables the study of emergent behaviours amongst agents. The software does not require the user to have a programming background and has widely been used in academia (Wilensky, 1999).
3. AnyLogic, which allows for the development of agent based models as well as discrete and dynamic event simulations. The software uses a graphical modelling language which uses an object library to build simulations using modular components. The software is aimed at corporate clients or can be used with an educational licence (Anylogic, 2015).

[4] The application of the framework is described in Section 3.3.1.

4. MATLAB, a multi-dimension computing environment that allows for customised complex numeric computations, data analysis and visualisation. MATLAB uses an array structure and can allow various other non-simulation software packages to be incorporated for use in the software (Matworks, 2014).

The proposed simulation will be built using the MATLAB R2007b software package, given that the author is already familiar with the programming language. Furthermore MATLAB provides the flexibility to build customised models and is able to compute large models.

3.2.4 Validation

Validation occurs after verification of the model has been performed, to ensure the computer simulation operates as intended. Model validation determines how accurate a model representation is of the system being studied. Methods of validation vary, with common techniques used including:

1. Obtaining real-world data, whereby data is fed into the model in historical order. The model outputs should then replicate a time series of events similar to that which were obtained for the historical data.
2. Simple tests, such as visual inspection of the output data to the expected model behaviour can be evaluated to determine simulation accuracy.
3. Statistical tests (such as regression analysis, t-statistic testing) and hypothesis testing of expected results to simulation outputs.
4. Sensitivity analysis which considers drastic what-if analysis scenarios which the simulation model is tested against (Klientjie, 1995).

3.3 Method

3.3.1 Model overview

An agent based simulation approach is developed, using the Agentology framework described by Salomon (2011), for a static Dial-a-Ride Problem with a single depot. The ABS is discrete, stepping to each time event in the model. Each agent (k), representing a vehicle, bids to service all customers (i) awaiting to be serviced at a specific time t. Agents will make bids using a weighted objective function (Z), which aims to minimise:

1. The travel distance it takes to service all customers,
2. Customer transit times, and
3. Customer waiting times.

A similar approach to Zeddini *et al*. (2008) will be used, whereby all agent bids, each representing the cost of adding a customer to the agent's route, are communicated to a central bidder agent. The central bidder agent assigns the customer for servicing to the lowest agent bid. The winning agent then determines the Actual Pick-up Time (APT) and Actual Delivery Time (ADT) for the new customer, based on the agent's current service schedule.

To build an ABS, an understanding of the different activities that will occur during the simulation must be clarified. Figure 3.3 presents an activity cycle diagram of the different tasks the agents would have to perform and interactions with customers.

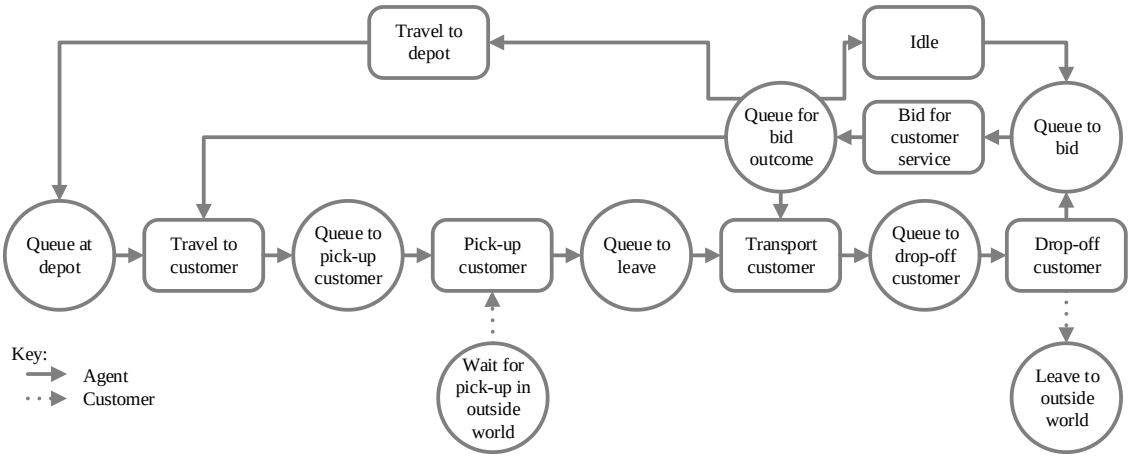


Figure 3.3: An activity cycle diagram of the DARP to be simulated

3.3.2 Data for the DARP study

The proposed ABS approach will be applied to 20 test scenarios which were developed by Cordeau and Laporte (2003). The data in the test scenarios are based on realistic assumptions, and uses input data regarding time window widths, vehicle capacity, route durations and maximum ride durations gathered from the Montreal Transit Commission.

The test scenarios are diversified by the number of vehicles available and range between 24 to 144 customer requests, with the different test scenarios summarised in Table 3.1. Each scenario has a unique depot location defined as the average location between all customer node points; refer to Cordeau *et al*. (1997) for a detailed description of this procedure. All scenarios had half the data sets with customers specifying collection time windows, and the other half of the customers specifying delivery time windows. The input data files for the test scenarios are accessible at http://neumann.hec.ca/chairedistributique/data/darp/tabu/.

The ABS model will be tested against two additional data sets (Pr21, Pr22). These additional data sets will aim to establish if the proposed approach is able to yield feasible solutions for much larger numbers of agents and customers. The location of the depots for these data sets will be set to the average of the x and y co-ordinates of all customer nodes. Data set Pr21 will be composed of data from Pr02, Pr04 and Pr05, and data set Pr22 composed of data from Pr01, Pr02, Pr03, Pr04 and Pr05.

Table 3.1: Data set test scenarios for the DARP

Data Set		Number of Vehicles	Number of Customers
Pr01	Pr11	3	24
Pr02	Pr12	5	48
Pr03	Pr13	7	72
Pr04	Pr14	9	96
Pr05	Pr15	11	120
Pr06	Pr16	13	144
Pr07	Pr17	4	36
Pr08	Pr18	6	72
Pr09	Pr19	8	108
Pr10	Pr20	10	144
Pr21		25	264
Pr22		35	360

All the data sets, except Pr21and Pr22, have solution files obtained by Cordeau and Laporte (2003) using a tabu search algorithm.

The difficulty in solving the data sets will vary according to the ratio of available vehicles to customers to be serviced. Figure 3.4 provides an overview of the ratio of vehicles to customers. The data sets are expected to increase in difficulty to solve as the number of vehicles decreases. The fewer vehicles that there are available to service a larger number of customers, will make it harder to satisfy the DARP constraints.

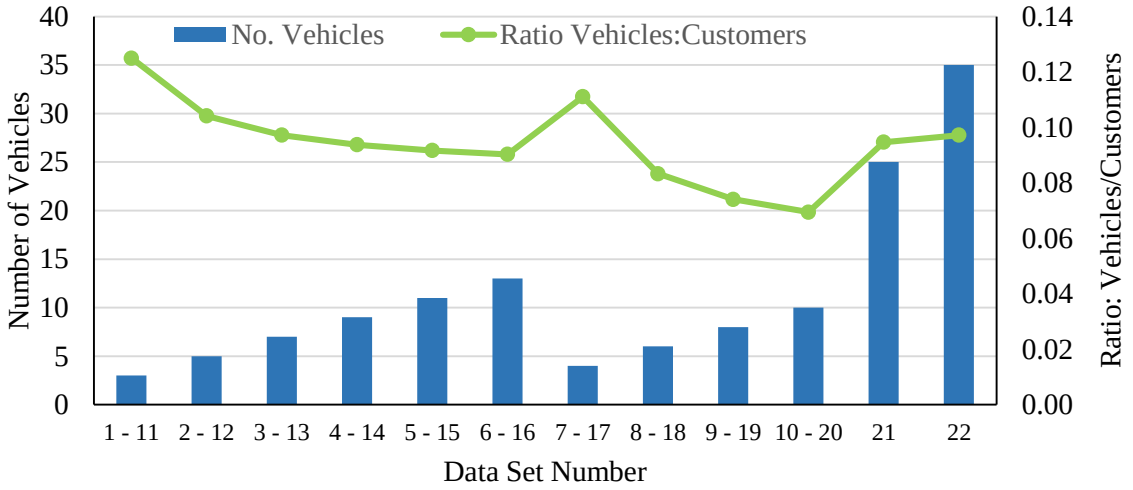


Figure 3.4: An overview of the benchmark DARP data sets

3.3.3 Model assumptions

The following assumptions were made when implementing the proposed ABS model. The assumptions marked with an asterisk (*) are the same assumptions made by Cubillos *et al.* (2009), Cordeau and Laporte (2003) and Jorgensen *et al*., which will be used as comparative studies.

1. Only the static DARP is considered*.
2. Drivers can work for a maximum of 480 minutes*, where upon they must return to the depot for a break and can depart immediately from the depot to service the next customer[5].
3. The service time for a customer at their pick-up and delivery locations is 10 minutes*.
4. Each customer request is only for a single person*.
5. Customers do not require a return journey to their origin*.
6. Euclidean distance will be used to determine the travel time and distance between any two co-ordinate locations*.
7. The time it takes vehicles to travel between any two points is equivalent to the Euclidean distance between the points, with no consideration given to possible traffic delays*.
8. The vehicle fleet size to be used for each test scenario does not need to be determined, and is defined by the test scenario*.
9. All vehicles used in the fleet have the same seating configuration, can carry a maximum of 6 passengers and are able to travel between any two locations*.

[5] Driver break time is equal to 0 minutes.

10. Constraints related to union rules, even distribution of customers amongst vehicles, fixed costs (such as costs of vehicles, driver salaries, and depot costs) are excluded from the model.
11. The depot has a zero vehicle loading or servicing time, with no customers allowed to go to the depot*.
12. The planning horizon to transport customers per day is equal to 1440 minutes*.
13. Customers specify either a desired pick-up time from their origin or a desired delivery time to their destination*. Thus customers are constrained by either a pick-up or delivery time window.
14. Hard time windows for customers apply, with a maximum permissible customer ride time of 90 minutes. Thus no customer with a specified pick-up (delivery) time window will be picked up (delivered) earlier (later) than his desired pick-up (delivery) time window*[6].
15. Minimising the routing cost of servicing all customer requests will not be directly considered. By minimising vehicle travelling distance, it is deemed to be equivalent to the lowest cost route.
16. The closer a vehicle (passenger) is to a passenger's origin (destination) the more attractive this routing option is to the vehicle (passenger).
17. Customers may not cancel or alter their transportation requests.
18. Customers have no preference which vehicle they are serviced by.
19. A vehicle is not allowed to wait idly to service another customer, when it already has one or more passengers in transit*.
20. Vehicle idle time between the servicing of customers does not need to be minimised.

3.3.4 Evaluation of the performance of the ABS model

The key performance metrics to be determined for each of the test scenarios are:

1. The total cost (distance travelled) to service all customer requests,
2. The total customer waiting time incurred for customers to be picked up from their origin,
3. The total customer transit time.

These performance measures are calculated using the formulas:

$$v = \sum_{k \in V} \sum_{i \in N} \sqrt{\left(x_{i_d} - x_{i_p}\right)^2 + \left(y_{i_d} - y_{i_p}\right)^2} \qquad (3.1)$$

[6] Cubillos *et al.* (2009) used hard times, while Cordeau and Laporte (2003) and obtained Jorgensen *et al.* (2007) used soft time windows.

$$w_t = w_{DDT} + w_{DPT} \quad (3.2)$$

$$w_{DDT} = \sum_{i=1}^{n}(ADT_i - EDT_i) \quad \forall\, i = 1, \ldots., n, where\ i\ is\ a\ DDT\ customer \quad (3.3)$$

$$w_{DPT} = \sum_{i=1}^{n}(APT_i - EDT_i) \quad \forall\, i = 1, \ldots., n, where\ i\ is\ a\ DPT\ customer \quad (3.4)$$

$$c_t = \sum_{i=1}^{n}(ADT_i - APT_i) \quad (3.5)$$

Where:

C	Total travel distance (cost)
x	x co-ordinate
y	y co-ordinate
k	Agent number
$P = \{n+1, \ldots, n\}$	Set of pick up locations
$D = \{n+1, \ldots, 2n\}$	Set of delivery locations
$N = P \cup D$	Set of pick-up and delivery locations
$V \subset k$	Set of vehicles used
w_t	Total customer waiting time
w_{DDT}	Total customer waiting time of Desired Delivery Time (DDT) customers
w_{DPT}	Total customer waiting time of Desired Pick-up Time (DPT) customers
c_t	Total transit time
ADT_i	Actual delivery time of customer *i*
APT_i	Actual pick-up time of customer *i*
EDT_i	Earliest delivery time of customer *i*

The results obtained from the ABS will be compared against the benchmarked results obtained by Cordeau and Laporte (2003), Jorgensen *et al*. (2007) and Cubillos *et al.* (2009). Cordeau and Laporte (2003) used a tabu search algorithm, with the other authors using variations of a genetic algorithm approach. A comparison of solutions will determine whether the proposed ABS approach is able to yield better results than the metaheuristic techniques applied to the DARP.

In the benchmark solutions, Cordeau and Laporte (2003) provide results detailing the total cost, total customer waiting time and total customer transit times. Jorgensen *et al*. (2007) and Cubillos *et al.* (2009) only provide the total customer transit times in their solutions. Solutions common to all authors, exist only for data sets Pr01, Pr02, Pr03, Pr05, Pr11, Pr12, P13, Pr15, Pr17 and Pr19.

Some solutions sets provided by Cordeau and Laporte (2003) are incomplete, only providing the total cost to service all customers (Pr04, Pr07, Pr08, Pr13, Pr14).

Despite having incomplete solution sets, the ABS model will be run on all data sets to determine if the method is able to yield feasible solutions for all the data sets. Only the data sets common to all authors will be used for analysis purposes in this research.

The performance of the solutions obtained by the ABS, Jorgensen *et al*. (2007) and Cubillos *et al.* (2009) will be determined by comparing them against the benchmark solutions obtained by Cordeau and Laporte (2003). The following formula will be used to determine if the specific performance measure was either superior or inferior to the benchmark performance measure:

$$\lambda = \left(\frac{v_c - v_{benchmark}}{v_{benchmark}}\right) \times 100 \qquad (3.6)$$

Where:

λ Relative performance parameter [%]

v Performance metric

c Performance metric of set {ABS, Jorgensen *et al*. (2007), Cubillos *et al.* (2009)}

A negative λ indicates the performance metric v_c is an improvement over the benchmark metric, whereas a positive λ indicates the performance metric is worse than the benchmark metric.

The quality of the customer service offered by the ABS and benchmark approaches will be determined by calculating the ratio of customer waiting time to travelling time, using the formula:

$$\rho = \left(\frac{t_{customer\ transit_e}}{t_{customer\ waiting_e}}\right) \qquad (3.7)$$

Where:

ρ Customer Service ratio metric

t Time metric

e Performance metric of set {ABS, Cordeau and Laporte (2003)}

Linear regression will be used to test whether the solution costs obtained from the ABS scale linearly with the number of customer requests. In particular, it will be determined if the linear regression is maintained with the inclusion of the large data sets Pr21 and Pr22. The curve will be fitted using the equation:

$$y_i = \alpha + \beta c_i \qquad (3.8)$$

Where:

y_i Number of customers

c_i Solution cost to service i customers

$$\beta = \frac{\overline{cy} - \bar{c}\,\bar{y}}{\overline{c^2} - \overline{y^2}} \quad (3.9)$$

$$\alpha = \bar{y} - \beta\bar{c} \quad (3.10)$$

The sample correlation coefficient r_{cy}^2 will be calculated to determine the applicability of fitting a linear trend to the data. The closer r_{cy}^2 is to 1 the more linear the solution cost are to the number of customer requests (Montgomery and Runger, 2011). r_{cy} will be calculated by:

$$r_{cy} = \frac{\overline{cy} - \bar{c}\,\bar{y}}{(\overline{c^2} - \overline{y^2})\sqrt{(\overline{c^2} - \overline{y^2})}} \quad (3.11)$$

Central Processing Unit (CPU) time was also used as a performance measure for the solutions obtained by Cordeau and Laporte (2003), Jorgensen *et al*. (2007) and Cubillos *et al.* (2009). This will not be used as a performance measure in this research, as the hardware and software used by the other authors compared to that used for this research are too dissimilar. However, CPU time will be measured to establish if ABS can obtain solutions in polynomial time. The CPU times obtained, will be fitted to an exponential distribution, described by:

$$y_i = \alpha\, e^{\beta t_i} \quad (3.12)$$

Where t_i is the CPU time required to obtained a solution for i customer requests. t_i would replace c_i in equations (3.9), (3.10) and (3.11)

The sample correlation coefficient r_{cy}^2 can also be used to determine the applicability of fitting an exponential trend to the data. This is provided the non-linear relationship (3.12) is converted into a linear correlation by taking the natural logarithm of both sides of the equation (3.12), yielding:

$$\log(y_i) = \log(\alpha) + \beta c_i \quad (3.13)$$

Therefore the closer r_{ty}^2 is to 1, the stronger the correlation of an exponential relationship between the two variables (Montgomery and Runger, 2011).

3.4 ABS Model

To simulate the activity cycle diagram shown in Figure 3.3, the ABS must be transformed into a model that can be programmed in Matlab, as prescribed by Salamon's (2011) Agentology framework. Figure 3.5 provides a high level flowchart of the Matlab ABS model. The major steps

and decisions the model performs are shown in the centre of the flow chart, with the model inputs, agent decision criteria and possible agent moves also provided. Detailed development of the major process steps shown in the flowchart are provided henceforth.

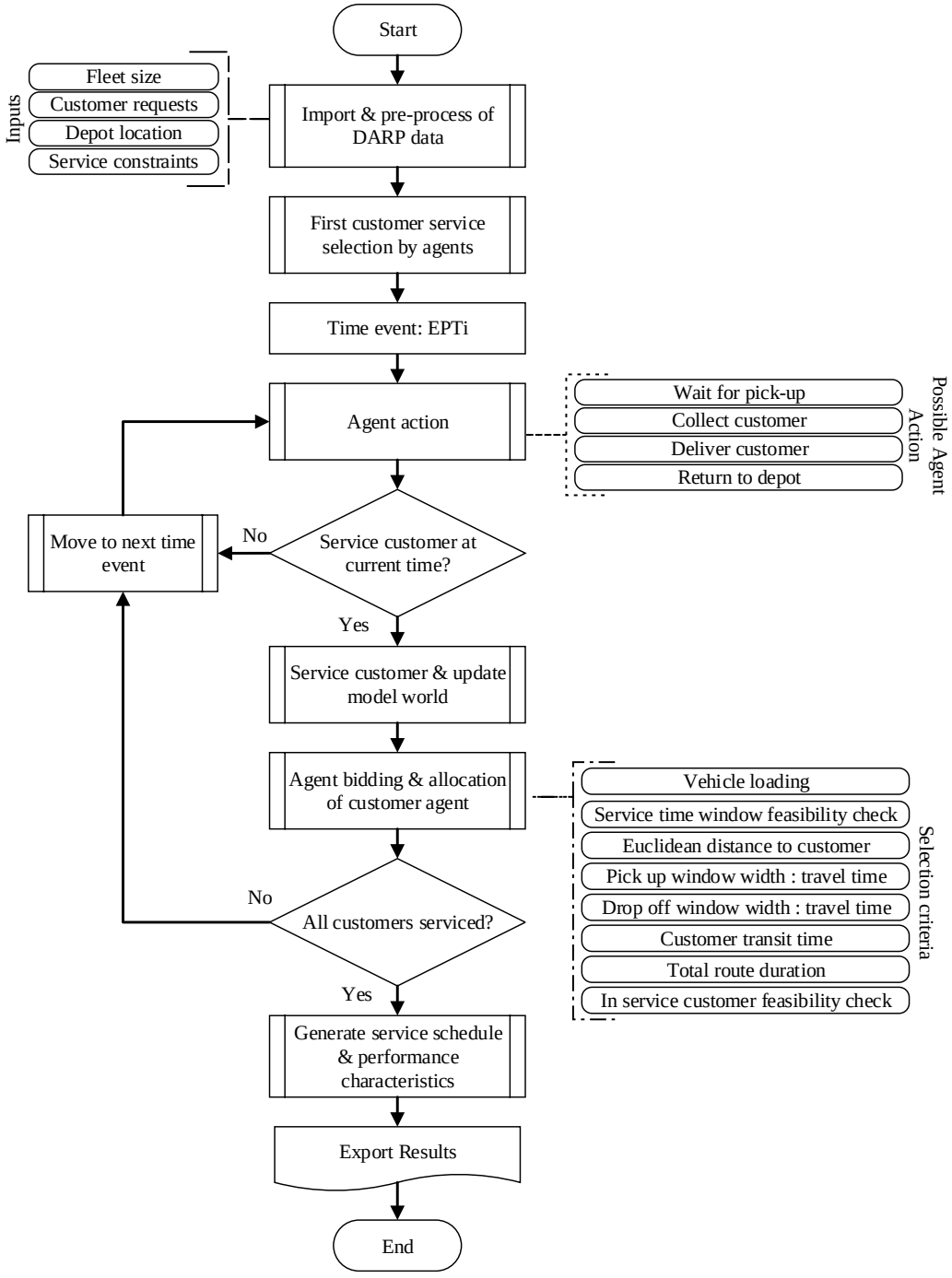


Figure 3.5: An overview of the agent based simulation model for the DARP

3.4.1 Import and pre-processing of DARP data

In the data sets there are two types of customers (i), of equal number, namely:

1. Customers which have specified a desired pick-up time (DPT_i) window, where the earliest pick-up time (EPT_i) and latest pick-up time (LPT_i) they must be serviced in is defined.
2. Customers which have specified a desired drop-off (DDT_i) time window, where the earliest delivery time (EDT_i) and latest delivery time (LDT_i) they must be serviced in is defined.

Given that only the pick-up (drop-off) time limits for DPT (DDT) customers are defined, the EDT and LDT (EPT and LDT) are set to 0 and 1440 minutes, respectively. It will be computationally cumbersome to search through time window widths of 1440 minutes to find feasible servicing times, as shown in Figure 3.6. Furthermore the search will predominately yield infeasible solutions due to the maximum ride time constraint of 90 minutes.

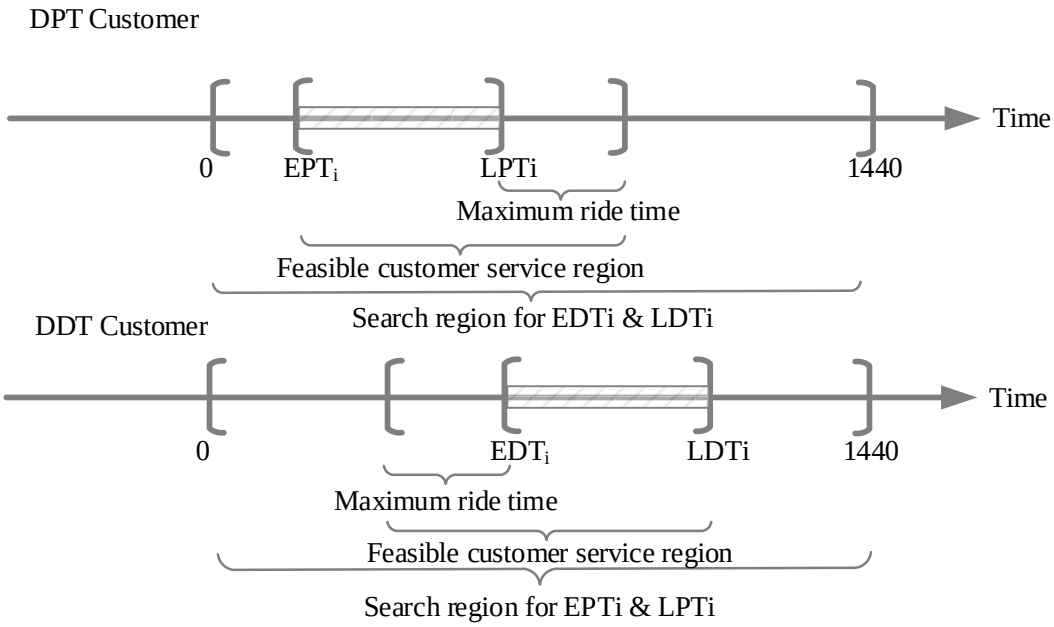


Figure 3.6: Search timelines of DPT and DDT customers in the raw input data

In the test data sets the required time precedence of events is not considered. The defining of a single pick-up or drop-off time window results in the possibility of an agent considering a drop-off event before a pick-up event. Therefore there is the need to ensure the required precedence of time events occurs, whereby a customer pick-up happens before a customer drop-off. This can be achieved by calculating the feasible delivery (pick-up) time window limits for a DPT (DDT)

customer. The window limits depend on the defined maximum ride time (MRT) constraint and vehicle loading time (L). Pre-processing of the raw data to adjust the pick-up and delivery time window limits will greatly reduce the required agent search space for feasible customer servicing times. In the ABS the pick-up and delivery time windows are pre-processed to adjust the time window limits using the formulas:

For a DPT customer:

$$EDT_i = EPT_i + L + DRT_i \quad (3.14)$$

$$LDT_i = LPT_i + L + MRT \quad (3.15)$$

For a DDT customer:

$$EPT_i = EDT_i - L - MRT \quad (3.16)$$

$$LPT_i = LDT_i - L - DRT_i \quad (3.17)$$

Where:

DDT_i	Desired delivery time of customer *i*
DPT_i	Desired pick up time of customer *i*
DRT_i	Direct ride time of customer *i*
EDT_i	Earliest delivery time of customer *i*
EPT_i	Earliest pick up time of customer *i*
LDT_i	Latest delivery time of customer *i*
L	Loading/unloading time at customer origin/destination
LPT_i	Latest delivery time of customer *i*
MRT	Maximum permissible ride time of customers

The ABS model will perform this simple heuristic at the start of the simulation, setting smaller EDT (EPT) and LDT (LPT) time window limits for DPT (DDT) customers. This will result in the time window limits and search spaces as shown in Figure 3.7. Knowing the time window bounds of customers can greatly improve the searching efficiency of feasible customer insertions into agent routes.

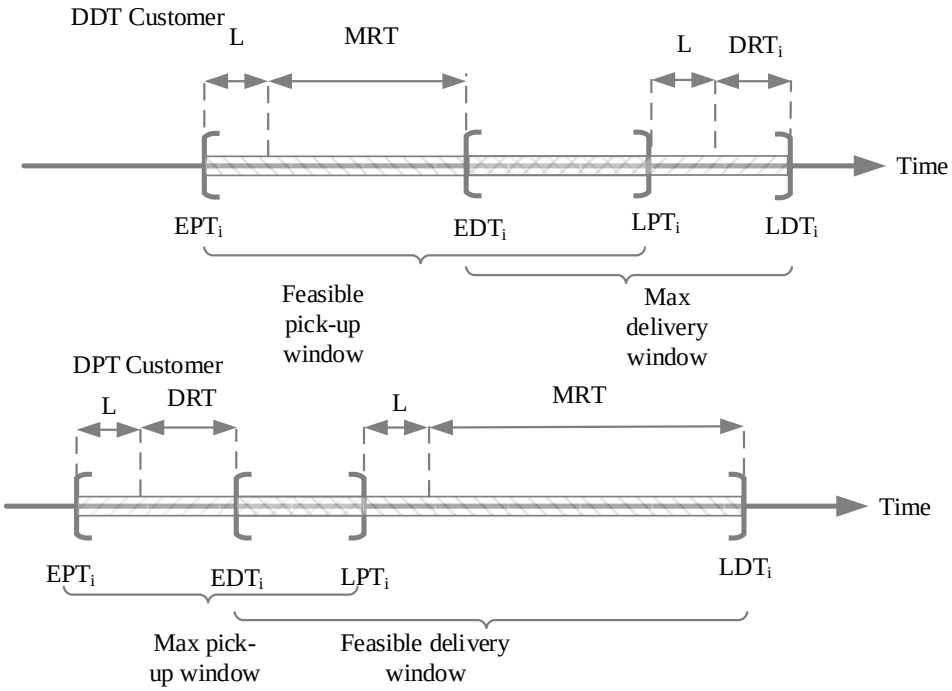


Figure 3.7: Feasible pick-up and delivery time window limits for DPT and DDT customers

3.4.2 Agent selection of customers

Central to the ABS is the 'bidding' process agents use to search for feasible insertions of customers into their schedules. The addition of a customer to an agent's schedule must not violate the service constraints of customers already in transit.

At the start of the model, *k* agents select one of *n* customers (*i*) to service, based on the i^{th} earliest possible pick-up times (EPT_i). Each agent calculates the actual pick-up time (APT_i) it should collect a customer, to minimise customer waiting and transit times for the first customer serviced. Once the APT_is' are known the agents determine the time they must leave the depot to begin their customer service schedules. The EPT_i was chosen as the time event for customer selection to prevent selection bias between DDT and DPT customers. Had EDT_i been used these events would typically occur earlier for DDT customers than DPT customers, following the time window modification. An objective function (Z) is not used for the first customer selection as the Z scores would all be the same for all agents, given that they are all starting from a common depot.

In the ABS, agents will progressively build their own schedules as the model moves to pick-up and drop-off time events. Following a drop-off event, all agents are triggered to place bids for the 'cost' of servicing all awaiting customers that have not already been allocated to an agent for servicing. The bids are communicated to a central co-ordinator agent who informs the agent with the lowest bid, which customer it has been allocated to service at a future time event. At each bidding round only one customer is allocated to one agent. The co-ordinating agent keeps track

of customers awaiting the commencement of their service, customers in transit, and the customers which have been serviced. Figure 3.8 shows the interactions between the vehicle agents, co-ordinating agent and customers. Depending on the outcome of the bidding process, the agent which has just serviced a customer will perform one of the actions shown in Figure 3.8. The model moves through discrete time events by going to the next scheduled pick-up time event (APT_i) or drop off event (EDT_i), depending on which event occurs first in the progression of time.

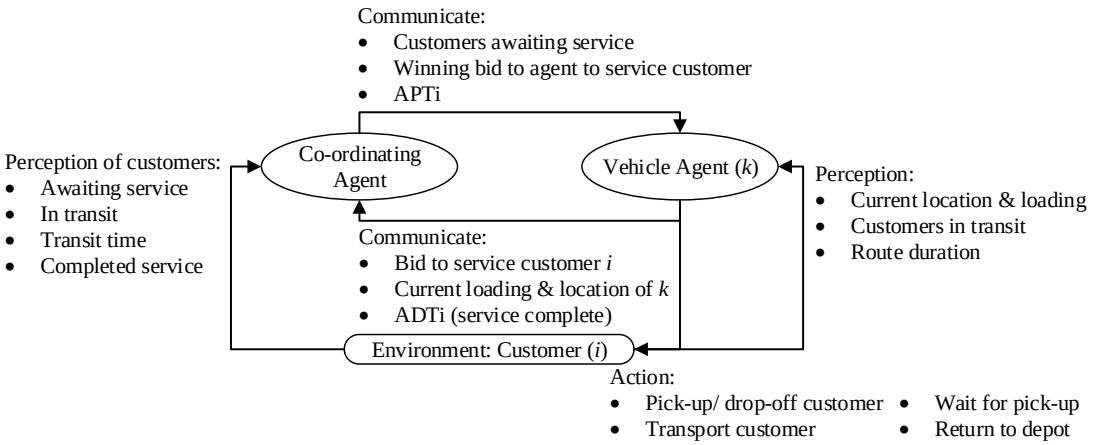


Figure 3.8: ABS agent and environmental interactions

Each agent bids to service customers based on 5 factors, which are combined into a weighted objective function expressed as:

$$Z_{ki} = w_1 f_{1ki} \times (w_2 f_{2ki} + w_3 f_{3ki}) \times w_4 f_{4ki} \times w_5 f_{5ki} \tag{3.18}$$

Where $f_{1,\dots,5ki}$ are calculated by:

$$f_{1ki} = \log(\sqrt{(x_{oi} - x_k)^2 + (y_{oi} - y_k)^2}) \tag{3.19}$$

$$f_{2ki} = \frac{EDT_i - APT_i}{\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}} = \frac{EDT_i - t - \sqrt{(x_{oi} - x_k)^2 + (y_{oi} - y_k)^2}}{\sqrt{(x_{di} - x_{oi})^2 + (y_{di} - y_{oi})^2}} \tag{3.20}$$

$$f_{3ki} = \frac{LDT_i - ADT_i}{\sqrt{(x_{di} - x_{oi})^2 + (y_{di} - y_{oi})^2}} = \frac{LDT_i - t - \sqrt{(x_{oi} - x_k)^2 + (y_{oi} - y_k)^2} - t_s}{\sqrt{(x_{di} - x_{oi})^2 + (y_{di} - y_{oi})^2}} \tag{3.21}$$

$$f_{4ki} = EDT_i - \left(t - \sqrt{(x_{di} - x_{oi})^2 + (y_{di} - y_{oi})^2}\right) - t_s \tag{3.22}$$

$$f_{5ki} = (EDT_i - t) + (LPT_i - t) \tag{3.23}$$

s.t

$$if\ LPT_i < APT_i < EPT_i\ , then\ f_{2ki} = 0 \tag{3.24}$$

$$if\ ADT_i < EDT_i\ , then\ ADT_i = EDT_i \tag{3.25}$$

$$if\ ADT_i > LDT_i\ , then\ f_{3ki} = 0 \tag{3.26}$$

$$if\ LPT_i < t < EPT_i\ , then\ f_{5ki} = 0 \tag{3.27}$$

$$if\ f_{2ki} < 0\ , then\ f_{5ki} = |f_{2ki}| \tag{3.28}$$

$$if\ l_k = 0\ , then\ APT_i = EPT_i \tag{3.29}$$

$$if\ Z_{ki}\ =\ 0\ , then\ Z_{ki}\ =\ M \tag{3.30}$$

$$if\ l_k = V, then\ Z_{ki}\ =\ M \tag{3.31}$$

$$if\ R_{ki} > R_{\mathrm{m}}\ ,\ then\ Z_{ki}\ =\ M \tag{3.32}$$

$$q_{ki} = \begin{cases} 1\ \ k\ can\ service\ \ i\ without\ violating\ service\ constraints\ of\ intransit\ customers \\ 0\ k\ violates\ service\ constraints\ of\ intransit\ customers, Z_{ki} = M \end{cases} \tag{3.33}$$

Where:

Z_{ki}	Agent *k's* bid to service customer *i* at current time in model
$w_{1,\ 2,\ \ldots,\ 5}$	Weighted constant
$f_{1,\ 2,\ \ldots,\ 5ki}$	Decision factor for agent *k* to service customer *i* at current time in model
x_k	Agent *k's* current x co-ordinate
y_k	Agent *k's* current y co-ordinate
x_{oi}	Customer *i's* pick-up point x co-ordinate
y_{oi}	Customer *i's* pick-up point y co-ordinate
x_{di}	Customer *i's* drop-off point x co-ordinate
y_{di}	Customer *i's* drop-off point y co-ordinate
t	Current time in simulation
t_{s}	Constant service time
l_k	Current customer loading of vehicle *k*
V	Maximum loading capacity
M	Very large number (big M)
R_{ki}	Expected route duration if customer *i* is added to *k's* schedule
R_{m}	Maximum permissible route duration of *k*
q_{ki}	Binary value
k & *i*	Defined according to the DARP data set used

The first factor (3.19) in the objective function (3.18) calculates the Euclidean distance between an agent's current location and the customer's location. In (3.19) the calculated distance is transformed using a lognormal distribution, to ensure customers closest to a vehicle's current location have a lower factor score than customers further away. By using a lognormal distribution,

the ‘weighting’ of this factor in the objective function (3.18) is reduced in the event of large distances between an agent and a potential customer. This enables a higher chance of a feasible agent bid, if the other factors in the objective function are small in magnitude. If a lognormal distribution was not used, customers with potentially good service windows, but located far away from an agents current location would result in a high Z_{ki} value, making it unlikely these customers would be selected for servicing.

Factors f_{2ki} (3.20) and f_{3ki} (3.21) aim to determine the customer service window ‘tightness’. Factor f_{2ki} (3.20) determines a ratio between the time window of the actual pick-up time and earliest delivery time, to the travel time incurred. The travel time is calculated assuming an agent transports a customer directly from his origin to destination. Similarly, factor f_{3ki} (3.21) determines the ratio between the actual delivery time and latest delivery time window, to the direct transit time of a customer. These factors enable the measurement of the space-time separation between stops. The time window tightness yields a lower score if a customer is serviced between the EDT_i and LPT_i. Figure 3.9, illustrates how the magnitude of the ratios change between infeasible (inf.), very large (++) and negative (--); depending on the ADT_i and APT_i calculated. The use of these factors also avoids the servicing of customers close to the limits of EPT_i and LDT_i, as this would result in long transit times and high scores for these factors.

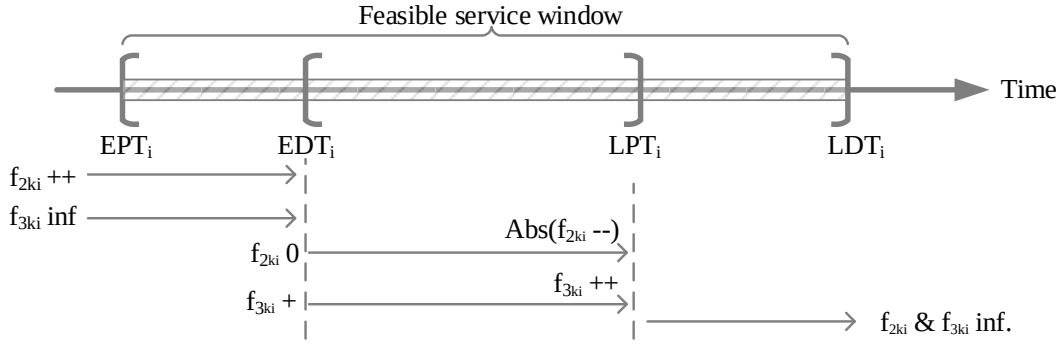


Figure 3.9: Changing magnitudes of f_{2ki} and f_{3ki} between service time window limits

In allocating a customer to a vehicle, the expected customer transit time is considered in the agents bid by factor f_{4ki} (3.22).

To prevent customers with EPTs and LPTs far in advance of the model’s current time from being favoured for selection, factor f_{5ki} (3.23) was added to the Z_{ki} (3.24). During development of the model, in the absence of this factor, it was found customers closely located to an agent’s current location were being favoured for selection. This was despite their service times being far in advance of the models current time, with certain customers not being serviced at all, yielding infeasible solutions.

Equations (3.24) to (3.28) are constraints imposed to ensure the time precedence of events is adhered too. These equations ensure a pick-up (drop-off) event does not occur after (before) a drop-off (pick-up) event. If a vehicle is empty and it is assigned to service a customer, the APT_i is set equal to EPT_i (3.29) of the customer, to minimise waiting and transit times.

If Z_{ki} (3.18) is found to be infeasible at a bid event, it is set to a very large M value ($1x\ 10^{100}$). In the event all bids are infeasible at the current time in the simulation, the model moves forward 1 unit in time. Thereafter all vehicle agents re-submit bids to service customers at the new time. The model will continue moving forward 1 unit in time, until either a feasible bid is found or until the next APT_i or EDT_i event occurs. This approach ensures a sequential progression of time during the simulation. The approach also prevents customers from being skipped for servicing, had the simulation just moved to the either the next APT_i or EDT_i event. The feasibly of Z_{ki} (3.18) is also dependant on the available vehicle capacity, described by equation (3.31).

Equation (3.32) ensures agents do not exceed the maximum vehicle route duration. R_{ki} is calculated using a sub-function which sums an agent's route duration based on: the route duration already incurred by agent *k*, the route duration to service the customers currently in transit by *k*, the expected additional route duration for the potential new customer *i*, and the route duration to return to the depot from the potential new customer's destination. This sub-function assumes that the potential new customer will be serviced last in *k's* schedule, directly between the customer's origin and destination.

In general when adding a new customer to an agent's routing schedule there are 4 possible route insertion options to be considered, illustrated in Figure 3.10, namely:

1. Both the pick-up and the delivery events are inserted at the end of last schedule block.
2. Both the pick-up and drop-off events of a customer are inserted between two other successive vehicle stops.
3. The pick-up of a customer takes place somewhere between other scheduled stops, with the customer drop-off taking place at the end of the vehicle's scheduled sequence.
4. The pick-up and drop-off events of a customer are separated by at least one other stop in the vehicle's scheduled sequence.

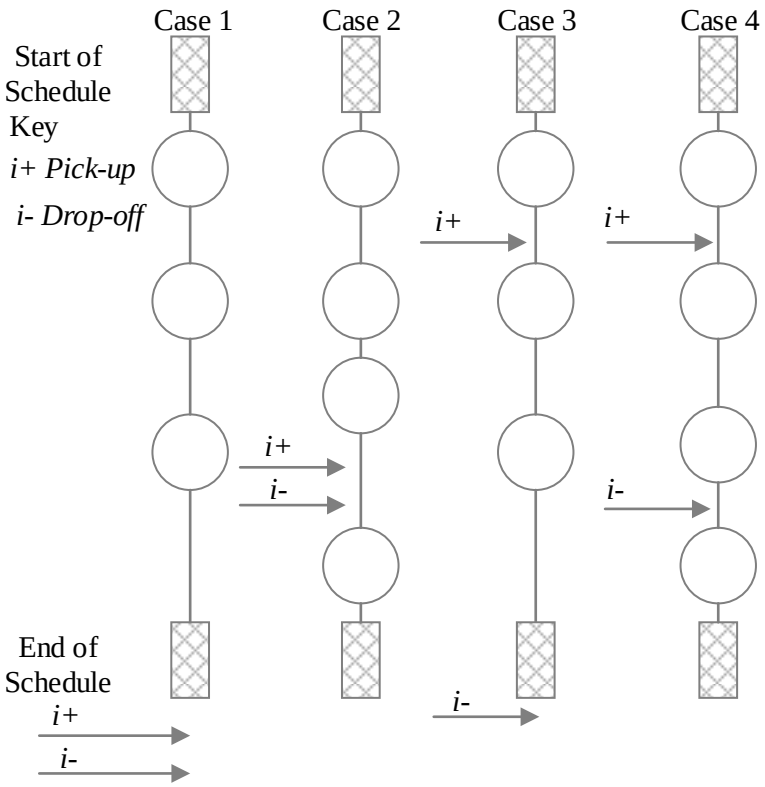


Figure 3.10: Customer route insertion options

A customer insertion into an agent route can result in the schedule becoming infeasible for cases 2 to 4 shown in Figure 3.10. The required servicing time windows of the customers already in transit are not considered in equations (3.20) to (3.32). Thus a branch and bound sub-function was added to check the servicing feasibility of the customers already in transit when adding a potential new customer to a route. Each time the branch and bound sub-function is run there are $2i!$ possible routing sequences, as each customer has a unique origin and destination. Most of the sequences would be infeasible due to the required time precedence of event (pick-up event before drop off event). It would be computationally impractical to conduct a branch and bound search of a fully loaded 6 seated vehicle as 12! ($> 479 \times 10^6$) possible routing combinations would have to be considered. Therefore, the branch and bound heuristic determines route feasibility based on the delivery time windows (EDT_i and LDT_i) only. Given that the customers are already in transit, this results in a maximum of 6! (720) possible routing combinations that must be considered by the agent. The branch and bound heuristic runs until a feasible service schedule is found, and returns a binary value (3.33), allowing the new customer to be added to the agent's schedule.

Given that the DARP is NP-hard and the solution space is highly non-convex, the 5 factors in the objective function are weighted to help discover good feasible solutions in the search space. The weighted constants were determined by using a genetic algorithm (GA) with Monte Carlo

sampling. The algorithm was able to search the feasible region of the DARP and determine weights which would produce an improved overall solution cost (total vehicle distance travelled). Details of the how the weights were obtained and running of the ABS are explained in the experimental procedure (3.4.1).

Detailed flowcharts on how the ABS model described in this section runs in MATLAB are provided in Appendix A. The MATLAB code programmed for this simulation is provided in Appendix B.

The MATLAB code used for the genetic algorithm, was provided by Campbell (2015). The genetic algorithm:

1. Used a population size of 10,
2. Selected parents using Monto Carlo simulation,
3. Used a mutation rate 0.15, and
4. Stopped after 10 iterations.

3.4.3 Experimental procedure

Two procedures are detailed, namely:

1. The procedure used to determine the weighted constants of the objective function Z_{ki}.
2. The procedure used to solve the DARP for the different data sets.

Data set Pr08 was selected to determine the weighted constants to use in the objective function (Z_{ki}). This data set has 6 vehicles to service 72 customers, and is similar to the overall average number of vehicles[7] and customers[8] across all the data sets to be tested. The ratio of vehicles to customers was relatively low in comparison to the other data sets, as seen in Figure 3.4, thus it would be harder to find a feasible solution. Therefore, if good weights could be found from data set Pr08 to yield a low solution cost, these weights would be suitable for all other data sets.

[7] Average of 7.6 vehicles across data sets Pr01-Pr20.
[8] Average of 86 customers across data sets Pr01-Pr20.

The procedure used to determine the weighted constants was:

1. A set of 5 random numbers between 0-1 were generated in MS excel using the RAND function.
2. The ABS was run, using the 5 weights in equation (3.18) with the overall solution cost being recorded.
3. Steps 1 and 2 were repeated 10 times.
4. The set of 5 random weights which yielded the lowest solution cost out of the 10 iterations was selected as the initial starting population of the GA.
5. The GA was run to obtain new 'good' weights.
6. The DARP was then re-run using the weights obtained from the GA in step 5.
7. Steps 5 and 6 were repeated three times, replacing the initial population with the output weights obtained each time from the GA.
8. From the 3 GA solutions obtained, the weights which yielded the lowest solution cost were selected for use in running the ABS for all DARP data sets.

The 10 sets of random weights generated by the RAND function can be found in Appendix C. The weights and the solutions obtained from GA are presented and discussed in Chapter 4.

The procedure used to solve the DARP for the different data sets (Pr01-20) was:

1. Copy the data test from http://neumann.hec.ca/chairedistributique/data/darp/tabu/ into column A1 of an MS excel document called "*dataset_test*".
2. Split the data into columns by selecting *Data* > *Text to Columns* > *Delimited* > *Next* > *Space* > *Finish.*
3. Save and close the MS excel document.
4. Run the ABS code in MATLAB.

All models were solved on a Lenovo PC with Intel i5 core processor, CPU 2.3 GHz, 4.00 GB RAM, Windows 8.1 Prof, MATLAB R2007B.

Precaution:

- To avoid interrupting the writing of the solution file obtained by the model, MS excel should not be opened while MATLAB is running.

For each of the ABS test scenarios a solution file was generated detailing:

1. A list of the customers' APTs, ADTs and agent which they were serviced by,
2. The solution cost (total travel distance) to service all customers,

3. The total customer waiting time,
4. The total customer transit time,
5. The CPU time,
6. The route duration and service count for each vehicle, and
7. The route used by each vehicle to service the customers.

3.4.4 Validation

The model was validated by conducting simple tests throughout the development of the ABS. At first a small data set of 12 customers and 2 agents was used repetitively to test the ABS during the development phase of the main function and sub-functions shown in Appendix A. The tests checked to ensure:

1. All constraints were adhered to,
2. From the solution files obtained, the performance metrics were calculated manually to determine if the model produced the correct values.

The small test data set used data from Pr01, consisting of the first 6 customers with desired pick-up time windows and the first 6 customers with desired delivery time windows defined. The depot location was kept unchanged, as defined in Pr01.

Further tests conducted included:

1. Reducing the number of vehicles to 1, to check the routing sequence.
2. Adjusting the maximum customer ride to 10 minutes, to ensure an infeasible solution would be obtained.
3. Reducing the vehicle capacity 1 to, to check the vehicle routing sequence and customer selection.
4. Comparing the routing sequence of Pr01 to the benchmark solution. This comparison is presented in Chapter 4.

3.5 Summary of Method

This chapter has presented a detailed explanation of the process used to develop this study's research tool. It has provided details of the elements required from the methodology to develop an ABS. Most importantly this chapter has presented how the ABS was formulated as the research tool. The formulation includes explanations of all the factors involved in the ABS, data sources and data processing procedures for both the inputs and outputs of the research tool.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Results

Table 4.1 provides a summary of the optimal objective function weights obtained from the genetic algorithm. Iteration 0 is the initial input weights used for the running of the genetic algorithm. The weights (w_1,…, $w5$) obtained from iteration 2 were used in the objective function of the ABS (equation 3.18) for all data sets.

Table 4.1: Objective function weights obtained from the genetic algorithm

Iteration	w_1	w_2	w_3	w_4	w_5	Soln. Cost (Z)
0	1.000	0.294	0.237	0.531	0.092	569.92
1	1.000	0.639	1.000	1.000	0.639	597.18
2	0.157	1.000	0.957	0.485	0.679	564.09
3	1.000	0.642	0.485	0.784	1.000	606.00

A complete sample of the solution output file obtained for Pr01 is provided in Appendix D. Table 4.2 provides the optimal ABS scheduled and service times for Pr01.

Table 4.2: Optimal ABS schedule for Pr01

Customer No. (*i*)	EPT	LPT	APT	EDT	LDT	ADT	Transit Time [min]	Agent (*k*)	Service Sequence for Agent
1	0	1440	245	258	287	260	14	2	5
2	0	1440	320	329	361	337	17	3	7
3	0	1440	203	209	252	220	17	3	5
4	0	1440	411	416	460	427	17	3	9
5	0	1440	297	305	349	313	16	2	6
6	0	1440	431	432	458	449	18	1	3
7	0	1440	193	202	236	204	11	3	3
8	0	1440	210	225	252	227	17	2	4
9	0	1440	100	102	123	121	21	3	1
10	0	1440	256	260	276	270	14	3	6
11	0	1440	175	178	215	195	20	3	2
12	0	1440	374	381	397	386	12	3	8
13	325	358	333	0	1440	352	18	2	7
14	111	152	117	0	1440	135	19	2	2
15	395	421	395	0	1440	408	13	3	10
16	84	143	87	0	1440	109	22	1	1
17	86	114	86	0	1440	105	18	2	1
18	409	426	414	0	1440	432	19	3	11
19	454	470	458	0	1440	481	23	2	8
20	175	202	175	0	1440	192	17	3	4
21	416	453	419	0	1440	437	18	3	12
22	147	177	147	0	1440	165	18	2	3
23	471	499	473	0	1440	487	14	1	4
24	321	346	325	0	1440	344	18	1	2
Average	-						17	-	
Std. dev.	-						3	-	

Table 4.3 shows the scheduled routes obtained for each agent for the ABS and benchmark solutions. The number of customers serviced by each agent is shown by *i*. Depot stops are represented by a -1. A customer pick-up event is represented by the customer number *n* and customer drop-off event by *n+24*.

Table 4.3: A comparison of Pr01's vehicle routes for the ABS and benchmark solutions

Service Sequence					
ABS			**Benchmark (Cordeau & Laporte, 2003)**		
***k* = 1, *n* = 4**	***k* = 2, *n* =8**	***k* = 3, *n* =12**	***k* = 1, *n* =2**	***k* = 2, *n* =9**	***k* = 3, *n* =13**
-1	-1	-1	-1	-1	-1
16	17	9	10	14	9
40	41	33	11	22	17
24	14	20	35	3	33
48	38	11	34	27	8
6	22	44	-1	46	20
30	46	7		38	1
23	8	35		12	41
47	32	3		24	7
-1	1	31		48	31
	25	27		6	44
	5	10		36	32
	29	34		15	2
	13	2		18	25
	37	26		30	5
	19	12		21	13
	43	36		39	29
	-1	15		42	26
		39		45	16
		4		-1	4
		18			28
		21			37
		28			19
		42			23
		45			40
		-1			47
					43
					-1

Table 4.4 provides a summary of the performance metrics obtained for the ABS for problem sets common to Cordeau and Laporte (2003) Jorgensen *et al*. (2007) and Cubillos *et al.* (2009), including the additional large data sets Pr21 and Pr22. The results obtained for all other data sets are provided in Table E.1 in Appendix E.

Table 4.4: A summary of the benchmark, ABS and genetic algorithms' results

Data Set	Cost of Soln.		Total Customer Transit Time [min]				Total Customer Waiting Time [min]	
	Benchmark	ABS	Benchmark	GA_J	GA_ C	ABS	Benchmark	ABS
Pr01	190	242	1095	310	525	414	211	146
Pr02	302	383	1976	1330	838	730	723	320
Pr03	532	685	3587	2894	1598	1257	607	493
Pr05	640	848	6156	4837	2935	1764	833	788
Pr11	164	206	1042	549	450	372	321	140
Pr12	296	359	2393	1300	745	709	309	269
Pr15	590	830	6105	4720	3153	1741	606	814
Pr17	248	339	1762	784	612	611	129	249
Pr19	602	902	5581	5358	2516	1663	487	657
Pr21	-	2045	-	-	-	3677	-	1464
Pr22	-	3000	-	-	-	5250	-	2424

Key

Benchmark	Cordeau & Laporte (2003) solution using TS
GA_J	Jorgensen *et al.* (2007) solution using GA
GA_ C	Cubillos *et al.* (2009) solution using GA
ABS	ABS solution
	ABS worse than Benchmark solution
	ABS better than Benchmark but worse than GA solutions
	ABS better than Benchmark & GA solutions

Figure 4.1 presents the relationship between the solution cost of the ABS and benchmark solutions versus the number of customer requests. The linear trend was fitted using equation (3.8) and correlation coefficient determined using equation (3.11). The linear trend was first fitted to the same data sets as the benchmark solutions[9] for the ABS solutions, and then a second time to include the large data sets Pr21 and Pr22.

[9] Pr01, P02, Pr03, Pr05, Pr11,Pr15,Pr17, Pr19

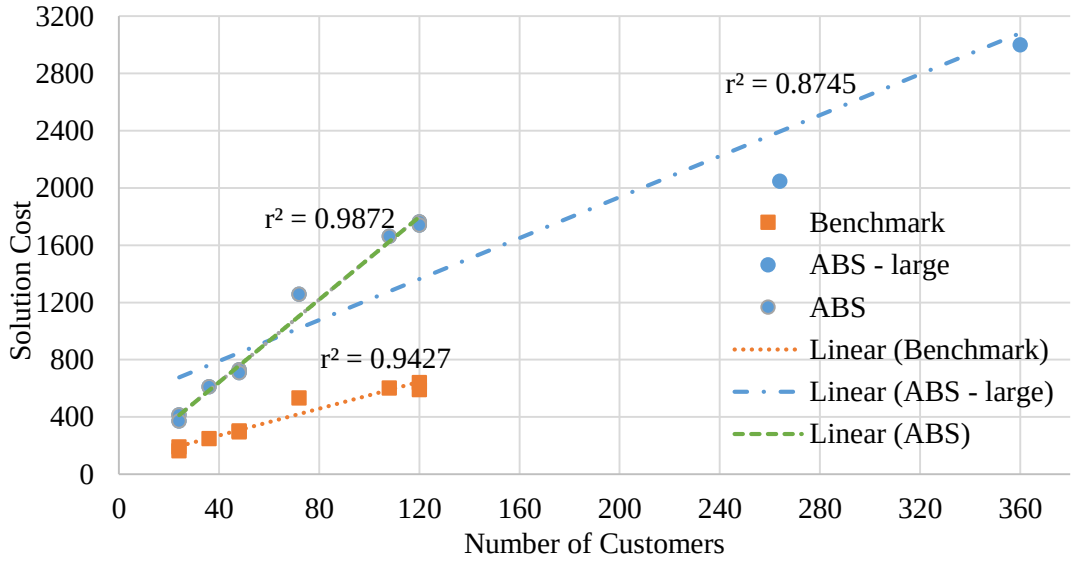


Figure 4.1: Linear regression of customer requests to solution costs

Table 4.5 shows the relative performance of the ABS approach, and solutions obtained by Jorgensen *et al*. (2007) and Cubillos *et al.* (2009), against the benchmark solutions obtained by Cordeau & Laporte (2003). The relative performance was calculated for the three performance measures using equation (3.6). A negative (positive) percentage indicates the performance measure was superior (inferior) to the benchmark performance. The additional performance measure of the ratio of the customer transit time to waiting time was determined using equation (3.7). The larger (smaller) the ρ value is, the higher (less) amount of total transit time was incurred in comparison to customer waiting time.

Table 4.5: The relative performance of the ABS, Jorgensen *et al.* (2007), and Cubillos *et al.* (2009) to the benchmark

Data Set	λ (3.6) = Cost	λ (3.6) = Customer Transit Time			λ (3.6) = Customer Waiting Time	ρ (3.7) = Transit Time : Waiting Time	
		GA_J	GA_ C	ABS	ABS	Benchmark	ABS
Pr01	27%	-72%	-52%	-62%	-31%	5.2	2.8
Pr02	27%	-33%	-58%	-63%	-56%	2.7	2.3
Pr03	29%	-19%	-55%	-65%	-19%	5.9	2.5
Pr05	32%	-21%	-52%	-71%	-5%	7.4	2.2
Pr11	25%	-47%	-57%	-64%	-96%	0.3	2.7
Pr12	21%	-46%	-69%	-70%	-13%	7.8	2.6
Pr15	41%	-23%	-48%	-71%	34%	10.1	2.1
Pr17	37%	-56%	-65%	-65%	93%	13.7	2.5
Pr19	50%	-4%	-55%	-70%	35%	11.5	2.5
Ave.	32%	-36%	-57%	-67%	-6%	7.2	2.5
Std. dev	8%	20%	6%	4%	52%	3.9	0.2

The average customer waiting times for the ABS and benchmark solutions are shown in Figure 4.2.

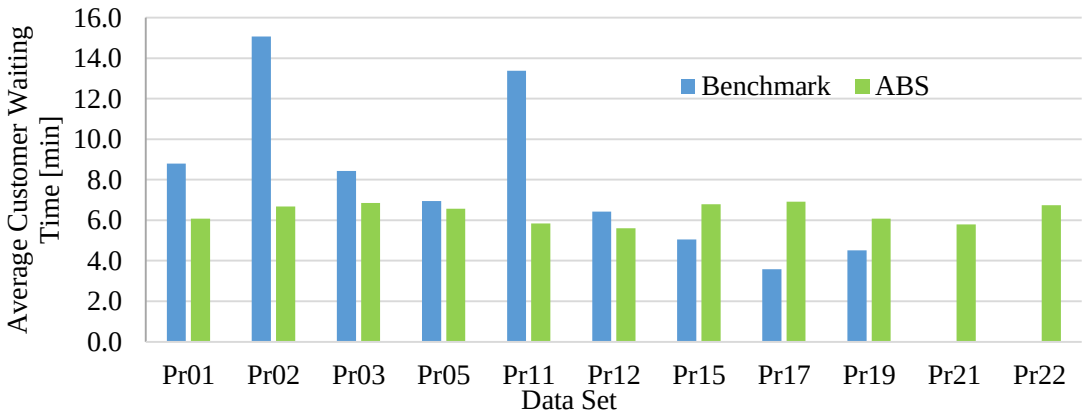


Figure 4.2: Average benchmark and ABS customer waiting times

Given that Cordeau & Laporte (2003) used soft time windows and the ABS approach used hard time windows, a comparison between the customers' transit times, for customers which exceeded the maximum permissible transit time[10] for data sets Pr01, P02, Pr03 and Pr05 was analysed. Table 4.6 provides a summary of the customer number, APT, ADT and transit time for customers

[10] 90 minutes

which exceeded the ride time constraint in the benchmark solutions. The solutions obtained for the same customers using the ABS approach are also provided.

Table 4.6: A comparison of transit times using hard versus soft time windows

Data Set	Customer (*i*)	Benchmark (soft time windows)			ABS (hard time windows)		
		APT_i	ADT_i	Transit Time [min]	APT_i	ADT_i	Transit Time [min]
Pr01	14	152	252	100	117	135	19
	17	105	205	100	86	105	18
Pr02	30	248	348	100	220	235	15
Pr03	67	341	441	100	351	363	12
	69	90	190	100	67	86	19
Pr05	1	138	238	100	211	223	12
	9	306	406	100	395	408	13
	24	134	234	100	206	218	12
	34	268	368	100	361	377	16
	56	222	322	100	317	332	15
	69	201	301	100	203	218	15
	70	171	271	100	153	173	20
	74	259	359	100	256	273	17
	84	179	279	100	149	166	17
	100	302	402	100	278	295	17

Figure 4.3 shows the ABS CPU time versus the number of customer requests for all data sets (Pr01-Pr22). The CPU times for each of the data sets can be found in Table E.2 in Appendix E.

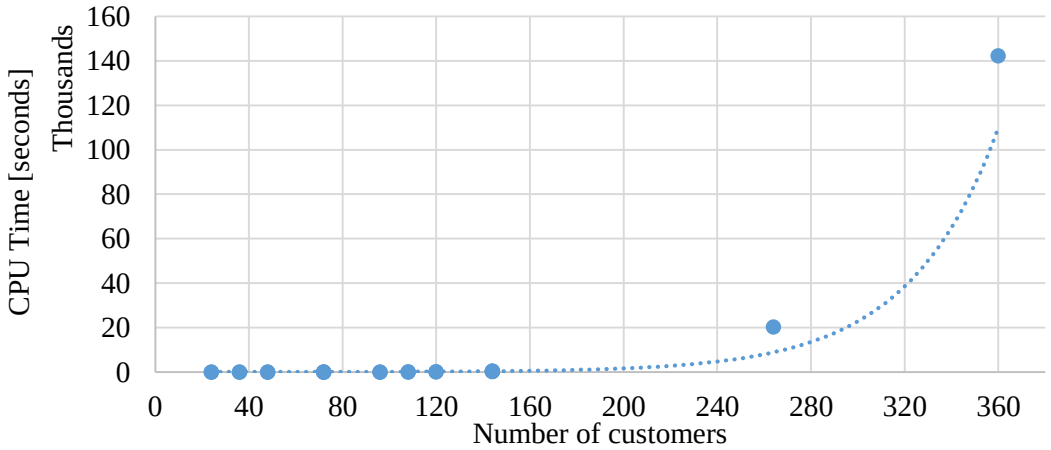


Figure 4.3: Exponential distribution of customer requests to CPU time

The exponential curve was transformed into a linear graph, shown in Figure 4.4, using equation (3.13). The correlation coefficient in Figure 4.4 was determined using equation (3.11).

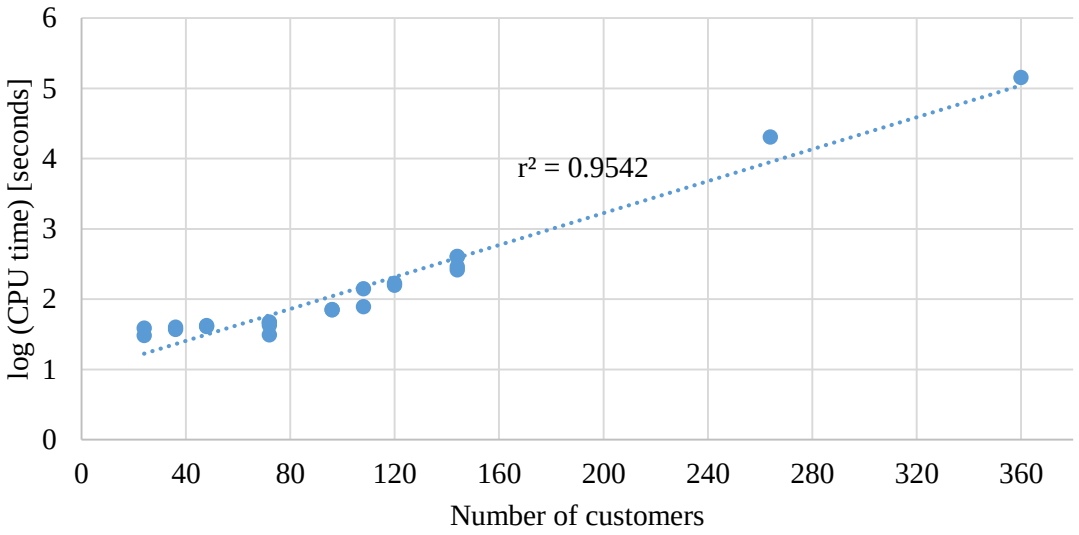


Figure 4.4: Sample correlation coefficient of customer requests to CPU time

4.2 Discussion

The interpretation of the results will:

1. Discuss the weighted constants obtained for use in the objective function of the ABS.
2. Evaluate the solutions obtained by the ABS against the benchmark solutions and other studies to answer this study's research objectives.
3. Discuss the similarities between the findings of this study and the literature reviewed.
4. Discuss the differences in the solutions obtained due to the use of hard time windows for the ABS and soft time windows used by the benchmark solutions.
5. Evaluate the speed in which the ABS solutions were obtained.
6. Determine the practical significance of this study's results and identify the limitations thereof.

4.2.1 Weighted constants

In Table 4.1 it can be seen that the first iteration of the genetic algorithm resulted in a higher solution cost in comparison to the starting population (iteration 0). The benefit of having run the genetic algorithm 3 times to obtain 'good' weighted constants is now clear, with the best weights being obtained from iteration 2. The performance of the genetic algorithm is typical of metaheuristic methods where solution deterioration can take place, with iteration 3 yielding a worse solution than iterations 0 to 2.

A limitation of this study is not having investigated the impact of different mutation rates, population sizes and the number of iterations performed by the genetic algorithm, to determine if improved solution costs could have been obtained.

The weights obtained from iteration 2, weighted the customer pick-up tightness window factor and customer drop-off time window factor with the highest weights for use in the objective function. The pick-up tightness window factor (w_2) was assigned a value of 1, with the drop-off time window factor (w_3) a value of 0.97. These high values indicate that these two factors least influence the agents' bids. The travel distance factor (w_1) had the lowest weighted value of 0.15, indicating that the lognormal Euclidean distance values obtained most influences the agents' bid values, and subsequently the overall solution costs obtained. The influence of travel distance factor (w_1) on the overall solution cost is seen in iterations 0, 1 and 3 in Table 4.1. These iterations had w_1 set to 1, resulting in the higher solution costs being obtained than in iteration 2.

Using a low weighted value (0.157) for the travel distance factor (w_1) will result in customer selection being focused on customer service time windows. The use of the low factor weighting enables customers with good service windows, but potentially located further away from the agent's current location, to be selected for service.

4.2.2 Performance of the ABS

The ABS approach was able to obtain feasible solutions for all 22 test data sets. It can be seen in Table 4.2 that the ABS was able to obtain short and consistent transit times for all customers in Pr01, averaging a transit time of 17 minutes with a standard deviation of 3 minutes. From inspecting the APTs in Table 4.2 of the DPT customers (13 to 24), the ABS scheduled pick-up events close to or at the customers' EPTs. Similarly for the DDT customers (1 to 12), their ADTs occur closer to the customers' desired EDTs than their LDTs.

The ABS routes obtained for Pr01, shown in Table 4.3, were different to the benchmark solution with the number of customers, allocation of customers and servicing sequence between the two approaches differing. The ABS was able to achieve a more even distribution of customers amongst the agents. In the benchmark solution agents 1, 2 and 3 serviced 2, 9 and 13 customers respectively, resulting in an unbalanced utilisation of the vehicles. The ABS achieved a more balanced distribution of customers amongst the three vehicles, with agents 1, 2 and 3 servicing 4, 8 and 12 customers respectively.

For Pr01 the scheduled pick-up and drop-off events for each customer are able to occur consecutively for all customers serviced by agents 1 and 2. This demonstrates the ABS is able to provide a feasible solution, without having customers needing to share routes between their

desired locations. Conversely, the benchmark solution schedules most customers on indirect routes between their transport nodes. This results in a total transit time of 681 minutes higher than the ABS (determined from Table 4.3). The benefits of the direct serving of customers is reflected in the short transit times shown in Table 4.2. In reality the ability of a shared taxi service to regularly directly transport customers between their destinations would be highly beneficial for customers but not the higher costs.

The ABS approach returned higher solution costs across all the data sets in comparison to the benchmark values, as shown in Table 4.4. The ABS solution costs showed a strong correlation (r^2 = 0.97) between the number of customer requests and total solution costs. The high correlation coefficient of the ABS, excluding the large data sets, indicates the solution cost scales linearly with the number of requests as seen in Figure 4.1. The linear ABS correlation is similar to that obtained for the benchmark solution costs (r^2 = 0.95), despite yielding higher solution costs. The linear scalability of the ABS approach does however deteriorate to r^2 = 0.84, with the inclusion of the larger data sets Pr21 and Pr22. This indicates that the increase in solution costs begins to decrease for larger problems.

The ABS approach was able to provide lower customer transit times for all the data sets than that obtained by the other three studies, as shown in Table 4.4. However, there is the exception of the total travel time for Pr01 which was 104 minutes more than Jorgensen *et al*. (2007) achieved. It is not known why the ABS yielded a worse total travel time for only Pr01, with the complete solution of Jorgensen *et al*. (2007) not provided to show how the customers were routed for servicing.

The ABS customer transit times were significantly better than the benchmark times, averaging 67% less than the benchmark values. From Table 4.5 it can be seen that the solution costs of the ABS averaged 32% more than the benchmark solutions. In reality it can be argued that the higher solution costs incurred are worth paying for, given the substantially better customer transit times that would be provided. This would be scenario specific and dependent on whether the customer is willing to pay more for the gains in service that would be achieved. In the event the DARP was run as a public transportation system, the current ABS solution costs would not be preferred due to the higher costs required to service customers.

The ABS approach yielded better total customer waiting times in comparison to the benchmark solutions, for 6 out of the 9 data sets analysed, shown in Table 4.4. The ABS was able to maintain a consistent ratio of approximately 2.5 minutes of customer transit time incurred for each minute of waiting time across all the data sets, as shown in Table 4.5 . While the ABS approach yielded worse customer waiting times for data sets Pr15, Pr17, Pr19 in comparison to the benchmark

values, the total customer transit times obtained by the benchmark solutions were significantly higher than the ABS approach. For Pr17 the benchmark solution results in a ratio of 13.7 minutes transit time for each minute of waiting time, in comparison to the ABS's ratio of 2.5. Overall the benchmark data sets average a ratio of 7.2 minutes of transit time for each minute of waiting time. This indicates that the benchmark solutions scheduled numerous non-sequential customer pick-ups and drop-offs on the routes, resulting in the higher transit times. This is further evident from the benchmark ratio value of 5.2 for Pr01, in Table 4.5, and is confirmed by the scheduled sequence of Pr01 in Table 4.3. In terms of overall customer service, a balance should be sought between the amount of waiting and travel time incurred. It is clear that the ABS has been able to find a suitable and consistent balance across all the data sets. The improved balance of customer transit time to waiting times can further justify the higher solution costs incurred.

The ABS approach was able to produce consistent average customer waiting times of approximately 6 minutes for all the data sets, including the large data sets Pr21 and Pr22. The benchmark solutions average waiting times were erratic as seen in Figure 4.2. The ability of the ABS approach to consistently yield the same average customer waiting times, would enable the agents to inform their customers of their approximate APTs given their defined EPTs.

It is evident that the ABS approach developed is able to solve the static DARP for all test instances. The approach provides significantly better customer service, through lower travelling and waiting times, than the solutions obtained using conventional operations research methods. The improved customer service is at the expense of marginally higher solution costs incurred.

4.2.3 The ABS approach compared to seminal papers

The ABS approach differs from conventional metaheuristics methods by allocating customers to vehicles based on the lowest agent bid at the current time event *t*. The approach avoids the use of the common two phased approach of cluster-first, route-second, used by authors like Fabri and Recht (2007). However the approach is similar to conventional methods, like Cordeau and Laporte (2003), by requiring agents to make a decision on which customer to service by considering the servicing sequence and scheduling of customer pick-up and drop-off events.

The approach developed used many of the elements and characterises needed in an ABS defined by Segfried *et al.* (2009)[11] and Zeddini *et al.* (2008). The similar elements used in the model included: a time step-wise execution approach, defining the model world, model events and model entities. This demonstrates that a generalised ABS structure can be used to solve different types

[11] Shown in Figure 2.4 page 21.

of VRPs. The ABS, like Zeddini *et al.* (2008), had a central bidder agent with each vehicle representing a unique agent. Similarly, the simulation used a central bidder agent to collect all bids and assign customers to the lowest bid vehicle.

Zeddini *et al.* (2008) applied their agent based approach to a dynamic VRP. Their travel distance solutions, like this study's ABS solutions, were worse than the benchmark distance solutions obtained using metaheuristic methods. Therefore, the use of a similar agent based approach on a different type of VRP appears unable to beat typical metaheuristics methods to obtain lower travel distances. The fact that the ABS approach developed in this study used no route construction or improvement heuristics such as the Clark and Wright Savings algorithm or Sweep algorithm, could be the reason for the poorer travel costs obtained.

The ABS approach used a similar concept to that of Cubillos *et al.* (2009) to reduce the search space, with the authors constructing a preliminary precedence table of time events for their tabu search method. The ABS pre-processed the raw data using equations (3.14) – (3.17) to reduce the search space of the delivery (pick-up) time windows for DPT (DDT) customers, such that it was within the feasible constrained limits of EDT (EPT) and LDT (LPT). It is important to note that the improvement in the customer transit and waiting times obtained by this study are mainly explained by preconditioning of the time windows. The existence of the customer time window constraints in the DARP, makes the region of feasible solutions highly non-convex, making it easy to move from a feasible solution to an infeasible solution. The setting of the more restrictive time limits makes the solution region more convex, by reducing the 'sea' of infeasible solutions that needs to be considered by the agents.

The use of preconditioned time windows in this study, like Cubillos *et al.* (2009), results in the more restrictive constraint of hard time windows. Both approaches using hard time windows, yielded better customer transit times than those obtained by Cordeau and Laporte (2003) and Jorgensen *et al*. (2007), as shown in Table 4.5, who used soft time windows. The use of hard time windows makes it easier to enforce optimisation of customer travel times. This is seen in Table 4.6 where customers exceeded the transit time constraint of 90 minutes from a sample of the benchmark solution sets. Similarly the use of soft time windows allows for a trade-off to be made between the servicing costs and customer service requirements. The use of soft time windows by Cordeau and Laporte (2003) would have also helped achieve their lower solution costs.

4.2.4 Speed of the ABS

Solutions for data sets Pr01 to Pr20 were obtained in reasonably short periods of time. The largest number of customer requests within these data sets, totalling 144 requests, required 6.7 minutes to obtain a solution. The ability of the ABS to quickly obtain a solution, makes the method feasible to use on a daily basis to schedule vehicle routes. The ABS yielded similar computational times for the data sets with the same number of customers. This is seen by the close clustering of data points in both Figures 4.3 and 4.4, indicating consistency in the running of the simulation. The computational times obtained will rely on how quickly the branch and bound algorithm in the model is able to find a feasible solution. The algorithm has been designed to re-initialise if infeasible branches are found, resulting in duplicate computations of the route segments.

The scalability of simulations to obtain solutions for varying problem sizes is an important feature, with a need to solve increasingly large problems which are more representative of the demands of real public transport systems. Importantly this study was able to obtain solutions in polynomial time for an NP-hard problem, for the large data sets of Pr21 and Pr22. Data sets Pr21 and Pr22 are 45% and 60% larger than Pr20, which has 144 customers, with the computational time required to obtain solutions increasing exponentially as seen in Figure 4.3. The exponential computational time is confirmed by the strong correlation coefficient of $r^2 = 0.95$ shown in Figure 4.4. The design of this study's ABS, requires each agent to place a bid for all awaiting customers at each time event. This results in an increase in computational time required as the number of agents and customers increases, as was also found by Salamon (2011). The increasing computational time for larger problem sizes imposes a limit on the further scalability of the model beyond 360 customer requests.

4.2.5 Practical significance and limitations of the ABS approach

This study has demonstrated the general applicability of agent based simulation as a method to use in simulating complex systems. The model developed adds further coherence on how ABSs should be applied, with this study having used numerous elements from the works of Segfried *et al.* (2009) and Salamon's (2011) Agentology methodology. The ability of the method developed to obtain solutions in linear time for larger data sets, both for an ABS approach and for the DARP, is significant.

Principally, the application of agent based simulation to solve the DARP is a novel approach with no known application of this method for the DARP found in literature by the author. Furthermore, from the literature reviewed for this study the author did not find any other applications of ABS which serviced more than 200 customers, with this study being able to service 360 customers.

The approach developed is limited by forcing a customer to be allocated to a vehicle at a bid event. The model does not consider the case when no customer should be scheduled for service at a specific time event, when all agent bids are very poor. If the model moved forward in time, a more optimal solution could possibly be obtained if the agent skipped a decision, dropped off a customer and waited for a future time event to make a decision on which customer to service next. It is clear from the higher solution costs obtained for all data sets, this study is lacking in the use of vehicle routing heuristics to minimise the travel distances incurred.

CHAPTER 5: CONCLUSION & RECOMMENDATIONS

5.1 Conclusion

It is evident that an agent based simulation method can be used to solve the static DARP. The method developed expands upon the relative unknown field of ABS, requiring agents to submit bids at each time event in the model to service a potential new customer. The decision to allocate a customer to a vehicle was decided by a central bidder agent. Pre-processing of the raw customer data to set modified pick-up and delivery time windows, given customer service constraints, was critical in reducing the solution search space and ensuring adherence to the time precedence of events.

The ABS yielded on average 67% better customer transit times and 6% better customer waiting times than known solutions which used common metaheuristic techniques, but at the expense of increased cost. The approach demonstrated its adaptability to solve problems of varying numbers of customers and vehicles. The method was able to yield consistently good customer servicing times, in short computational times. The ABS was unable to service customers at a lower cost than the benchmark solutions. However, the approach was able to solve large DARPs in polynomial time.

5.2 Recommendations

Based on the application and findings of the ABS approach developed, the following recommendations are made for future work:

1. The effect of different mutation rates and the number iterations in the genetic algorithm to obtain weighted constants for the objective function should be determined.
2. Methods to balance the number of customers between vehicles and the effect of geographically clustering customers to vehicles should be investigated.
3. The use of soft time windows for the ABS approach should be assessed.
4. Different software package should be investigated to determine if solutions for the DARP, using an ABS approach, can be obtained more quickly.

REFERENCES

Anylogic, 2015. [Computer software]

Berbeglia, G., Cordeau, J.F, Gribkovskaia, I. & Laporte, G., 2007**.** Static pick-up and delivery problems: a classification, *TOP*, 15 (1), 1-31.

Barbucha, D., & Jedrzejowicz, P., 2007. An Agent-Based Approach to Vehicle Routing Problem, *International Journal of Computer, Information, Systems and Control Engineering*, 1:2, 282-287.

Baugh J.W., Kakivaya, G.K.R., & Stone, J.R, 1998. Intractability of the Dial-A-Ride Problem and a Multi-objective Solution Using Simulated Annealing, *Engineering Optimization*, 30:2, 91-123

Campbell, I., 2015. *GA* [Matlab code].

Campbell, I., 2015. *Monte* [Matlab code].

Campbell, I., 2015. *Run* [Matlab code].

Campbell, I., 2015. *Evaluate* [Matlab code].

Colorini, A., & Righini, G., 2001. Modelling and optimizing dynamic dial-a-ride problems, *International Transactions in Operations Research*, 1, 155-166.

Cordeau, J.F., Gendrau, M., & Laporte, G., 1997**.** A Tabu Search Heuristic for the Static Dial A Ride Problem. *Transport Res B-Meth*, 37, 579-594.

Cordeau, J.F., 2003, A Branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, *54*, 573–586.

Cordeau, J.F., Gendreau, M., Laporte, G., Potvin, J.Y., Semet, F., 2002. A Guide to Vehicle Routing Heuristics, *The Journal of the Operational Research Society*, 53 (5), 512-522.

Cordeau, J.F. & Laporte, G., 2002. Tabu Search Heuristics for the Vehicle Routing Problem, *Metaheuristic Optimization Via Memory and Evolution: Tabu Search and Scatter Search*, 30 (2005) 145-163.

Cordeau, J.F., Gendreau, M., Hertz, A., Laporte, G. & Sormany, J.S., 2004, New Heuristics for the Vehicle Routing Problem, *Logistics Systems: Design and Optimization*, 2005, 279-297.

Cordeau, J.F. & Laporte, G., 2002. A Tabu Search Heuristics for the Static Multi-vehicle Dial-A-Ride Problem, *Transportation Research Part B*, 37 (2003), 579-594.

Cordeau, J.F & Laporte, G., 2003. The Dial-a-Ride Problem (DARP): Variants, modelling issues and algorithms, *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1 (2) 89-101.

Cordeau, J.F., Laporte, G., Savelburgh, M.W.P, & Vigo, D., 2007. Vehicle Routing, In Transportation, *Handbooks in Operation Research and Management Science*, 14, pp 367-428.

Cordeau, J.F & Laporte, G., 2007. The Dial-A-Ride Problem: models and algorithms, *Annals of Operations Research,* 153 (1) 29-46.

Cubillos, C., Urra, E., Rodriguez, N., 2009. Application of Genetic Algorithms for the DARPTW Problem, *Int. J. of Computers, Communications & Control,* IV (2), 127-136.

Darragi, N, Bon, P., Collart-Dutilleul, S., El Koursi, E.M., 2013. Tropos for Real-time Control System Modelling and Simulation. *4th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, HAL, France.

Desrosiers, J., Dumans, Y., Soumis, F., Taillefer, S., & Villeneue, D., 1986. An Algorithm for Mini-Clustering Handicapped Transport, *Technical Report G-91-02. GERAD,* HEC Montréal.

Fabri, A. & Recht, P., 2007. Online Dial-A-Ride Problem with Time Windows: an exact algorithm using status vectors, *Operations Research Proceedings*, 2006, 445-450.

Fu, L., 2002. Scheduling Dial-A-Ride Paratransit under Time Varying, Stochastic Congestion. *Transport Research B-Methods*, 36,485–506.

Hansen, J.B., 2010. The Dial-A-Ride Problem and Exploiting Knowledge about Future Customer Requests, Master's thesis, Aarhus University, Denmark.

Hillier F. S. & Lieberman G.J., 2010. *Introduction to Operations Research (9th ed.)*, McGraw-Hill, Singapore

Janssen, M.A., 2005. Agent Based Modelling, *International Society for Ecological Economics,* 11: 37.

Jaw, J.J., 1984**.** Solving A Large Scale Dial A Ride Problem Vehicle Routing And Scheduling Problem, *PhD Thesis,* Massachusetts Institute of Technology, Flight Transportation Laborator, Cambridge.

Jaw, J.J., Odoni, A.R., Psaraftis, H.N. & Wilson, N.H.M., 1986. A New Extension of Local Search Applied To the Dial-A-Ride Problem, *Transport Research Part B: Methodological,* 20B (3), 243-257.

Jorgensen, R. M., Larsen, J., & and Bergvinsdottir, K. B., 2007. Solving the Dial-a-Ride Problem Using Genetic Algorithms, *The Journal of the Operational Research Society,* 58 (10), 1321-1331.

Klientjie, J.P.C., 1995. *Verification and validation of simulation models,* European Journal of Operational Research, 82 (1), 145-162.

Lin., S., 1965. Computer solutions of the traveling salesman problem. *Bell System Technical Journal,* 44, 2245-2269.

Macal, C.M. & North, M.J., 2009. Agent Based Modelling and Simulation, *Proceedings of the 2009 Winter Simulation Conference,* 86-96.

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

Matworks, 2014. MATLAB [Computer Software].

Murray, A.T., Davis, R. & Stimson, R.J., 1998. Public Transportation Access, *Transportation Research Part D: Transport and Environment*, 3 (5) 319- 328.

Montgomery, D.C, & Runger, G.C, 2011. Applied Statistics and Probability for Engineers. John Wile & Sons Inc., Asia.

Psaraftis, H.N., 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transport Science*, 14, 130–154.

Rekiek B, Delchambre A., & Saleh H.A., 2006. Handicapped person transportation: An application of the grouping genetic algorithm. *Engineering Applications of Artificial Intelligence* 19,511–520.

Repast, 2015. http://repast.sourceforge.net/ [Computer software]

Salamon, **T**, 2011. *Design of Agent-Based Models*. Zivonin: Tomas Bruckner.

Savelsbergh, M.W.P, & Sol, M., 1995. The General Pick-up and Delivery Problem, *Transportation Science*, 29 (1) 17-29.

Segfried, R., Lehmann, A., Khyanari, R.E.A, Kiesling, T., 2009. A Reference Model for Agent-Based Modelling and Simulation, *Proceedings of the 2009 Spring Simulation Multi-conference,* Article 23.

Sexton T.R., & Bodin, L.D., 1985. Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling, *Transportation Science*, 19 (4), 378-410.

Transit Cooperative Research Programme, 2009. *Guidebook for Rural Demand-Response Transportation Measuring, Assessing and Improving Performance*. TCRP report 136. Washington, DC.

Toth, P. & Vigo, D., 2002. The Vehicle Routing Problem, *Society for Industrial and Applied Mathematics*.

Tuomisto, J.T. & Tainio, M., 2005. A way of reducing health, environmental, and other pressures of urban traffic: a decision on trip aggregation. *BMC Public Health*, 5.

Van Breedam, A., 2001. Comparing Descent Heuristics and Metaheuristics for the Vehicle Routing Problem, *Computers & Operations Research,* 28, 293-315.

Vokrinek, J., Komenda, A. & Pechoucek, M., 2010. Agents Towards Vehicle Routing Problems, *Proc. of 9th Int. Conf. on Autonomous Agents and Multi-agent Systems (AAMAS).*

Wilensky, U. 1999. NetLogo. http://ccl.northwestern.edu/netlogo/. Centre for Connected Learning and Computer-Based Modelling, North-western University, Evanston, IL.

Xiang, Z., Chu, C., & Chen, H., 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints, *European Journal of Operational Research*, 174 (2), 1117–1139.

Zeddini, B., Temani, M., Yassine, A. & Ghedira, K., 2008. An agent-oriented approach for the dynamic vehicle routing problem. *International Workshop on Advanced Information Systems for Enterprises*, 70–76.

Zidi, I., Mesghouni, K., Zidi, K. & Ghedira, K., 2012. A Multi-Objective Simulated Annealing For The Multi-Criteria Dial A Ride Problem, *Engineering Applications of Artificial Intelligence*, 1121-1131.

Appendix A: Logic Flowcharts of ABS

Figures A.1 to A.10 provide the programming logic used to build the ABS in Matlab.

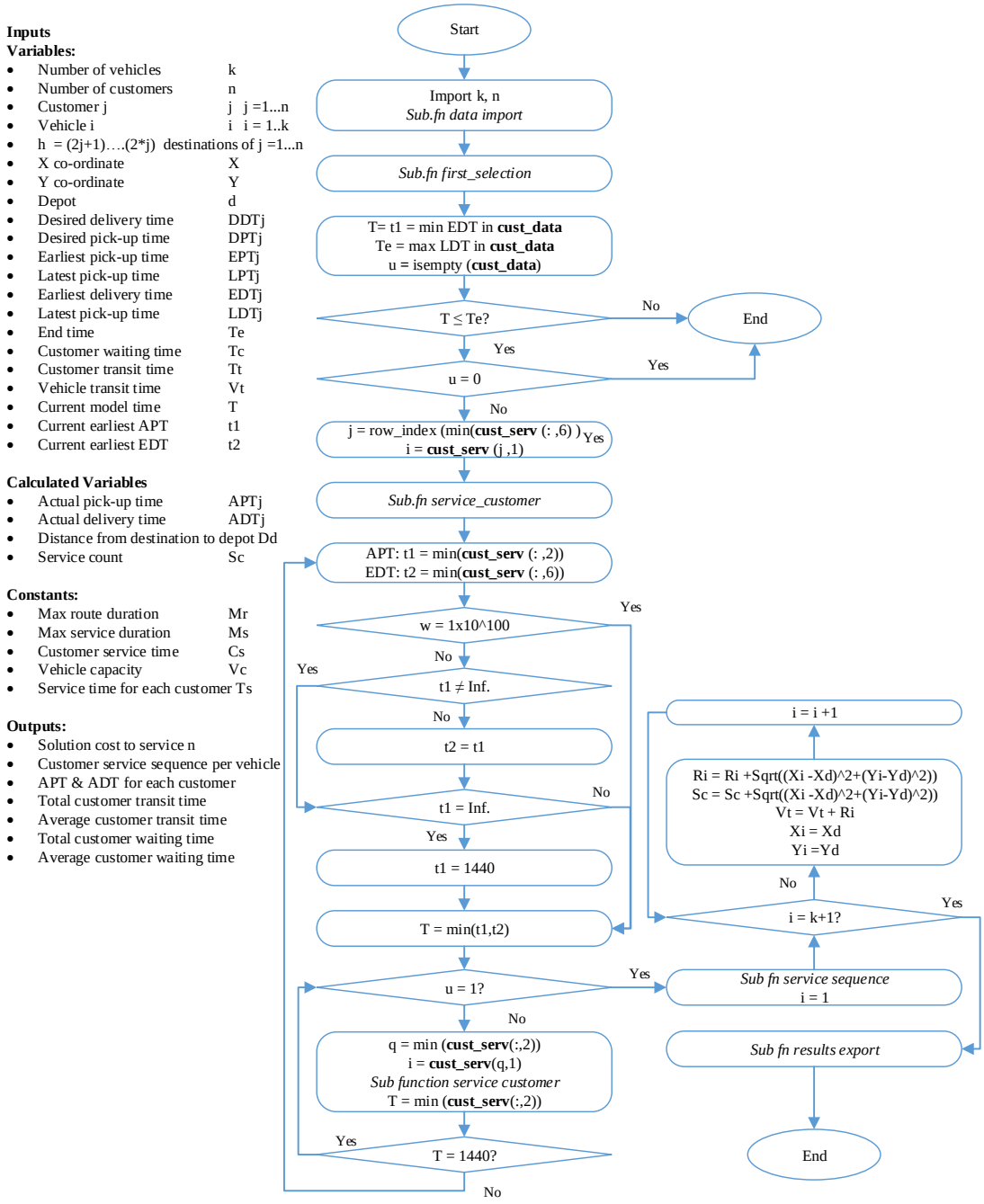


Figure A.1: Primary programming logic flow chart of the ABS model

Sub.fn data_import

Inputs
Variables:

- Number of vehicles k
- Number of customers n
- Customer j j =1...n
- Vehicle I i i = 1..k
- X co-ordinate X
- Y co-ordinate Y
- Depot d
- Desired Delivery Time DDTj
- Desired Pick-up Time DPTj
- Earliest Pick-up Time EPTj
- Latest Pick-up Time LPTj
- Earliest Delivery Time EDTj
- Latest Pick-up Time LDTj

Calculated Variables

- Actual Pick-up Time APTj
- Actual Delivery Time ADTj
- Distance from destination to depot Dd
- Service count Sc
-

Constants:

- Max route duration Mr
- Max service duration Ms
- Customer service time Cs
- Vehicle capacity Vc
- Service time for each customer Ts

Arrays
fleet_loc = [Xi Yi i Ci Ri Si]
Xi X co-ordinate of i
Yi Y co-ordinate of i
i Vehicle index
Ci Current load of i
Ri Route duration of i
Si Service duration of i

customer_time = [EPTj LPTj EDTj LDTj]
customer_loc = [Xj Yj Xh Yh]
cust_data = [i APTj ADTj EPTj LPTj EDTj LDTj Xj Yj Xh Yh Dd * * Ss j]
cust_serv = [i APTj ADTj EPTj LPTj EDTj LDTj Xj Yj Xh Yh Dd * * Ss j]
cust_completed = [i Sc APTj ADTj]

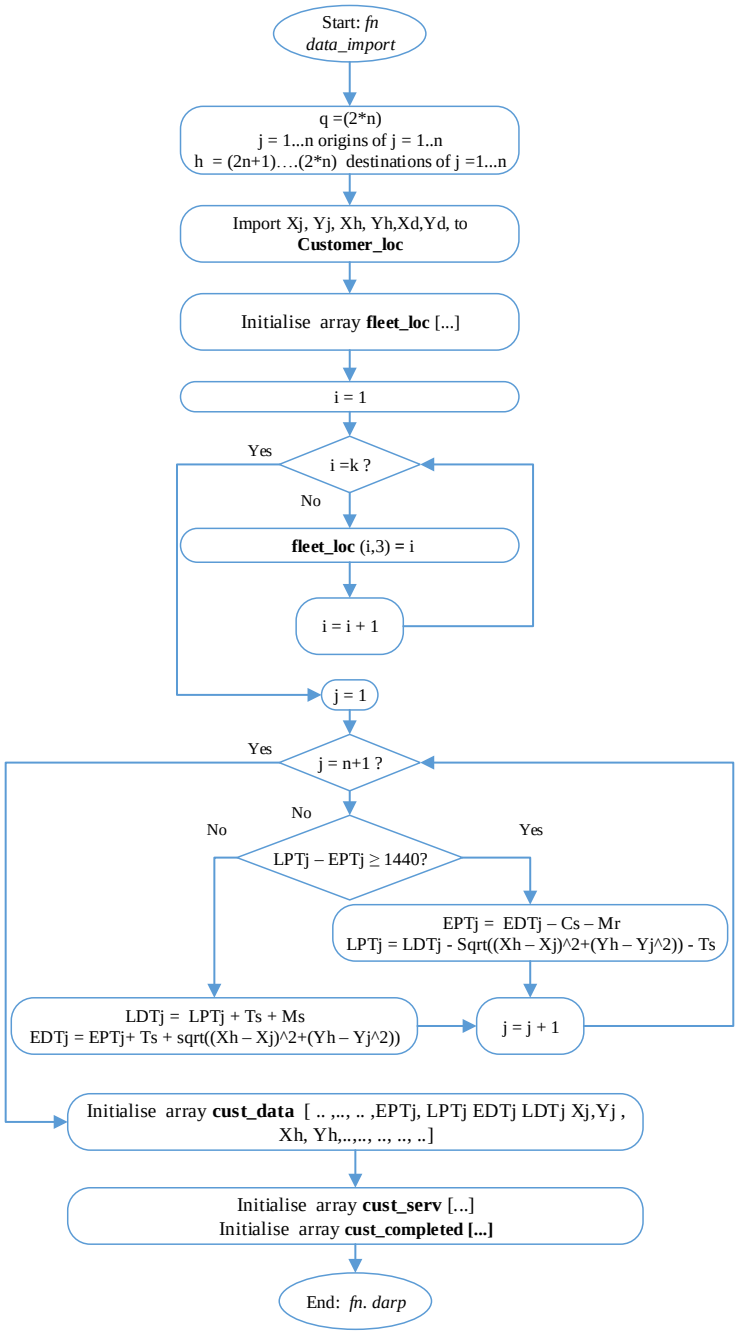


Figure A.2: Programming logic of the pre-processing of data in the ABS

Sub.fn first_selection

Inputs
Variables:

- Number of vehicles k
- Number of customers n
- Customer j j j =1...n
- Vehicle I i i = 1..k
- X co-ordinate X
- Y co-ordinate Y
- Depot d
- Desired Delivery time DDT
- Desired Pick-up time DPT
- Earliest Pick-up time EPT
- Latest Pick-up time LPT
- Earliest Delivery time EDT
- Latest Pick-up time LDT

Calculated Variables

- Actual Pick-up Time APT
- Actual Delivery Time ADT
- Distance from destination to depot Dd

Constants:

- Max route duration Mr
- Max service duration Ms
- Customer service time Cs
- Vehicle capacity Vc
- Service time for each customer Ts

Arrays
fleet_loc = [Xi Yi i Ci Ri Si]
Xi X co-ordinate of i
Yi Y co-ordinate of i
i Vehicle index
Ci Current load of i
Ri Route duration of i
Sti Service count of i

Cust_data = [i APTj ADTj EPTj LPTj EDTj LDTj Xj Yj Xh Yh Dd * * Ss j]
Cust_serv = [i APTj ADTj EPTj LPTj EDTj LDTj Xj Yj Xh Yh Dd * * Ss j]
Cust_completed = [i Sc APTj ADTj]

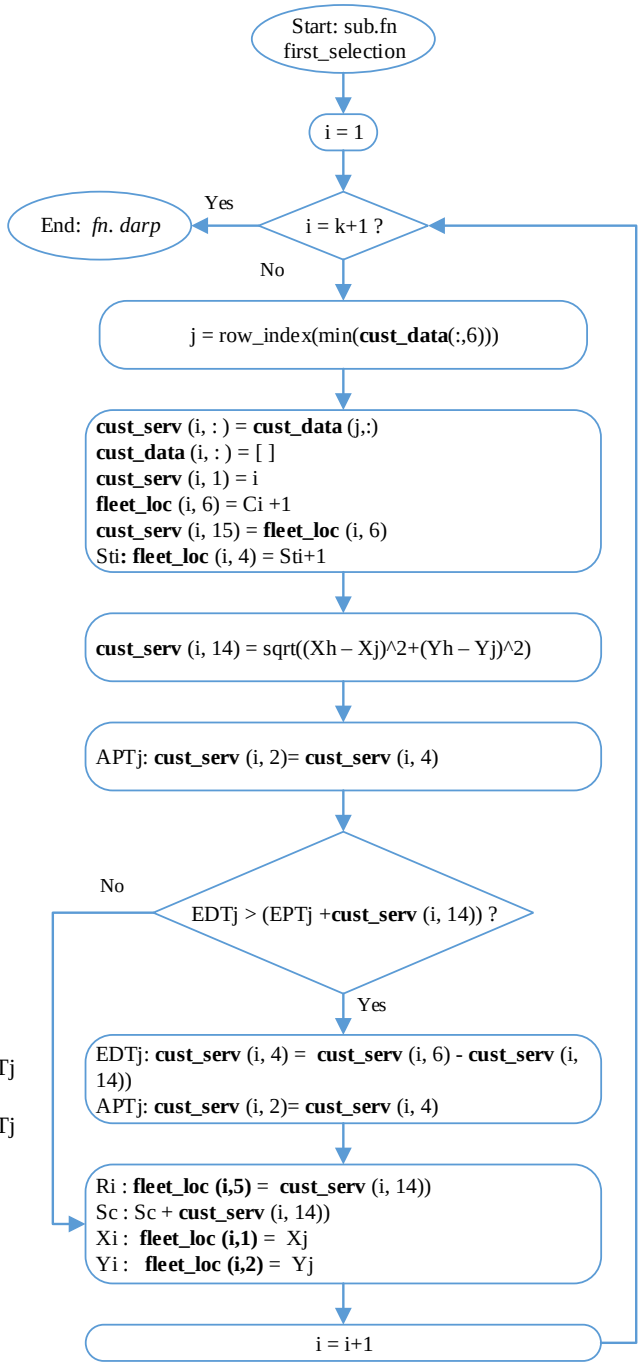


Figure A.3: Programming logic of the first customer selection method for the ABS

Sub.fn service_customer

Inputs

Variables:

- Number of vehicles k
- Number of customers n
- Customer j j =1...n
- Vehicle i i i = 1..k
- h = (2j+1)….(2*j) destinations of j =1...n
- X co-ordinate X
- Y co-ordinate Y
- Depot d
- Desired delivery time DDTj
- Desired pick-up time DPTj
- Earliest pick-up time EPTj
- Latest pick-up time LPTj
- Earliest delivery time EDTj
- Latest pick-up time LDTj
- End time Te
- Customer waiting time Tc
- Customer transit time Tt
- Vehicle transit time Vt
- Current model time T
- Current earliest APT t1
- Current earliest EDT t2

Calculated Variables

- Actual pick-up time APTj
- Actual delivery time ADTj
- Distance from destination to depot Dd
- Service count Sc
-
-

Constants:

- Max route duration Mr
- Max service duration Ms
- Customer service time Cs
- Vehicle capacity Vc
- Service time for each customer Ts

Outputs:

- Solution cost to service n
-

Arrays

fleet_loc = [Xi Yi i Ci Ri Si]

Xi	X co-ordinate of i
Yi	Y co-ordinate of i
i	Vehicle index
Ci	Current load of i
Ri	Route duration of i
Si	Service count of i

cust_data = [i APTj ADTj EPTj LPTj EDTj LDTj Xj Yj Xh Yh Dd * * Ss j]
cust_serv = [i APTj ADTj EPTj LPTj EDTj LDTj Xj Yj Xh Yh Dd * * Ss j]
cust_completed = [j i Sc APTj ADTj]

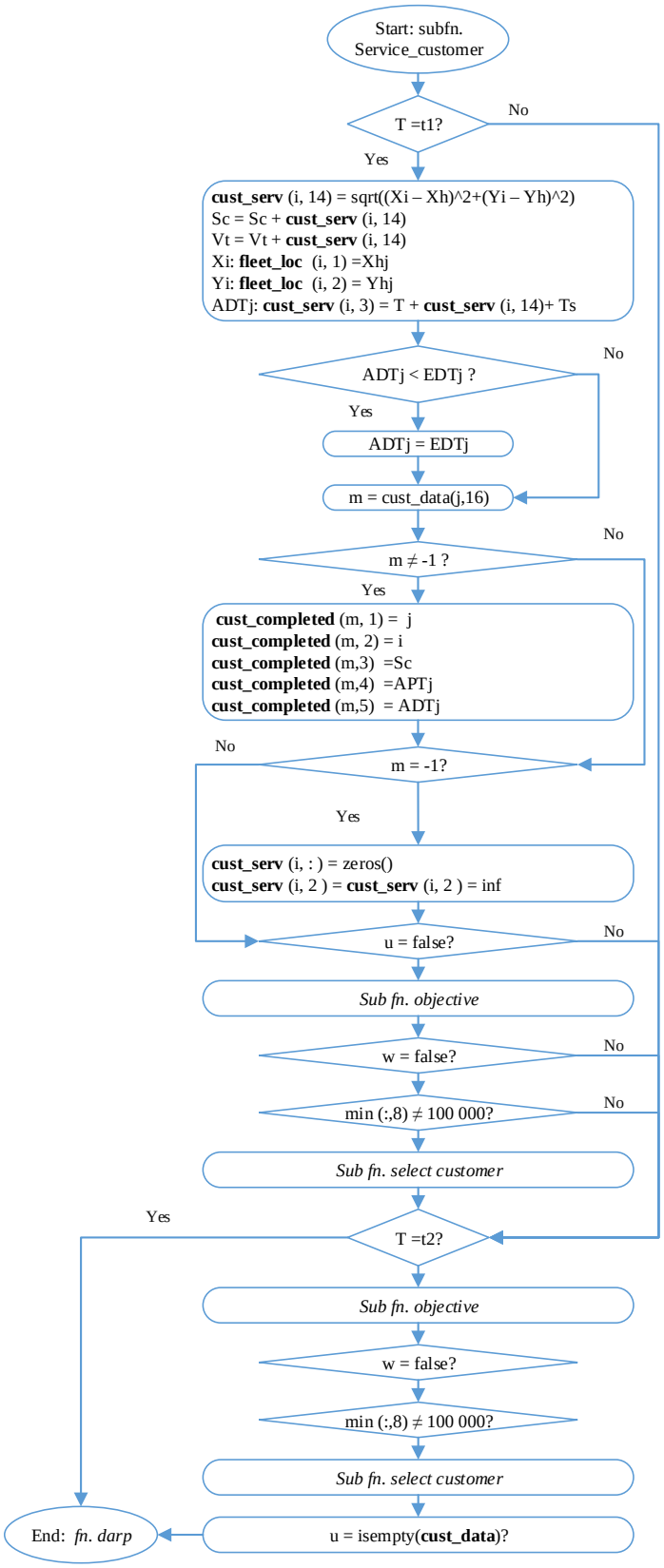


Figure A.4: Programming logic of the ABS's general customer selection method

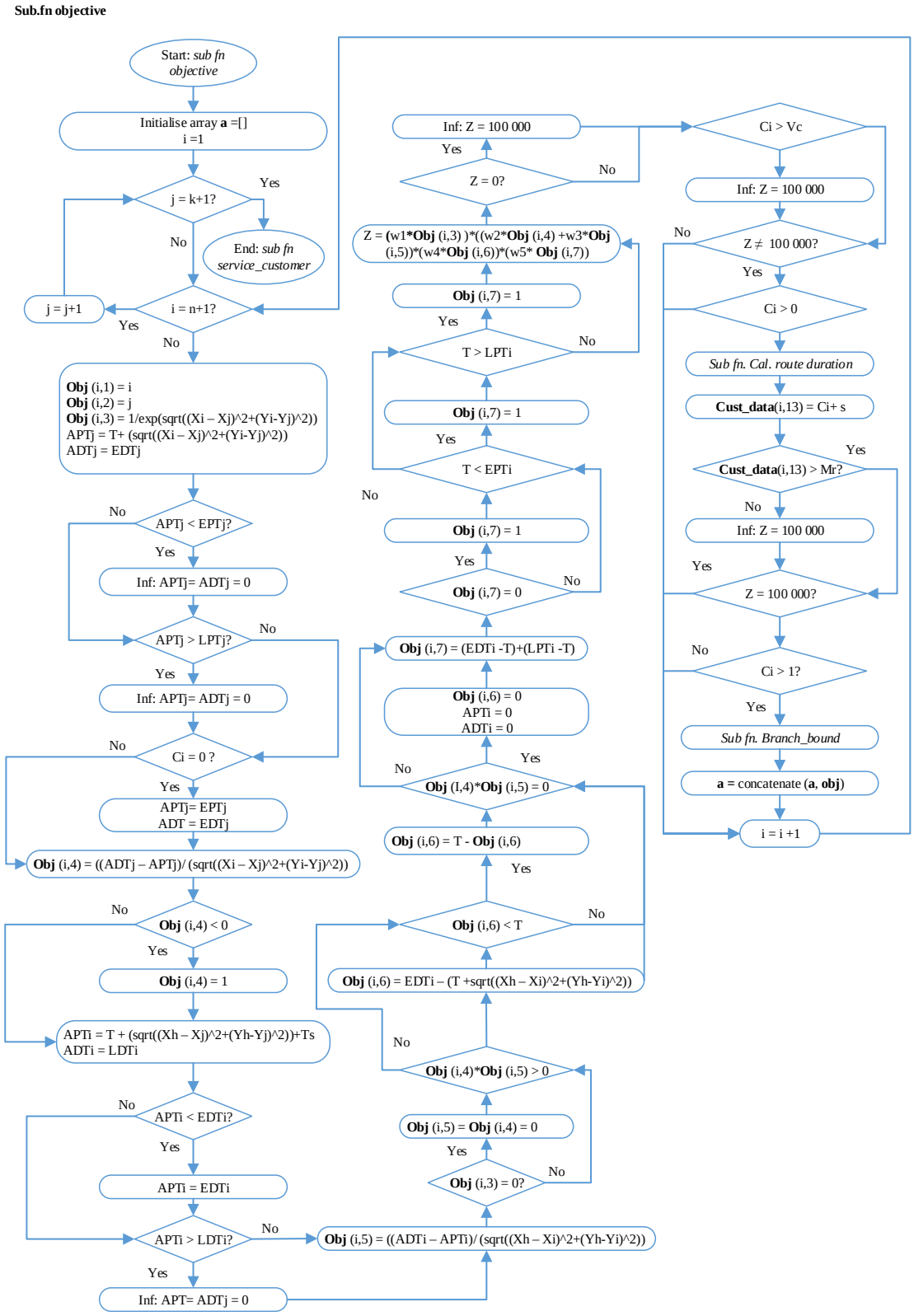


Figure A.5: Programming logic of the ABS's objective function

Sub.fn select_customer

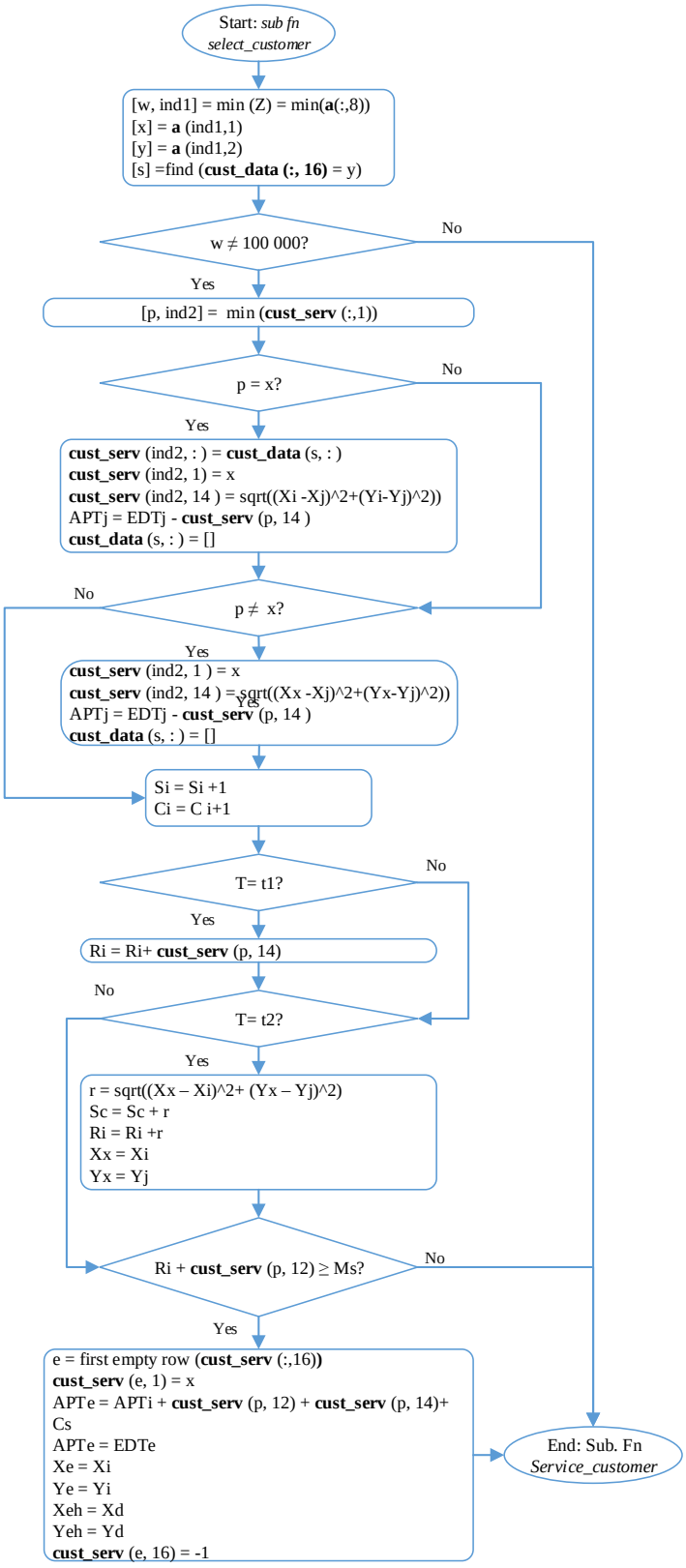


Figure A.6: Programming logic used by the ABS to allocate a customer to an agent

Sub.fn route_duration

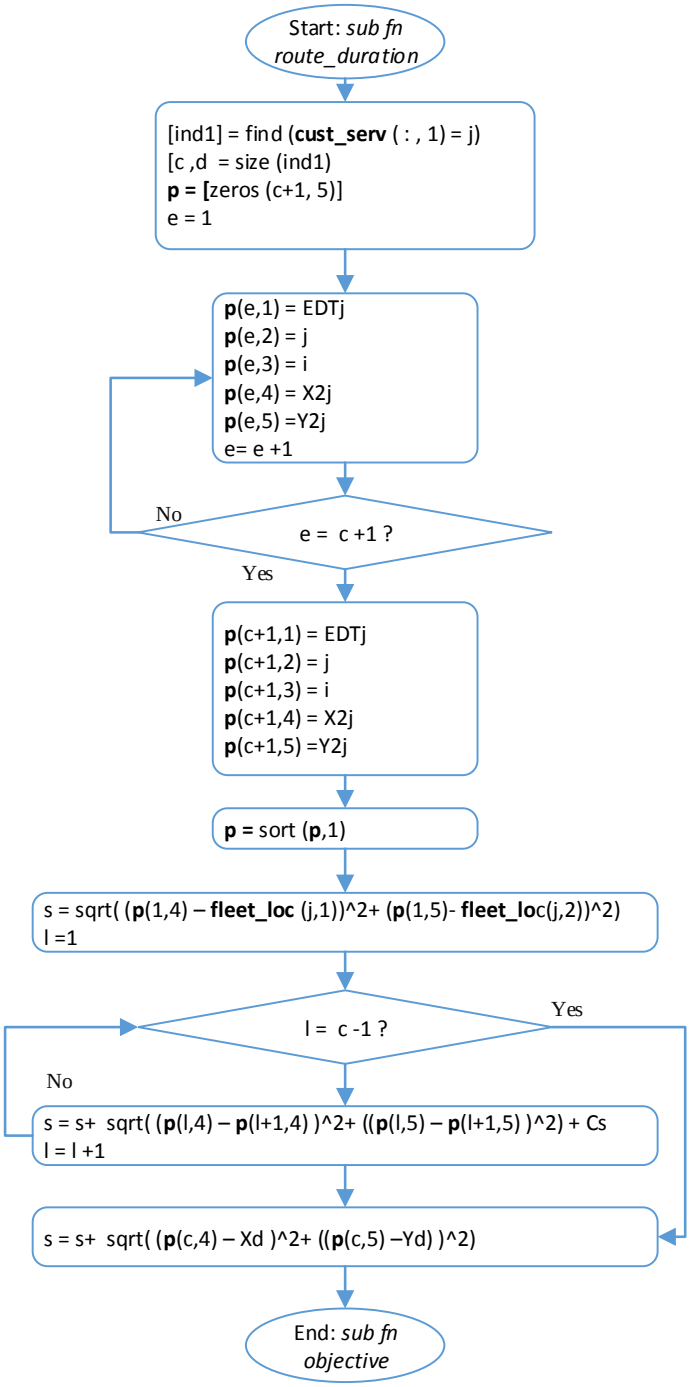


Figure A.7: Programming logic of the ABS to determine an agent's route duration

Sub.fn service_sequence

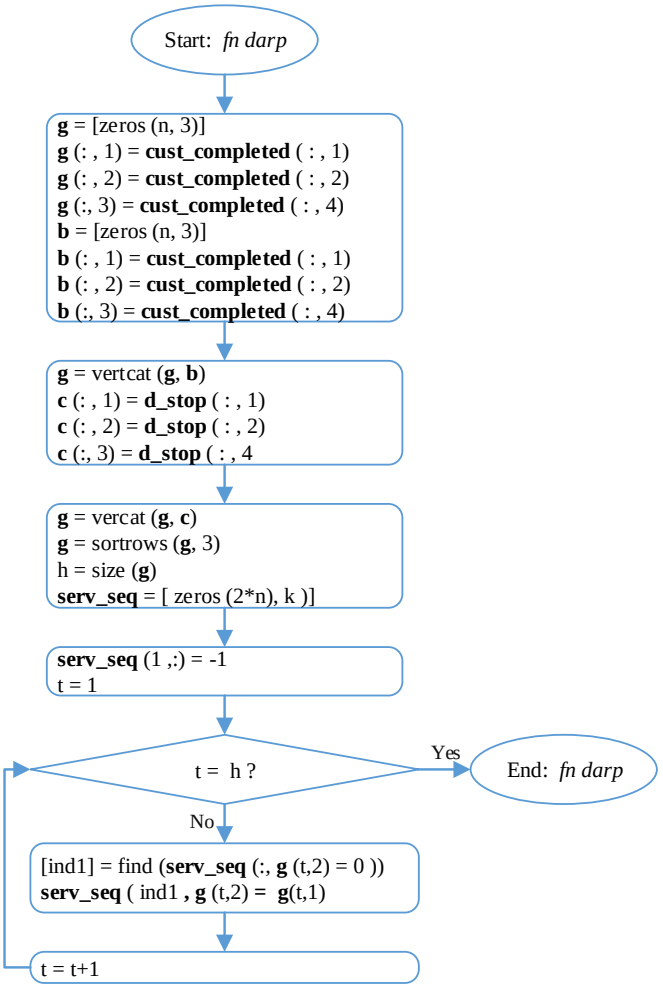


Figure A.8: Programming logic flow of the ABS to determine an agents customer service sequence

Sub. fn branch_bound

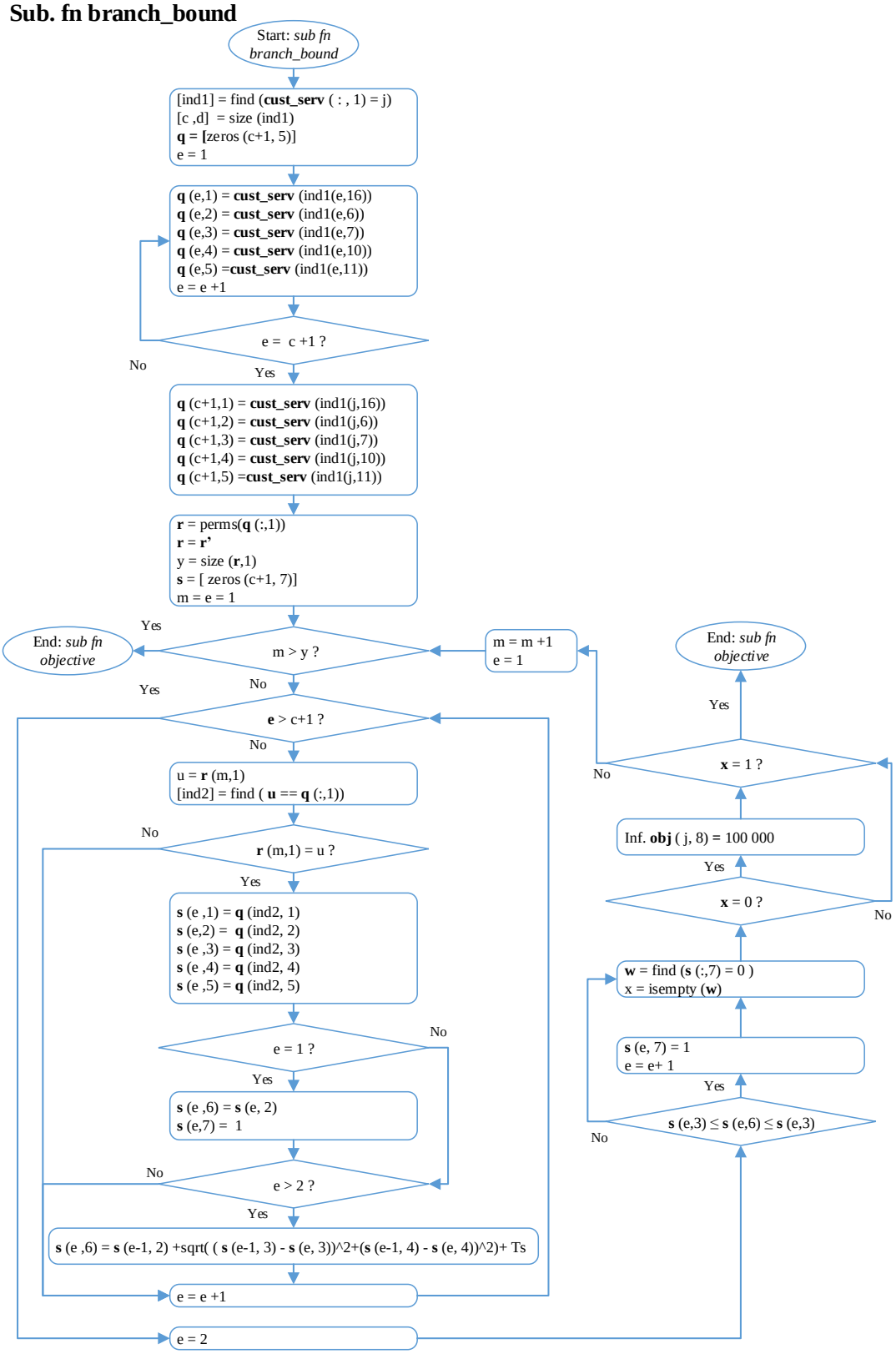


Figure A.9: Programming logic of the ABS's branch and bound algorithm

Sub. fn results_export

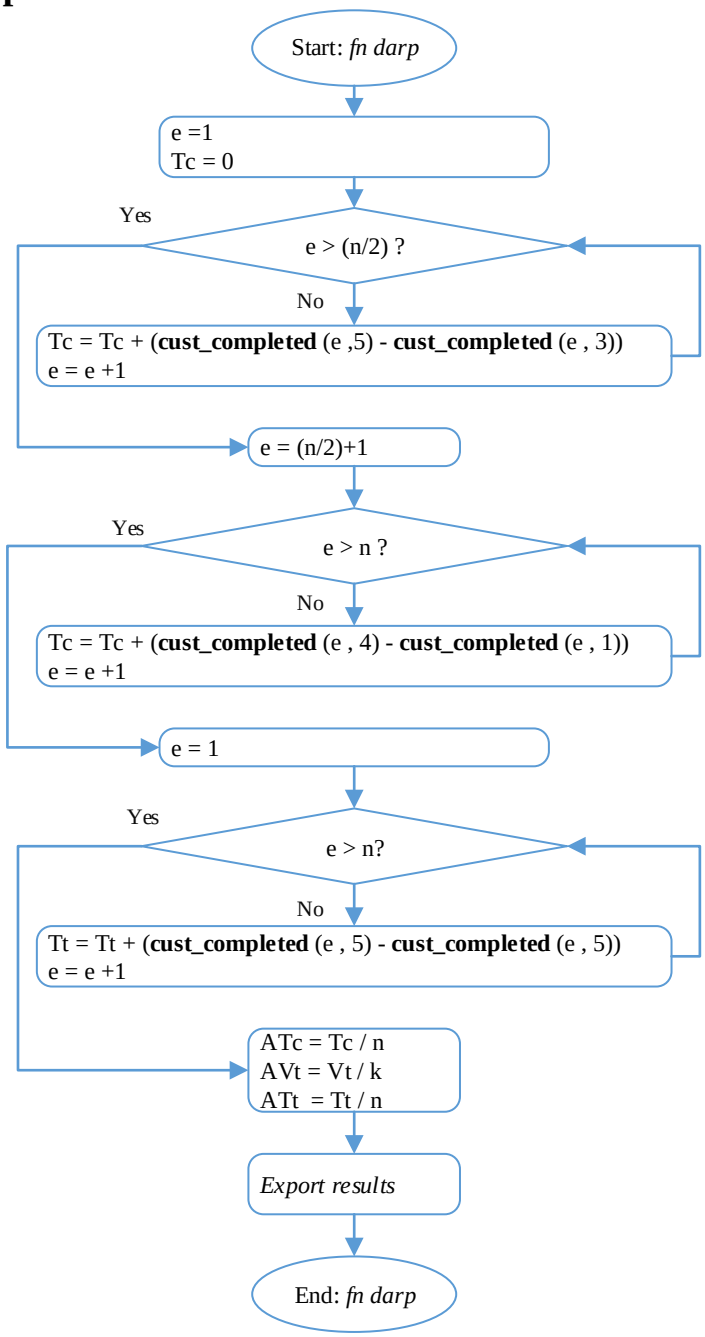


Figure A.10: Programming logic used by the ABS to export the model's results

Appendix B: MATLAB Code of ABS

```
function  darp(k)
%Import variables
k = xlsread('data_settest.xlsx',1,'A1');    % k no. of vehicles
n= xlsread('data_settest.xlsx',1,'B1')/2    % n no. of customers
max_route_duration = 480;
veh_capacity = 6;
max_ride_time = 90;
service_time = 10;
soln_cost = 0;
c_time= 0;
customer_loc = [];
w1 = 0.157;
w2 = 1;
w3 =0.9572;
w4 = 0.4854;
w5 = 0.6787;

data_import           %Sub fn.
first_selection       %Sub fn.

%Global variables needed for model:
end_time = max(cust_data (:,7));
c_wait_t = 0;                       %Customer waiting time
c_transit_t = 0;                    %Customer transit time
u = isempty(cust_data);             %Check if customers still awaiting service
a = [];                              %Obj fn array
d_stop = [];
v_transit_t = 0;                    %Initial transit time of veh. To 1st customers
t1= min(cust_serv(:,6));            %Move to first time event based on EDT
t2= min(cust_data(:,6));
time = t1;
h = 0;                              %Customer index for sub fn route duration
```

```
while time <= end_time
   if u == 0
      [val, j] = min(cust_serv(:,6));
      v = cust_serv(j,1);              %Veh. index at time event
      service_customer;                %Sub fn.
      t1 = min(cust_serv(:,2));        %Select earliest APT
      t2 = min(cust_data(:,6));        %Select EDT

      if w == 1.0e+100                  %No feasible customer pick up at time event
        if t1 ~= Inf                    %Move to next closest time event
          t2 = t1;
        end
        if t1 == Inf
          t1 = 1440;
        end
      end
      time = min(t1,t2);                %Move earliest time event (APT or EDT)
      end
   if u == 1 %All cust. have been selected for service, remaining cust. in transit to be dropped off
     [val, j] = min(cust_serv(:,2));
     v = cust_serv(j,1);
     service_customer;
     t1 = min(cust_serv(:,2));
     time = t1;
     if time == inf
       time = 1440;
     end
   end
end
service_sequence;       %Sub fn.

%Return vehicles to depot
for v = 1:k
   fleet_loc(v,5)= fleet_loc(v,5)+(sqrt((fleet_loc(v,1)-depot(1,1))^2+(fleet_loc(v,2)-depot(1,2))^2));
```

```
    soln_cost = soln_cost+(sqrt((fleet_loc(v,1)-depot(1,1))^2+(fleet_loc(v,2)-depot(1,2))^2));
    v_transit_t =  v_transit_t + fleet_loc(v,5) ;
    fleet_loc(v,1) = depot(1,1);
    fleet_loc(v,2) = depot(1,2);
    [ind1] = find(serv_seq(:,v)== 0, 1, 'first');
    if serv_seq(ind1-1, v)~= -1
        serv_seq(ind1, v) = -1;
    end
end
customer_time;
serv_seq;
soln_cost
results_export;
e = cputime

%Sub fn: Performs customer pick-up & drop-off events by agents
    function service_customer
        if time == t1 %cust drop off after pick up
            %k travelling to cust.  pick up point
            cust_serv(j,14) =sqrt((fleet_loc(v,1)-cust_serv(j,10))^2+(fleet_loc(v,2)-cust_serv(j,11))^2);
            soln_cost = soln_cost+ cust_serv(j,14);
            v_transit_t =  v_transit_t+ cust_serv(j,14);
            fleet_loc(v,1)= cust_serv(j,10);
            fleet_loc(v,2)= cust_serv(j,11);
            cust_serv(j,3) = time + cust_serv(j, 14)+ service_time;    %ADT

            if cust_serv(j,3) < cust_serv(j,6)  %Non-overlapping time windows
               cust_serv(j,3) = cust_serv(j,6);
            end

            m = cust_serv(j,16);
            if m ~= -1 %Linked to veh. not returning to depot
               cust_completed (m,1) = cust_serv(j,16);
               cust_completed (m,2) = cust_serv(j,1);
```

```
        cust_completed (m,3) = cust_serv(j,15);
        cust_completed (m,4) = cust_serv(j,2);
        cust_completed (m,5) = cust_serv(j,3);
        fleet_loc (v, 4) = fleet_loc (v, 4) - 1; % Decrease veh. load
        fleet_loc (v, 5) = fleet_loc (v, 5)+ cust_serv(j,14)+ service_time;
    end
    if m == -1 %veh returning to depot, due to max. ride time exceeded
        v_transit_t = v_transit_t + fleet_loc(v,5);
        fleet_loc(v,5) = 0;
        c = [cust_serv(j,16) cust_serv(j,1) cust_serv(j,15) cust_serv(j,2) cust_serv(j,3)];
        d_stop= vertcat(d_stop,c); % Depot stops by k
    end
    cust_serv (j,:) =zeros();
    cust_serv (j,2) =Inf;
    cust_serv (j,6) =Inf;

    %Current veh. to make a decision on which cust. to service next
    if u == 0
      objective;
      if w == 0
         if  min(a(:,8)) ~= 1.0e+100    % All obj fn infeasible. don't select cust
          select_customer;         %Sub. fn
         end
      end
    end
end
if time == t2 % Selecting  from cust_data  array, veh empty
    objective;
    if w == 0
        if  min(a(:,8)) ~= 1.0e+100
            select_customer;
        end
        if min(a(:,8)) == 1.0e+100  %No feasible customers, move to next EDT event
            [val, j] = min(cust_data(:,6));
```

```
            cust_data(j,6) = cust_data(j,6)+ 1;
            if cust_data(j,6) > cust_data(j,7)
              cust_data(j,6)= cust_data(j,7);
            end
          end
        end
      end
    u = isempty(cust_data);
    end


%Sub fn: Determines Z score of servicing specific customer
   function objective
      [r, c] = size(cust_data);
      obj = NaN(r, 8);
      a = [];
      for j = 1:k
      obj = [];
        for i = 1: r %r no. cust. still to be serviced
          obj (i, 1) = fleet_loc(j,3);    %veh_index
          obj (i, 2) = cust_data(i, 16);  %cust_index
          obj(i,3) = log(sqrt((fleet_loc(j,1)-cust_data(i,8))^2+(fleet_loc(j,2)-cust_data(i,9))^2));%Dist from
k veh loc to possible cust.
          cust_data(i,2)= time +(sqrt((fleet_loc(j,1)-cust_data(i,8))^2+(fleet_loc(j,2)-cust_data(i,9))^2));%
APT if veh. had to go directly  to cust.
          cust_data(i,3)= cust_data(i,6);


           if cust_data(i,2) < cust_data(i,4)  %EPT feasibilty check
              cust_data(i,2)= 0;             %Cannot pick up cust. earlier than EPT
              cust_data(i,3)= 0;
           end
           if  cust_data(i,2)> cust_data(i,5)  %LPT feasibilty check
              cust_data(i,2)= 0;
              cust_data(i,3)= 0;
           end
           if fleet_loc(j,4)== 0
```

```
            cust_data(i,2)= cust_data(i,4);
            cust_data(i,3)= cust_data(i,6); %Empty veh. can pick up cust. at EPT
          end

        obj(i,4) = (cust_data(i,3)-cust_data(i,2))/(sqrt((fleet_loc(j,1)-cust_data(i,8))^2+(fleet_loc(j,2)-
cust_data(i,9))^2)); %Pickup window width to travel time ratio
          if obj(i,4) < 0                %APT after EDT
            obj(i,4) = abs(obj(i,4));
          end
          if obj(i,4) == 0  %APT = EDT
            obj(i,4) = 1/(sqrt((fleet_loc(j,1)-cust_data(i,8))^2+(fleet_loc(j,2)-cust_data(i,9))^2));
          end
          cust_data(i,2)= time+(sqrt((cust_data(i,10)-cust_data(i,8))^2+(cust_data(i,11)-
cust_data(i,9))^2))+ service_time;
          cust_data(i,3)= cust_data(i,7);

          if  cust_data(i,2) < cust_data(i,6) %EDT feasibilty check
            cust_data(i,2) = cust_data(i,6);
            %Cal. ADT cannot occur before desired cust.r EDT
          end
          if cust_data(i,2) > cust_data(i,3) % LDT feasibility check
            cust_data(i,2)= 0;
            cust_data(i,3)= 0;
          %EDT cannot be after LDT, infeasible
          end

        obj(i,5) = (cust_data(i,3)-cust_data(i,2))/(sqrt((cust_data(i,10)-
cust_data(i,8))^2+(cust_data(i,11)-cust_data(i,9))^2));%Dropoff window width to travel time ratio
          if  obj(i,3)== 0; %Don't cal. ratio for infeasible cust.
            obj(i,4) = 0;
            obj(i,5) = 0;
          end
          if (obj(i,4)*obj(i,5))> 0 %Determine min in-transit service time

            obj(i,6) = cust_data(i,6)-(time+(sqrt((cust_data(i,10)-cust_data(i,8))^2+(cust_data(i,11)-
cust_data(i,9))^2))+ service_time);
```

```
        if obj(i, 6) < time
          obj(i,6) = time - obj(i,6);
        end
      end
      if    (obj(i,4)*obj(i,5)) ==  0
          obj(i,6)= 0;
      end
    cust_data(i,2)= 0;
    cust_data(i,3)= 0;

    obj(i,7)= (cust_data(i,6)-time)+(cust_data(i,5)-time);
      if obj(i,7)== 0;       %Case when current time centred between EDT & LPT
        obj(i,7)= 1;
      end
      if time < cust_data(i,4)
        obj(i,7) = 0;       %Prevent cust. with pick-up times far in advance from being favoured
      end
      if time > cust_data(i,5)
        obj(i,7) = 0;       %Prevent cust. with pick-up times far in advance from being favoured
      end

    obj(i,8) =  ((w1*obj(i,3))*((w2*obj(i,4))+(w3*obj(i,5)))* (w4*obj(i,6))*(w5*obj(i,7)));%Z
      if obj(i,8) == 0       %Inf. Z value set to large M value, veh. will not select as possible candidate
        obj(i,8) = 1.0e+100;
      end

      if  fleet_loc (j, 4) ==  veh_capacity
        obj(i,8) = 1.0e+100   ; %Full veh., additional customers cannot be added, Z set to large M
value
      end
      %check if feasible to added cust. to veh route with
      %exceeding max veh. travel time
      s = 0;
      if obj(i,8) ~= 1.0e+100  ;
       if fleet_loc(j,4) >= 1 % only do check if veh is loaded
```

```
            h = i;
            route_duration % Sub fn.
            cust_data (i,13) = fleet_loc(j,5)+ s;

            if cust_data (i,13) >  max_route_duration;
              obj(i,8) = 1.0e+100   ; %Max  veh route duration reached, additional cust. cannot be added, Z set to large M value
            end
            if obj(i,8) ~= 1.0e+100
             if fleet_loc(j,4) >= 1
                branch_bound
             end
            end
           end
          end
      a= vertcat(a,obj);%Combines all Z values for each veh. into 1 array for evaluation in select_customer sub fn.
      w = isempty(a);
     end
     end
  end

%Sub fn: Selects next customer to be serviced
  function select_customer
     [w, ind1] = min(a(:,8));
     [x] = a(ind1,1) ;   %Veh. index
     [y] = a(ind1,2);    %Cust. no.
     [s]= find(cust_data(:,16) == y);          %Index of cust. y in cust_data array
     if w ~= 1.0e+100                          %Only select cust. if at least 1 veh. has a feasible Z value
        [val, ind2] = min(cust_serv(:,1));      %Must not overwrite cust. currently in transit
        if v == x                              %Only service cust. if current veh. has lowest Z value
           cust_serv (ind2, :)= cust_data(s,:);
           cust_serv (ind2, 1) = v;
           cust_serv(ind2,14) =sqrt((fleet_loc(v,1)-cust_serv(ind2,8))^2+(fleet_loc(v,2)-cust_serv(ind2,9))^2);
           %for selected cust. determine intial APT before moving to
```

```
            %next time event
            cust_serv (ind2, 2) = cust_serv (ind2, 6)-cust_serv (ind2,14);
            cust_data(s,:) = [];
          end

        if v ~= x % Another veh. has a better Z value at current time
            cust_serv(ind2,:) = cust_data(s,:);
            cust_serv(ind2,1) = x;
            cust_serv(ind2,14) =sqrt((fleet_loc(x,1)-cust_serv(ind2,8))^2+(fleet_loc(x,2)-
cust_serv(ind2,9))^2);
            cust_serv (ind2, 2) = cust_serv (ind2, 6)-cust_serv (ind2,14);
            cust_data(s,:) = [];
        end
        fleet_loc (x, 6)= fleet_loc (x,6)+1;        %Service count
        fleet_loc (x, 4) = fleet_loc (x, 4)+1;      %Increase veh loading
        if time == t1
        fleet_loc (x,5) = fleet_loc (x,5)+ cust_serv(ind2,14);  %Increase veh. route duration, pick up leg
        end
        cust_serv(ind2,15) = fleet_loc (x, 6);      %Service count

        if time == t2 % Soln cost when next pick up, no drop off
            r = sqrt((fleet_loc(x,1)-cust_serv(ind2,8))^2+(fleet_loc(x,2)-cust_serv(ind2,9))^2);
            soln_cost = soln_cost + r;
            fleet_loc (x, 5) = fleet_loc (x, 5)+ r; %Veh travel time to pickup
            fleet_loc(x,1)= cust_serv(ind2,8);
            fleet_loc(x,2)= cust_serv(ind2,9);
        end
         if  (fleet_loc (x,5)+ cust_serv(ind2,12))>= max_route_duration  %Check if veh. must complete
route & return to depot
            [ind3] = find(cust_serv(:,1)== x);
            [c, d]= size(ind3);
            d = [zeros(c,2)];
            for l = 1: c
               p(l, 1) = cust_serv(ind3(l,1), 6);
               p(l, 2) = cust_serv(ind3(l,1), 14);
```

```
            end
            [ind4] = find(max(p(:,1)));
            [e] = find(cust_serv(:,16)== 0, 1, 'first');
            %Add phantom service cust., representing depot
            cust_serv (e, 1) = x;
            cust_serv (e, 2) = p(ind4,1)+p(ind4,2)+ cust_serv(ind2, 14)+ service_time;%APT arrive at
depot
            cust_serv (e, 6) = cust_serv (e,2);     %APT = EPT
            cust_serv (e, 8) = cust_serv (ind2, 10);
            cust_serv (e, 9) = cust_serv (ind2, 11);
            cust_serv (e, 10) = depot (1,1);
            cust_serv (e, 11) = depot (1,2);
            fleet_loc (x, 6) = veh_capacity;
            cust_serv (e, 15) = fleet_loc (x, 6);
            cust_serv (e, 16) = -1 ;              %Depot index
        end
    end
  end

%Sub fn: Calculate vehicle k's route duration if customer n added to route
  function route_duration
    [ind1] = find(cust_serv(:,1)== j);
    [c d]= size(ind1);
    p =[zeros(c+1, 5)];
    b = isempty(p);
    if b == 0
      for l = 1: c %Current cust. in tranist in veh
        p(l, 1) = cust_serv(ind1(l,1),6);   %EDT
        p(l, 2) = cust_serv(ind1(l,1),16);  %cust index
        p(l, 3) = cust_serv(ind1(l,1),1);   %veh index
        p(l, 4) = cust_serv(ind1(l,1),10);  %x drop-off co-ord
        p(l, 5) = cust_serv(ind1(l,1),11);  %y drop-off co-ord
      end
     %Add potential cust to route
        p(c+1, 1) = cust_data(h,6);
```

```
            p(c+1, 2) = cust_data(h,16);
            p(c+1, 3) = cust_serv(ind1(l,1),1);
            p(c+1, 4) = cust_data(h,10);
            p(c+1, 5) = cust_data(h,11);
        p = sort(p,1);    % Sort EDT in asceding order
        s= sqrt((p(1, 4)- fleet_loc(j,1))^2+ (p(1, 5)- fleet_loc(j,2))^2); %Dist from current veh loc to first cust for veh in cust_serv
        % servicing of all cust in transit for veh k
           for l = 1: (c-1)
              s = s+ sqrt((p(l, 4)- p((l+1),4))^2+ (p(l, 5)- p((l+1), 5))^2)+service_time;
           end
        s= s+ sqrt((p(c, 4)-depot(1,1))^2+(p(c, 5)-depot(1,2))^2); % travel time from last cust to depot
        end
    end

%Sub fn: Checks drop-off time feasibility of adding new customer to route
    function branch_bound
        [ind1] = find(cust_serv(:,1)== j);
        [c,d]= size(ind1);
        q =[zeros(c+1, 5)];
        b = isempty(q);
        if b == 0
           for l = 1: c %Current custs. in tranist in veh
              q(l, 1) = cust_serv(ind1(l,1),16);  % Cust index
              q(l, 2) = cust_serv(ind1(l,1),6);   % EDT
              q(l, 3) = cust_serv(ind1(l,1),7);   % LDT
              q(l, 4) = cust_serv(ind1(l,1),10);  % X drop off
              q(l, 5) = cust_serv(ind1(l,1),11);  % Y drop off
           end
            %Add potential cust to route
              q(c+1, 1) = cust_data(h,16);
              q(c+1, 2) = cust_data(h,6);
              q(c+1, 3) = cust_data(h,7);
              q(c+1, 4) = cust_data(h,10);
              q(c+1, 5)= cust_data(h,11);
```

```
end
r= perms(q(:,1));
r = r';
y = size(r, 1);
s = [zeros(c+1,7)];

for m = 1:y
 for l = 1: c+1
   u = r(m,l);
   [ind2] = find(u == q(:,1));
   if r (m,l) == u
      s (l,1) = q(ind2,1) ; % Cust index
      s (l,2) = q(ind2,2);  % EDT
      s (l,3) = q(ind2,3);  % LDT
      s (l,4) = q(ind2,4);  % X-drop
      s (l,5) = q(ind2,5);  % Y-drop
      if l == 1
         s (l,6) =  s (l,2);
         s (l,7) = 1;
      end
      if l >= 2
         s (l,6) = s (l-1,2)+ sqrt(( s (l-1,3)- s (l,3))^2+(s (l-1,4)- s (l,4))^2)+ service_time; %ADT
      end
   end
 end
 for  l = 2: c+1  % Feasabilty check
   if  s (l,3)  <= s (l,6) <= s (l,3)
      s (l,7) = 1;
   end
 end
 w = find(s (:,7) == 0);
 x = isempty(w);
 if x == 0
   obj(h,8) = 1.0e+100 ;
```

```
        end
        if x == 1
          break
        end
       end
    end

%Sub fn: Import data sets, adjusts time windows & create core arrays
    function data_import
     %Basic variable imports from data input file
     q= (2*n)+2; % q considers all customer origins & destinations in input file
     depot = [xlsread('data_settest.xlsx',1,'C2') xlsread('data_settest.xlsx',1,'D2')]; %depot location
     p=(n+2);                          % p only considers customer origins in input file
     r= p+1;                           % r only considers customer destinations in input file
     str3 = sprintf('%c%d:%c%d','C',3,'C',p);   % Create dynamic string based on input file
     str4 = sprintf('%c%d:%c%d','D',3,'D',p);
     str5 = sprintf('%c%d:%c%d','C',r,'C',q);
     str6 = sprintf('%c%d:%c%d','D',r,'D',q);
     str7 = sprintf('%c%d:%c%d','G',3,'G',p);
     str8 = sprintf('%c%d:%c%d','H',3,'H',p);
     str9 = sprintf('%c%d:%c%d','G',r,'G',q);
     str10 = sprintf('%c%d:%c%d','H',r,'H',q);
    customer_loc = [xlsread('data_settest.xlsx',1,str3) xlsread('data_settest.xlsx',1,str4)
xlsread('data_settest.xlsx',1,str5) xlsread('data_settest.xlsx',1,str6)];
    customer_time = [xlsread('data_settest.xlsx',1,str7) xlsread('data_settest.xlsx',1,str8)
xlsread('data_settest.xlsx',1,str9) xlsread('data_settest.xlsx',1,str10)];
    fleet_loc = [repmat(depot,k, 1) NaN(k,1) zeros(k,3)];% Fleet data: x & y position, index, current
loading, current route duration, serivce count

    for w = 1: k %Index each veh.
       fleet_loc(w, 3) = w;
    end

    for i= 1:n  %With hard time windows, time windows limits must be adjusted according to max ride
time & DRT
       if customer_time(i,2)- customer_time(i,1)>= 1440 %DDT cust.
```

```
        customer_time(i,1) = customer_time(i,3)- service_time - max_ride_time; %EPT

        customer_time(i,2)= customer_time(i,4)- (sqrt((customer_loc(i,1)-
customer_loc(i,3))^2+(customer_loc(i,2)-customer_loc(i,4))^2))-service_time;% LPT

      end

      if customer_time(i,4)- customer_time(i,3)>= 1440 %DPT cust.

        customer_time(i,3)= customer_time(i,1)+ (sqrt((customer_loc(i,1)-
customer_loc(i,3))^2+(customer_loc(i,2)-customer_loc(i,4))^2));% EDT

        customer_time(i,4)= customer_time(i,2)+ service_time + max_ride_time; %LDT

      end

    end


    cust_data= NaN(n, 16);

    cust_data = [zeros(n,1) NaN(n,2) customer_time  customer_loc NaN(n,2) zeros(n,2)];

    cust_serv  = [zeros(k*veh_capacity,1) NaN(k*veh_capacity ,15)];% Array of cust. currently being
serviced, size of max no. cust that can be transported

    cust_completed = zeros(n, 5);% Record of cust. that have been serviced

    for w = 1: n % index each cust.

      cust_data(w, 16) = w;

      cust_data (w,12) = sqrt((cust_data(w,10)-depot(1,1))^2+(cust_data(w,11)-depot(1,2))^2);  % Dist
veh to travel from  cust. drop-off to depot

    end

   end


%Sub fn: Each veh. at the depot selects its first customer to service

   function first_selection

     for j = 1:k  % Veh chooses cust. with EDT

        [val, ind] = min(cust_data(:,6));

        cust_serv (j, :)= cust_data(ind,:);

        cust_data(ind,:) = [];

        cust_serv(j,1) = fleet_loc(j,3);

        fleet_loc (j, 6)= fleet_loc (j, 6)+1;

        cust_serv(j,15) =  fleet_loc (j, 6);    % Service index

        fleet_loc (j, 4) = fleet_loc (j, 4)+ 1; % Veh. loading

        cust_serv(j,14) =sqrt((fleet_loc(j,1)-cust_serv(j,8))^2+(fleet_loc(j,2)-cust_serv(j,9))^2);

        cust_serv(j,2) = (cust_serv(j, 6));

        %EPT adjusted further to min cust. travel time for 1st cust & veh waiting time
```

```
        if ((cust_serv(j, 6)) > (cust_serv(j, 4)+ cust_serv(j, 14)))
          cust_serv(j,4) = (cust_serv(j, 6)-cust_serv(j, 14));
           cust_serv(j,2) = cust_serv(j, 4);% Adjusted APT
        end
        fleet_loc (j, 5) = cust_serv(j,14); % Time veh. takes to travel to first pick-up
        soln_cost = soln_cost + cust_serv(j,14);
        fleet_loc(j,1)= cust_serv(j,8);
        fleet_loc(j,2)= cust_serv(j,9);
    end
  end

%Sub fn: Determines service seq. k serviced customers
  function service_sequence
    g = [zeros(n,3)];
    g(:,1) = cust_completed (:,1);  %cust_index
    g(:,2) = cust_completed (:,2);  %veh_index
    g(:,3) = cust_completed (:,4);  % APT
    b = [zeros(n,3)];
    b(:,1) = cust_completed (:,1)+ n;
    b(:,2) = cust_completed (:,2);
    b(:,3) = cust_completed (:,5);  %ADT
    g = vertcat(g,b);
    u = isempty(d_stop);
    if u == 0
    c(:,1) = d_stop (:,1);
    c(:,2) = d_stop (:,2);
    c(:,3) = d_stop (:,4);
    g = vertcat(g,c);
    end
    g = sortrows(g,3);
    h = size(g);
    serv_seq = [zeros(2*n, k)];
    serv_seq (1,:) = -1; % Depot location represented by -1
    for t = 1:h
```

```
        [ind1] = find(serv_seq(:,g(t,2))== 0, 1, 'first');
        serv_seq (ind1, g(t,2)) =  g(t,1);
      end
 end

%Sub fn: Exports model results to MS Excel
   function results_export
      for i = 1:(n/2) % For DDT customers
         c_wait_t = c_wait_t +(cust_completed(i,5) - customer_time (i,3));
      end
      for j = ((n/2)+1): n % For DPT customers
         c_wait_t = c_wait_t + (cust_completed(j,4) - customer_time (j,1));
      end
      for i =1:n
      c_transit_t = c_transit_t+ (cust_completed(i,5) - cust_completed(i,4));
      end
      header = {'Customer No.', 'Vehicle No.', 'Service Sequence', 'APT', 'ADT',' ','Soln Cost',
'C_Wait_Time','C_Transit_Time',',','CPU Time',  ' ', 'Current X-Co-ord.', 'Current Y-Co-ord.', 'Veh Index',
'Veh Loading', 'Route Duration','Service Count', ','Service Sequence'};
      xlswrite('Darp Results.xlsx',header,'Results');
      xlswrite('Darp Results.xlsx',cust_completed,'Results','A2');
      xlswrite('Darp Results.xlsx', soln_cost,'Results','G2')
      xlswrite('Darp Results.xlsx',c_wait_t,'Results','H2');
      xlswrite('Darp Results.xlsx',c_transit_t,'Results','I2');
      xlswrite('Darp Results.xlsx',fleet_loc,'Results','m2');
      xlswrite('Darp Results.xlsx',serv_seq,'Results','t2');
      c_time = cputime
      xlswrite('Darp Results.xlsx',c_time,'Results','k2');
     end
end
```

Appendix C: Genetic Algorithm Weights

Table C.1: Initial random weights for the GA and DARP solution costs

Iteration	w_1	w_2	w_3	w_4	w_5	Soln. Cost
1	0.681	0.155	0.437	0.815	0.728	657.24
2	0.187	0.032	0.697	0.898	0.867	649.19
3	1.000	0.294	0.237	0.531	0.092	569.92
4	0.336	0.747	0.011	0.234	0.101	659.85
5	0.158	0.522	0.174	0.968	0.008	621.34
6	0.267	0.173	0.434	0.689	0.388	645.95
7	0.539	0.460	0.624	0.038	0.411	634.62
8	0.599	0.751	0.930	0.090	0.492	627.57
9	0.027	0.710	0.068	0.875	0.031	601.37
10	0.174	0.539	0.109	0.154	0.490	617.34

Iteration 3 was selected as the initial starting weights for use in the genetic algorithm.

Appendix D: ABS Output Solution

The complete set of ABS output results for data set PR01 are provided in Tables D.1 to D.4

Table D.1: Pr01 customer service times, sequence and allocated service vehicle

Customer No. (*i*)	Vehicle No. (*k*)	Service Sequence	APT_i	ADT_i
1	2	5	245.2909	259.6198
2	3	7	320.0046	336.9878
3	3	5	202.6131	220.0434
4	3	9	410.8996	427.4216
5	2	6	297.0406	312.9405
6	1	3	430.8966	449.041
7	3	3	192.8271	203.7453
8	2	4	209.6148	226.7656
9	3	1	99.75372	120.5723
10	3	6	255.9414	270.3063
11	3	2	174.9326	195.1799
12	3	8	374.1172	386.4421
13	2	7	333.4342	351.8942
14	2	2	116.7565	135.3957
15	3	10	394.9072	408.2561
16	1	1	87.18617	109.1065
17	2	1	86.23566	104.6566
18	3	11	413.5409	432.2895
19	2	8	458.2903	480.9064
20	3	4	174.0305	191.7433
21	3	12	419.084	437.2026
22	2	3	146.4749	165.0415
23	1	4	472.7808	487.0176
24	1	2	325.4851	343.6048

Table D.2: Performance metrics of Pr01

Soln Cost	C_Wait_Time	C_Transit_Time	CPU Time
241.5787	146.2718	414.0419	38.5156

Table D.3: Pr01 route duration and service count per vehicle

Current X-Co-ord.	Current Y-Co-ord.	Veh Index	Current Veh Loading	Route Duration	Service Count
-1.044	2	1	0	84.6023	4
-1.044	2	2	0	216.3594	8
-1.044	2	3	0	269.6514	12

In Table D.4 depot stops are represented by a -1. A customer pick-up event is represented by the customer number *n* and customer drop-off event by *n+24*.

Table D.4: Pr01 vehicle routes

Vehicle Route		
***k* = 1**	***k* = 2**	***k* = 3**
-1	-1	-1
16	17	9
40	41	33
24	14	20
48	38	11
6	22	44
30	46	7
23	8	35
47	32	3
-1	1	31
0	25	27
0	5	10
0	29	34
0	13	2
0	37	26
0	19	12
0	43	36
0	-1	15
0	0	39
0	0	4
0	0	18
0	0	21
0	0	28
0	0	42
0	0	45
0	0	-1

Appendix E: Results

Table E.1: Solutions from the ABS with incomplete solutions from benchmark data sets

Data Set	Cost of Soln.		Total Customer Transit Time		Total Customer Waiting Time	
	Benchmark	ABS	Benchmark	ABS	Benchmark	ABS
Pr04	573	845	-	1531	-	607
Pr06	801	1126	7273	2135	1375	888
Pr07	292	355	-	629	-	272
Pr08	495	564	-	1137	-	434
Pr09	672	847	5622	1740	323	573
Pr10	879	1225	7164	2257	721	976
Pr13	493	696	-	1266	-	589
Pr14	536	835	-	1541	-	698
Pr16	744	1092	7347	2181	449	976
Pr18	462	701	-	1198	-	531
Pr20	799	1182	7072	2218	362	1038

Key

Benchmark	Cordeau & Laporte (2003) solution using TS
-	No solution provided by Cordeau & Laporte (2003)
ABS	Agent Based Simulation solution
	ABS worse than Benchmark solution
	ABS better than Benchmark but worse than GA solutions
	ABS better than Benchmark & GA solutions

Table E.2: ABS CPU times for all data sets

Data Set	CPU Time [seconds]	Number of Customers	Number of Agents
Pr01	39	24	3
Pr11	30	24	3
Pr07	40	36	4
Pr17	37	36	4
Pr02	40	48	5
Pr12	42	48	5
Pr03	47	72	7
Pr08	43	72	6
Pr13	46	72	7
Pr18	31	72	6
Pr04	71	96	9
Pr14	70	96	9
Pr09	78	108	8
Pr19	139	108	8
Pr05	157	120	11
Pr15	167	120	11
Pr06	402	144	13
Pr10	260	144	10
Pr16	401	144	13
Pr20	289	144	10
Pr21	20254	264	25
Pr22	142326	360	35