Agent Based Simulation of the Dial-a-Flight Problem

D.T. Reddy

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 of Master of Science in Engineering.

Johannesburg, May 2018

Declaration

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

Signed this 24th day of May 2018

D.T. Reddy

Acknowledgements

I would like to thank the following people:

- Dr Ian Campbell and Dr Joke Buhrmann, for supervising and guiding me through this process.
- My Mother for all her love and support.
- My Father for his motivation and insight.

Abstract

Agent based simulation and modelling (ABSM) has been noted as a novel method in solving complex problems. This dissertation makes use of the ABSM method in conjunction with a Genetic Algorithm to find good solutions to the dial-a-flight problem. The task is to generate a schedule for a heterogeneous fleet of aircraft, with the objective to reduce operational cost but maintain customer satisfaction. By making use of booking list data from an air taxi business, operating in the Okavango Delta, two agent based models were designed, the first makes use of multi-criteria decision analysis (MCDA) and the other a method proposed by Campbell [7], to test their effectiveness against either upper bound or manual solutions. The solution quality varied between tests, with booking list sizes between 10 and 200 requests producing improvements to the upper bound and manual results with a mean improvement from the benchmarks of 1.61%. The method could also be refined further by adopting improvement mechanisms to final schedules or by making use of retrospective decision making aided by self learning techniques.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviation	Description
ABSM	Agent based simulation and modelling
ACO	Ant colony optimisation
C206	Cesna 206
C208	Cesna 208
CAS	Complex adaptive systems
CMVRPTW	Capacitative multi-vehicle routing problem with time windows
CVRP	Capacitative vehicle routing problem
CVRPTW	Capacitative vehicle routing problem with time windows
DAFP	Dial-a-flight problem
DARP	Dial-a-ride problem
EDT	Early departure time
GA	Genetic algorithm
ILP	Integer linear programe
LAT	Late arrival time
MARS	Modelling autonomous cooperating shipping companies
MCDA	Multi-criteria decision analysis
MCNF	Multi-commoditiy network flow
MIP	Mixed integer programe
PDP	Pick up and delivery problem
TSP	travelling salesmen problem
VRP	Vehicle routing problem

1 Introduction

This research investigates the use of agent based simulation and modelling (ABSM) in solving the dial-a-flight problem (DAFP). No significant results using ABSM have been achieved without the use of a complementary heuristic technique as seen in the work of Campbell [7]. In this Chapter the purpose, background, motivation and objectives for this research are presented.

1.1 Purpose of the study

Agent based models are a tool used to solve complex problems in operations research. These models have been employed to study air traffic control systems, distributed vehicle monitoring tasks and to solve vehicle routing problems [26]. In Campbell's research he used ABSM to generate better time nodes and flight legs for the DAFP that are constrained by a traditional integer linear program (ILP). This significantly reduced the number of variables to solve the dial-a-flight problem and thus makes finding a good solution faster. He then ran the newly formulated model with time discretions generated by the ABSM with a mixed integer program (MIP) to generate a result. This research aims to use ABSM to generate final solutions to the DAFP that are close to academic benchmarks as a continuation of Campbell's work.

1.2 Background

The airline industry has been at the forefront of information systems. They have employed the use of many optimisation tools and throughout the years the technology has moved forward with the industry. The next phase shift in the aviation industry is

the air taxi business model [13]. These businesses make use of light aircraft to transport passengers on demand from location to location. These locations are usually very close geographically. The complexity associated with this business model makes it difficult to effectively schedule the fleet of aircraft and keep the operational costs of the fleet low. This task requires either an expert in scheduling aircraft or an optimisation tool.

The method of optimisation used in this research was that of ABSM. This method has generally been used in sociological studies [42], but the method's fast execution and use of autonomous behaviour makes it a good candidate for finding good solutions to complex vehicle routing problems. The work conducted on vehicle routing problems thus far has focused predominantly on the capacitative vehicle routing problem (CVRP)[49], or has used the method as a complementary algorithm to conventional optimisation techniques [7].

The DAFP and other static vehicle routing problems have come to the forefront of the field of vehicle routing and have placed a lot of focus on the development of efficient algorithms to solve them. Figure 1.1 shows a many-to-many vehicle routing problem visualisation. The DAFP came about due to the need for an optimal method of scheduling aircraft. The key challenge for this particular problem is to develop a scheduling engine. The engine could be developed and tested using *agent based simulation and modelling* (ABSM). This methodology has been considered as a new and alternate method to solve problems that are highly constrained [7].

The DAFP falls into a class of problems known as the many-to-many static capacitative multi-vehicle routing problem with time windows (CMVRPTW) [6]. Many-to-many problems are characterised as having a set of nodes that constitute both pick up and drop off points. In the case of the DAFP, the vertices of the network represent airports. Algorithms have been largely aimed at the optimisation of single vehicle problems, but research into multi-vehicle routing has proven to be more valuable. This is because the results can be translated into real-world scenarios. This is especially true with regards to static pick up and drop off problems. These can be applied to many door-to-door transportation systems (such as the travelling salesman problem) as well as courier operations.

ABSM finds its roots in the study of adaptive systems [42]. It was traditionally used in examining system behaviours. Such simulations have been implemented in studies of bacterial spread, modelling pandemics, human social interaction as well as predicting

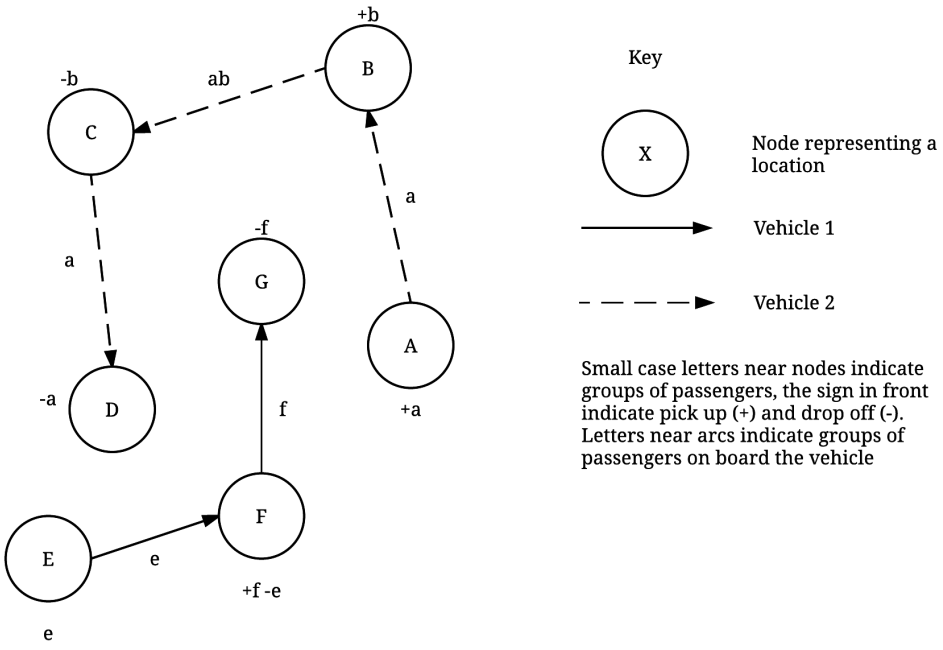

Figure 1.1: An example of a multi-vehicle DAFP *many-to-many* problem visualisation.

factors that influence failure. This method has made its way into the fields of economics, modelling fluctuations in stock and labour markets and industrial engineering, with the main focus on vehicle routing. This research aims at addressing the DAFP using this technique, by finding feasible solutions to this problem and comparing the solutions to academic benchmarks and manually generated schedules.

1.3 Wilderness air

The data used to conduct this research comes from an air taxi airline operating in the Okavango delta called Wilderness Air. Their primary target group are tourists. These tourists request transport to and from the 100 tourist camps the airline services. Wilderness air operates in Namibia, Zimbabwe, Zambia and South Africa. The organisation is based in Botswana, servicing approximately 49,000 passengers per year in this region alone. They are based at the Maun international airport [50]. This is essentially the hub of the operation since the majority of the flights arrive and depart from there. Typically tourists will visit the many safari camps in the area, but may also want to visit other natural attractions the region has to offer.

Figure 1.2: Wilderness air Cessna 206 G [50].

Figure 1.3: Wilderness air Cessna 208 B Grand Caravan [50].

Wilderness air currently use two types of light aircraft, namely the Cessna 206 G (C206) (Figure 1.2) and the Cessna 208 B Grand Caravan (C208) (Figure 1.3). The use of light aircraft makes the business cost effective and allows it to operate as an air taxi service. The Cessna 206 seats 5 passengers with the pilot and the Cessna 208 seats 11 with the pilot.

Currently, Wilderness Air employs experienced schedulers who compile the schedule and route the aircraft a few days in advance. These manual schedules are considered to be low cost. This poses a risk for the airlines since if these schedulers become unavailable it would take time to train new schedulers and the business would risk

operational cost at the expense of optimally scheduled aircraft. Since the passengers place requests a day or so in advance of the flight, the problem can be defined as a static DAFP [7] [43] [10].

1.4 Research motivation

Agent based simulations have the ability to model complex vehicle routing problems quickly and without the use of a complementary heuristic technique. Increased complexity and the need to relax assumptions in order to gain a better understanding of the problems means that ABSM is a viable option for researchers in the field of vehicle routing [7].

Vehicle routing problems pose interesting challenges in the logistics and transport industries. The dial-a-flight problem has significant implications for air taxi and air ambulance services [38]. There is a need to find optimal routes for vehicles to use, so as to reduce the operational cost and increase customer satisfaction [6]. Relatively fast computational time means that the method can be implemented on other types of vehicle routing problems that are more dynamic in nature.

The task was to develop a model in order to find solutions to the dial-a-flight problem. Given the test data and benchmarks provided by the University of the Witwatersrand [8], the results of this analysis should then be compared to benchmarked solutions and manually generated solutions provided by Wilderness Airlines. This will prove whether the use of agent based simulation of a multi-vehicle *many-to-many* problem is suitable for these problems. The analysis should also show what program structures, parameters and utility functions need to be investigated further to fully optimise the DAFP using this technique.

1.5 Objectives and research questions

The main aim of this research is to find good solutions to the DAFP. The solutions produced by the models will be judged against either the upper and lower bounds or the manual schedule provided from Wilderness Air. Good solutions are defined as solutions that have solution qualities near any of the benchmarked solutions.

The objectives are:

1. To develop an agent based simulation to produce schedules for the DAFP of high solution quality. Solutions to the problems must be an improvement or at worst within a range of 5-20% from the upper bound or manual solutions. This is the range of deviation found in literature pertaining to ABSM when applied to the DAFP [49][3].
2. To suggest methods of improvement for the agent based simulation so that full optimisation can be achieved.

This research aims to answer the following research questions:

1. What ABSM program structure best suits the agent model when applied to VRP?
2. What utility functions applied to the agent decision making faculties will produce the best results?
3. Once a successful model has been built, the solving time should be compared to similar problems.

1.6 Dissertation overview

This dissertation is structured in the following way. In chapter 1 the purpose,background and objectives of the study, as a continuation of the work conducted by Campbell [7], are discussed. The literature survey is discussed in chapter 2. The survey includes applicable multi-vehicle routing problems, agent based simulation and modelling and heuristic decision making techniques that can be used to solve the dial-a-flight problem. The method used to design the agent based models and the formulation of the dial-a-flight problem are presented in chapter 3. The models presented in chapter 3 are formulated using both ideas put forward by Campbell as well as make use of new techniques to solve the problem. The testing procedure as well as results and observations of the model are presented in chapter 4. In chapter 5, the findings from the experimentation as they relate to the research objectives are discussed. In chapter 6, a summary of the key conclusions and recommendations of this work are presented.

2 Literature Review

In this chapter a review of literature is conducted to give the reader context of the methods and techniques used in this paper. Firstly, multi-vehicle routing problems such as the pick up and delivery problem, the dial-a-ride problem and the dial-a-flight problem are defined. Secondly the subject of agent based simulation and modelling are discussed in terms of its application and use in vehicle routing problems. Lastly different heuristic techniques such as multi-criteria decision making, Monte Carlo methods and genetic algorithms are discussed.

2.1 Multi-vehicle routing problems

Multi-vehicle routing problems (MVRP) fall into a class of well known combinatorial optimisation problems. The aim of these problems are to route a set of vehicles through a graph in order to optimise a set of conditions (usually to minimise distance travelled or operational cost). This paper will focus on three MVRP's, the standard pick up and delivery problem, the dial-a-ride problem and the dial-a-flight problem.

2.1.1 Pick up and delivery problems (PDP)

Pick up and delivery problems (PDP) constitute a class of vehicle routing problems that require an algorithm to optimise the pick up and delivery of objects in a system. They can be divided into static and dynamic problems. In static problems the demand from the customers is known before the schedules are developed. In the dynamic version of the problem the requests for pick up are generated as the simulation progresses.

Static PDPs can be defined in a general framework put forward by Berbeglia [6]. The framework is as follows:

Let

$$G = (V, A) \tag{2.1}$$

be a complete and directed network with vertex set

$$V = \{0, ..., n\} \tag{2.2}$$

where node 0 represents the depot and the rest of the nodes represents customers or positions. The set of arcs are represented by:

$$A = \{(i, j) : i, j \in V, i \neq j\}. \tag{2.3}$$

Each arc $(i, j) \in A$ has a non-negative cost associated with it. For PDPs distance can be used as the cost.

At each node there has to be some set of commodities that require transport. Let:

$$H = \{1, ..., p\} \tag{2.4}$$

where H represents the set of commodities that require transportation. Each node can either be a demand or supply commodity node. Lastly the amount of commodity supplied or demanded can be defined by a commodity matrix $D = (d_{ih})$ where a positive d_{ih} represents the amount of commodity h supplied by the node i and a negative d_{ih} represents the amount of commodity h required at node i. For the system to be balanced the sum of the supply and demand has to be in equilibrium:

$$\sum_{i \in V} d_{ih} = 0$$

for each h

$$. \tag{2.5}$$

For the multi-vehicle problem, there has to be a set that represents the available set of vehicles to solve the problem. This can be represented by:

$$K = \{1, ..., m\}. \tag{2.6}$$

Any route can be defined by a circuit over a set of nodes starting and ending at the depot. The PDPs consist of at maximum m routes such that:

- All demand in the system is satisfied.
- The capacity of a vehicle is never exceeded.
- No transshipment of commodities occur.
- The sum of route costs / distance covered is minimised.

Depending on how the problem is structured, each vehicle may have defined variable costs such as distance per kilometre, idling costs or lateness penalties. The objective function is then defined as the minimisation of these costs, but may include multiple objectives if customer satisfaction constraints are included in the problem definition.

2.1.2 The dial-a-ride problem (DARP)

The Dial-a-Ride problem (DARP) can be considered a special case of the vehicle routing pickup and delivery problem. The problem consists of a set of customers who request pickup and drop off's from a taxi service [9]. The problem has additional constraints related to:

- Time windows, and
- Maximum ride times

The DARP was developed to solve problems in door-to-door transport systems with its main aim to reduce transportation costs. There are two forms of this problem, single vehicle and multi-vehicle (generally seen as a homogeneous fleet of vehicles). Both problems are aimed at finding the compromise between customer satisfaction and minimisation of operational costs. The satisfaction constraints are dependent on two major factors, the first being delivery time deviation and second excess ride times. In the past heuristic methods have been the most popular choice when solving these problems with the use of local search heuristics [24] and tabu search [10] proving to be effective in finding solutions to the problem. A schematic of the problem can be seen in Figure 2.1. Other methods such as dynamic programming are suitable in finding optimal solutions but are generally not suited to large problems [6].

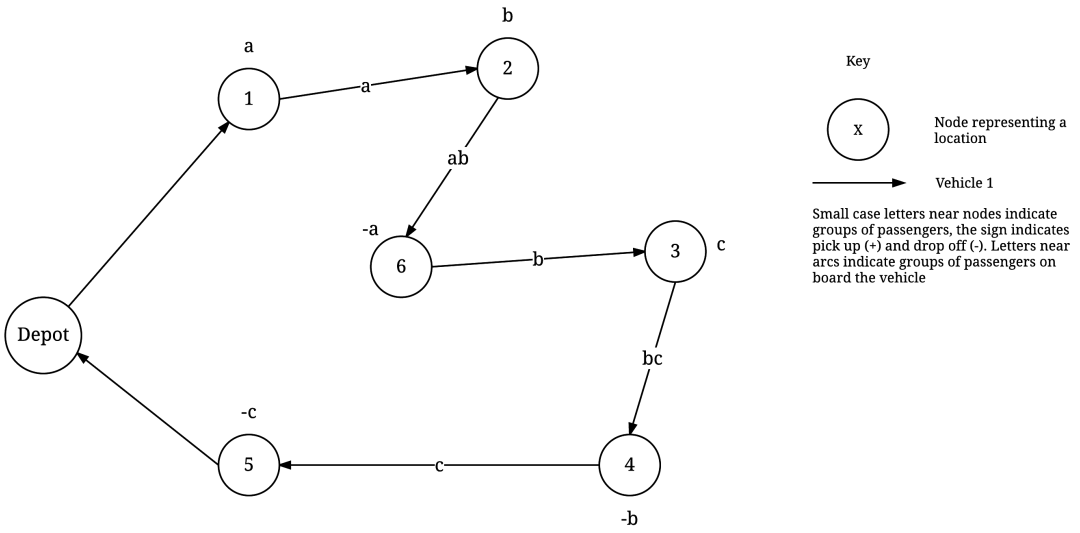

Figure 2.1: Schematic of the single vehicle dial-a-ride problem [6].

In the multi-vehicle problem, the assignment of requests for the vehicles must be done in conjunction with the design of the overall schedule taking into consideration that the vehicles may be starting from the same depot. For multi-vehicle routing, the solution space of the problem is much larger and thus more difficult to solve [6]. In a survey conducted by Cordeau et al. [9] it was noted that emphasis in literature has been placed on solving the DARP. Excellent heuristic techniques exist. The decomposition algorithm is a viable option for solving the larger instances of the problem. No mention was made of an agent based method.

2.1.3 The dial-a-flight problem (DAFP)

The dial-a-flight problem (DAFP) falls into the same class of problem as the multi-vehicle DARP. It was formulated as a way of optimising air taxi services [13]. Between the DARP and DAFP, there are a few differences to take into consideration. These differences are [4]:

- The request for the service from an air taxi business would occur in advance so the schedule can be developed using a static framework.
- Usually the request for a service means multiple destination pairs. Travellers who book flights will usually expect to return to their original location.

- The set of pick-up and drop-off points is relatively small compared to that of the DARP since only a finite number of airports can be considered.
- The model should take into account regulations for aircraft use such as:
  - number of hours each aircraft can be in flight,
  - how many flights each pilot can service, and
  - aircraft maintenance.

Due to the on-demand nature of the problem, the service provider may also reject or alter the requests at some cost to the company. The services are measured by how many flight legs the passenger is on before they reach their final destinations, how many plane changes are required and lastly how long they wait before they reach their final destinations.

Each aircraft can occupy two states;

1. In flight: refers to an aircraft that is currently on route to a destination.
2. Idle: refers to an aircraft that is on the ground waiting for arriving customers.

The idle time also includes the turnaround time of the aircraft. The turnaround time refers to the amount of time the aircraft needs to get ready to take off, having landed at a destination. Espinoza et al. [14] gave a flow chart of the key events from the perspective of the aircraft (Figure 2.2).

Espinoza et al. [14] approached the problem by formulating an integer multi-commodity flow model as well as a parallel local search. They found optimal solutions to 23 instances provided by an air taxi service provider [14] [15]. These schedules were unavailable for this research. The solutions are accurate, however, due to the nature of the problem the algorithms used are not flexible or fast. Since the DAFP is NP-hard, construction heuristic techniques to solve the problem is the next logical step in this field of research.

Campbell [7] made contributions by providing size reduction heuristics such as group aggregation and geographic heuristics. Campbell also introduced a composite variable formulation and lastly agent routing variable generation. Campbells work on agent

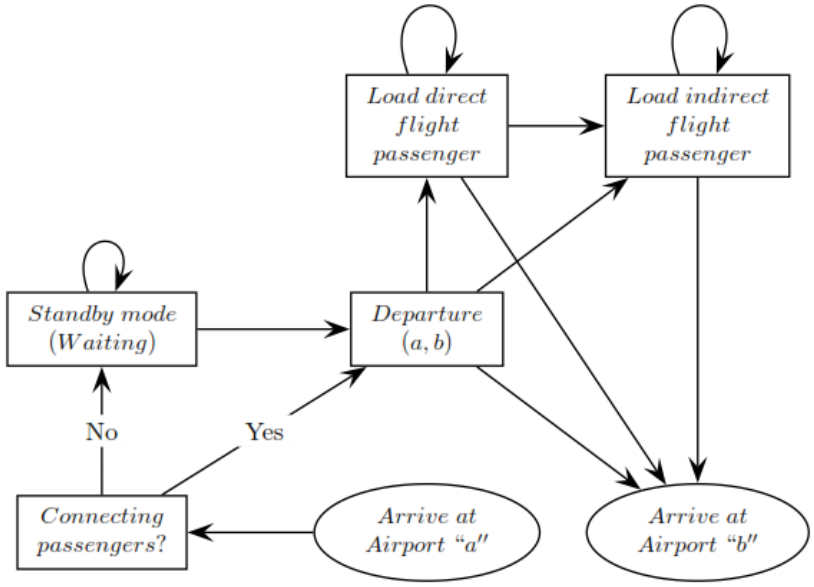

Figure 2.2: Key events from the perspective of the aircraft [14].

routing variable generation is the platform for the research conducted in this dissertation. In this section of his work Campbell used agent based simulation to reduce the number of variables in each model. The aircraft in his simulation were designed to find good time nodes as well as good flight legs to use in a MIP to find good solutions to the DAFP. Using this method Campbell found 13% deviation from the optimal solution and reduced the number of variables from 177 718, in a standard multi-commodity network flow (MCNF) ILP to 52 172 using the agent based method.

2.1.4 DAFP formulation

As stated by Campbell [7] and Espinoza [14] the dial-a-flight problem can be formulated as an integer linear program with the use of discreet time step increments. Here the time dimensions are split into time steps of 10 minutes. Nodes are located at all passenger departure and arrival times. Campbells formulation of the problem is stated as follows [7].

Decision variable definitions: Set of flight variables: $x_{ijf} \in X \ \forall i, j, f$ where each x_{ijf} corresponds to a specific flight leg. Each flight leg has an associated fleet $f \in F$ a starting node $i_{ut} \in N$ and ending node $j_{vt} \in N$. Where u and v are elements in the set of locations C and t is the respective arrival and departure times. These variables are integer.

Set of ground variables: Defined as $s_{ijf} \in S \ \forall i, j, f$, where f corresponds to the fleet type as an element of F. Where i_{ut} and j_{ut} correspond to the start and end node at the same location $u \in C$. These variables are referred to as ground arcs and are integer.

Set of passenger flight variables: Defined as $y_{xg} \in Y \ \forall x, g$ where x corresponds with a flight leg and g is a group of passengers within the set G. These variables are binary and indicate if a group of passengers are taking the flight or not.

Set of passenger ground variables T: Defined as $t_{ijg} \in T \ \forall i, j, g$ where g corresponds to the groups of passengers in G and connects i_{ut} to j_{ut} (where $u \in C$). This variable indicates whether a group of passengers is on the ground. They are referred to as passenger ground arcs.

The objective function given in Equation 2.7 is to minimise the operational costs:

$$min \sum_{x \in X} C_x x_{ijf} \tag{2.7}$$

Where C_x is the cost associated with the flight leg $x_{ijf} \ \forall i, j, f$.

Conservation of flow constraints:

These constraints ensure that fleet flow and passenger group flow at evey node are conserved and are defined in Equation 2.8 and Equation 2.9

For aircraft:

$$\sum_{i \in N} x_{inf} + s_{(n-1)nf} = \sum_{j \in N} x_{njf} + s_{n(n+1)f}, \forall n \in N, \forall f \in F \tag{2.8}$$

For passengers:

$$\sum_{i \in N} y_{x_{inf}g} + t_{(n-1)ng} = \sum_{j \in N} y_{x_{njf}g} + t_{n(n+1)g}, \forall n \in N, \forall g \in G \tag{2.9}$$

The passenger groups are bound by the capacity of the aircraft Equation 2.10.

$$\sum_{y \in Y} y_{xg} S_g \leq x W_x, \forall x \in X \tag{2.10}$$

where S_g is the number of people within group g and W_x is the capacity of the aircraft associated with the flight leg x.

There has to be a constraint set to ensure the number of aircraft of all fleet types are correct at the start of each day. Therefore at the starting node for each location $u \in C$, the ground arc s for each fleet type f must be set to equal the number of aircraft of that type at the starting location. This must also be done for the number of passenger groups at each location:

$$s_{n_i f} = v_{uf}, \forall f \in F, \forall u \in C \tag{2.11}$$

Where $s_{n_i f}$ is the ground arc for the fleet type f going into the first node of location u and v_{uf} is the number of aircraft of type f at location u at the start of the day.

$$t_{n_i g} = b_{ug}, \forall g \in G, \forall u \in C \tag{2.12}$$

where $t_{n_i g}$ is the passenger ground arc with node corresponding to the first node at location u and b_{ug} is binary, either 1 or 0 depending on whether u is the origin of g.

Constraints must be included to ensure the time windows of each of the passenger groups are not broken.

$$t_{n_e g} = 1, \forall g \in G \tag{2.13}$$

where $t_{n_e g}$ is the passenger group arc with end node corresponding to the expected departure time (EDT) and origin location of passenger group g.

$$t_{n_l g} = 1, \forall g \in G \tag{2.14}$$

where $t_{n_l g}$ is the passenger ground arc with start node corresponding with the late arrival time and destination of passenger group g.

It is important to note that this formulation by Campbell [7] allows passenger groups to swap aircraft during multi-leg journeys. The author states that preventing this

effect would require extra variables and would slow down the computational time of the model. This also means that this formulation would result in large waiting times for passengers. Therefore a constraint set was formulated to limit the amount of time a passenger could spend on the ground.

2.1.5 Parameters used to model the DAFR

The aircraft in the model need to make decisions about where to fly, which passengers to collected and what time to depart. These decisions are based on a set of criteria. These criteria are dependent on both internal factors as well as environmental factors.Campbell [7] makes mention of nine parameters that influence agents decisions in his research. The factors are as follows:

1. Consideration of the geographical distance.
2. A factor that prioritises on board groups for delivery before the late arrival time.
3. Consideration of the size of the group.
4. A factor that increases the attractiveness of groups on the ground going to the same destination as groups on board.
5. A factor that increases attractiveness of destinations which are the same as the destinations of groups on board.
6. A factor to increase the chance of leaving a destination after picking up a group of passengers.
7. A factor increasing the attractiveness of a destination based on the time urgency of the groups on board the aircraft.
8. A factor increasing the attractiveness of candidate groups by assessing the time closeness to their early departure times.
9. A factor that increases the attractiveness of waiting groups that share destinations with on board groups.

These factors represent some numeric value and can be combined to form a single number to represent the utility of the next move the agent makes. All parameters

concerned with the candidate groups are therefore the utility associated with the pickup function and all parameters concerned with candidate locations are associated with the relocate function.

2.2 Agent based simulation and modelling (ABSM)

Agent based simulation and modelling (ABSM) is considered to be a new paradigm shift in information technology. As mentioned in section 1.2 ABMS finds its roots in complex adaptive systems (CAS). The field of CAS asks the question of how complex behaviour arises. The mechanisms of CAS can be seen in Table 2.1. ABMS tends to be a descriptive method of simulation and therefore is suited to the analysis of the natural behaviour of a system rather than seeking the optimal behaviour [30].

Table 2.1: The mechanisms of complex adaptive systems [42].

Mechanisms of Complex Adaptive Systems
Aggregation
Non-Linearity
Transfer of information
Diversity (Allows different behavior)

The use of software agents to uncover patterns and gain insight into systems is growing. It provides a means to represent economic, social, ecological and other similar systems in a software environment. There is an abundance of research in this field but a lack of practical application, especially when using the approach in its operational capacity [42].

Agents are generally employed when the need for a distributed approach to problem solving is required. This is a problem that can be solved by a number of modules. They generally have some need for co-operation or evolutionary requirements. This makes ABSM a versatile technique [26].

The need for ABSM arises from various sources. Literature suggests that the suitability of the method comes from the world's movement to understand more complex systems. Macal and North [30] suggest that because of these increases in complexity traditional methods of simulation fall short. This could be due to the relaxation of assumptions

that could have been used to simplify more traditional simulation techniques. Lastly, the ABMS becomes a powerful tool when used to analyse big data sets [42].

2.3 Agents

Before a model can be developed, the definition of agents and their distinction from conventional simulation entities have to be defined. According to Salamon [42] the term agent has many definitions, the one that was most pertinent to this research was that 'An agent is a computer system that is suited in some environment, and that is capable of autonomous action in this environment in order to meet its designed objective'. This definition can only be used when defining artificial agents because, by the true definition of *agent*, they are not only observed and evaluated in a software environment. For instance any autonomous entity by definition can be considered an agent, such as robots and animals. For an agent to be considered an agent it must meet some basic criteria. These can be found in Table 2.2.

Table 2.2: The characteristics of agents [30].

Agents Characteristics
Attributes
Behavioural Rules
Memory
Resources
Decision Making Sophistication
Rules to Modify Behaviour

Wooldridge and Jennings [17] state that in order for a software component to be considered an agent it has to meet some criteria :

- Autonomy: Agents operate without direction or intervention from humans or others and have complete control of their actions.
- Social abilities: Agents must be able to interact with other agents or humans via some agent communication.
- Reactivity: Agents must be able to perceive their environment and respond to changes.

- Pro-activity: Agents should not simply act within their environment, but exhibit goal oriented behaviour.

A software agent is a component of the program they operate within. Agent based models can, therefore, be described in two components. The first being the agent itself and second the environment it falls into. The decision-making components of agents can be formulated using three major techniques, namely deliberate agency (that generally rely on the use of conditional statements such as 'if-statements'), utility based agents that make use of defined utility functions to differentiate an agents choices and lastly hybrids between the deliberate agents and utility based agents. Generally if an entity in a simulation has the ability to adapt its behaviour, communicate with other entities and perceive its environment, it can be considered to be an agent [30]. Figure 2.3 shows the general agent structure and the relationship agents have with their environments and other agents.

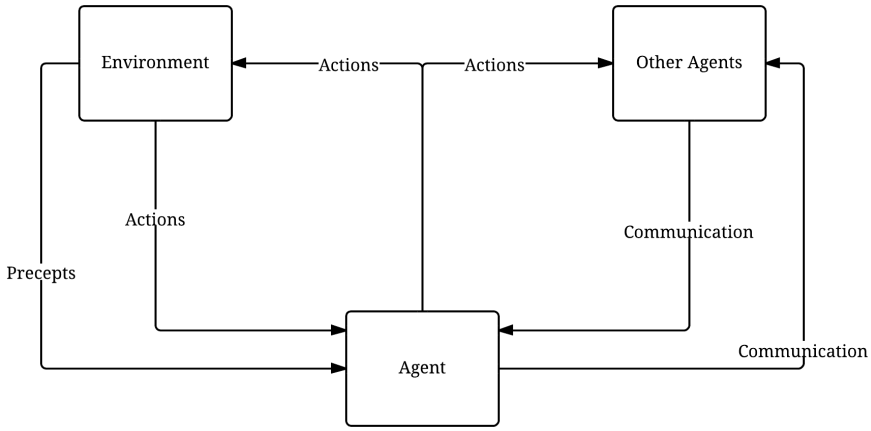

Figure 2.3: Generalised Agent Structure [42].

2.3.1 Deliberate agents

There are two important paradigms when describing the mechanisms of software agents. The first and simplest are deliberate or reactive agents. The idea of reactive agents was put forward by Rodney Brooks [42]. The reactive architecture consists of decentralized competence modules that are usually associated with the behaviours of the agents. These behaviours are then organised into the global structure of the agents. Reactive agents form part of a subsection of agents called deliberate agents. Deliberate agents

are the quintessential intelligent, rational agent. Deliberate agents are characterised by the factors below:

- they do not solely react to the environment they exist in,
- they peruse objectives,
- they co-operate,
- they have the capability to mimic behaviour, and
- they can model highly complex systems.

This general architecture can be seen in Figure 2.4. The functions of the general framework are defined below:

1. The *inputs* (see (1) in Figure 2.4) are fed into the system and are updated by the statement below:

   $$input: \quad E \rightarrow Inp$$

   Where E represents the environment and Inp represents the input.

2. The state of the agent (see (2) in Figure 2.4) updates by means of a *process* function.

   $$process: \quad Inp \times S \rightarrow S^{'}$$

   Where S represents the agents current state and $S^{'}$ which represents the agents future state.

3. Lastly the effector (see (3) in Figure 2.4) operates by means of an *action* function, this function performs the action the agent chooses (this process is dependent on the state S of the agent)

   $$action: \quad S \rightarrow Ac$$

   Where Ac is the action the agent takes and is dependent on the state S of the agent. The goals or objectives are built into this function.

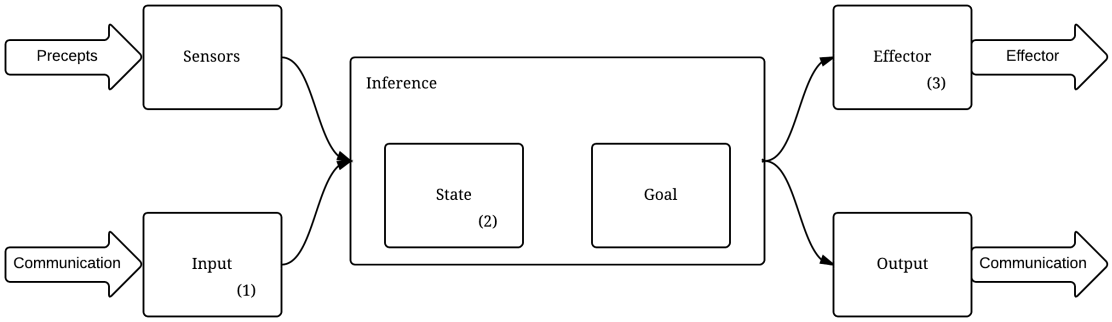

Figure 2.4: Framework for deliberate agents [42].

2.3.2 Utility based agents

Utility based agents form the second type of agent as described by Salamon [42]. Utility essentially is a measure of satisfaction. There are two categories for utility. *Cardinal utility* refers to an exact measurement of utility (such as money). The other form of utility is *ordinal utility*. This form of utility is implemented when the utility cannot be measured. Rather, it is a comparative method. Utility based agents are designed to maximise their respective cardinal utility. The utility of an agent is determined using a utility function. This can be built into the inference function of the agent. The utility function maps the possible states of the world and returns a real number corresponding to the utility of the move.

$$u: \quad O \rightarrow \Re$$

Where O is a set of real options $\Re$ that represent the final state of the agents. The agents will attempt to choose the states that provide the best utility. When the system provides perfect information, it is then possible to assume that the environment is accessible and deterministic. However, this is not always the case. The possible options represent the perception of the agents and in some cases agents will not always maximise their utilities.

The agents have an idea of what the probability of reaching a particular state given by $s_{t+1} \in O$ where s represents the state a agent will occupy given that it takes a action, represented by a. The probability can then be calculated using what is known as a utility function given as $T(s_t, a, s_{t+1})$. The sum of these probabilities over the set of available options are equal to 1 (see Equation 2.15).

$$\sum_{s_{t+1} \in O} T(s_t, a, s_{t+1}) = 1 \tag{2.15}$$

The expected utility of the move can then be defined as:

$$E[s_t, a, u] = \sum_{s_{t+1} \in O} T(s_t, a, s_{t+1}) u(s_{t+1}) \tag{2.16}$$

Where $u(s_{t+1})$ represents the utility of reaching state s_{t+1}. The agent will always seek to maximise its respective utility, (Equation 2.17).

$$\underset{a \in A}{Max} \quad (E[s_t, a, u]) \tag{2.17}$$

2.3.3 Hybrid agents

Hybrid agents are a combination of reactive agent and utility based agents as defined in subsection 2.3.1 and subsection 2.3.2 respectively. The hybrid agents use both sets of mechanics in order to achieve the agents sets of goals. These mechanisms are arranged in the program by means of layers. There are two predominant forms of layering, horizontal and vertical. Horizontal layering is when all software components are directly connected to the agents sensors and each layer acts as a sub-agent that gives the agent an idea of what to do next (Figure 2.5). Vertical layering allows the input information to pass through each layer. The interlayer exchange is much simpler, however layering the program in this fashion means that the simplicity is traded off for overall system performance. Because information passes through the entire system there is a greater chance for errors to increase. There are two main types of vertical layering namely:

- One-pass: the input information passes through the system one way (see Figure 2.6).
- Two-pass: the input information passes through the system both ways (see Figure 2.7).

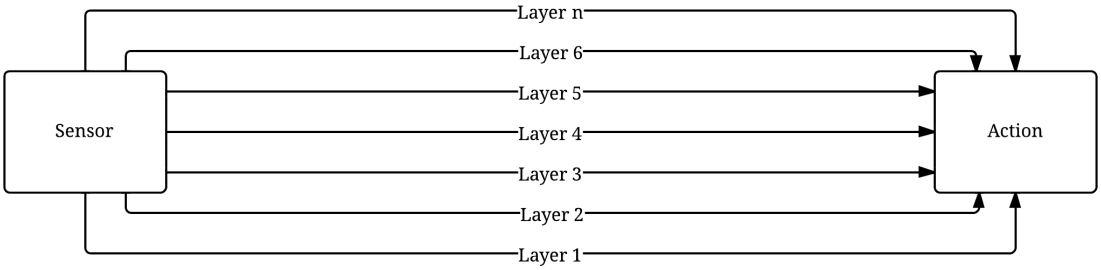

Figure 2.5: A schematic of horizontal layering [42].

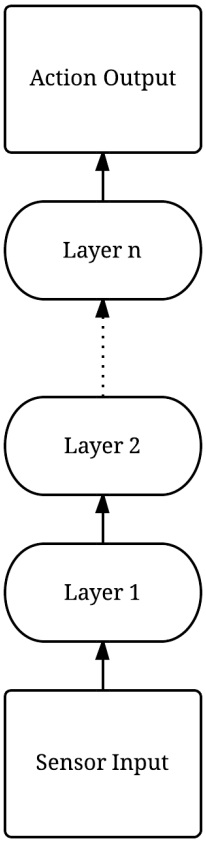

Figure 2.6: A schematic of vertical layering - One Pass [42].

The main advantage of using the concept of a hybrid agent is the ability to choose various agent architectures. This means that the model can be developed for a variety of applications. Literature suggests that when implementing hybrid agents there is no set methodology. Their design is always based on the application [42].

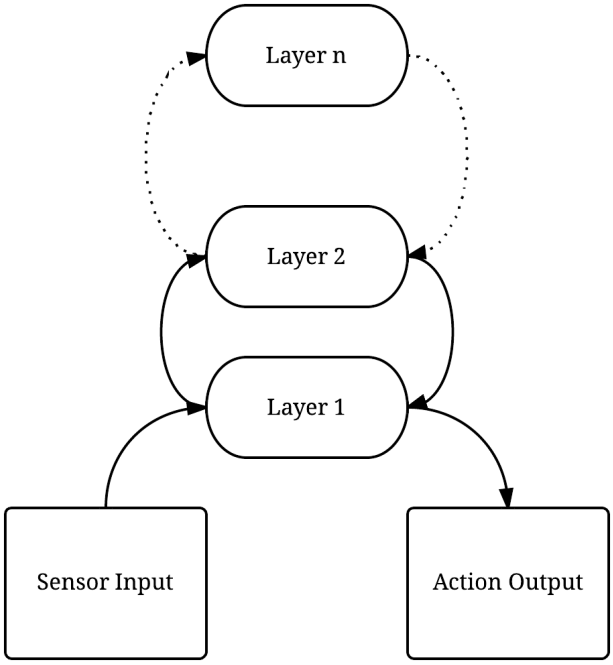

Figure 2.7: A schematic of vertical layering - Two Pass [42].

2.4 Agent environments

Before describing the decision making procedures of the agents, their environments have to be discussed in more depth. The basis of how these environments work and behave determines how the simulation will perform. There are six major dimensions to agent environments put forward by Salamon. They are as follows [42]:

- *Accessible vs. Inaccessible*
  - If the environment is accessible then the agent has the ability to gather information from the environment.
- *Deterministic vs. Non-Deterministic*
  - If the agents have defined paths and their actions lead to predictable results then the environment can be considered deterministic. If the model behaves stochastically then the model is non-deterministic.
- *Static vs. Dynamic*

  - If the agents are the only entities that affect the environment then the model can be considered static. If the environment the agents act in alters from one state to another then the model can be considered dynamic.

- *Discrete vs. Continuous*

  - Depends on whether the number of outcomes in the environment are finite or infinite. If there are a certain number of options the agent can act upon then the environment is discrete, otherwise it is continuous.

- *Episodic vs. Non-episodic*

  - If the model is episodic in nature then the agents only operate in segments of the environment. If the agents are able to fully access their surroundings then the model can be considered non-episodic.

- *Dimensional vs. Dimensionless*

  - If spatial factors are of importance to the agents then the system has dimension.

2.5 Application of ABSM

Since its conception, there have been many different applications of agent based technologies. Due to the autonomous nature of an agent [42], ABSM can be applied in a variety of fields. The most predominant field applicable to ABSM is traditional simulation. According to Wooldridge and Jennings [25] agent systems can be applied to solve problems that are beyond the capabilities of traditional methods. This is because agent simulations can be set up with fewer assumptions. This paper will focus on the applications of agent based systems as they apply to vehicle routing problems. However the literature also covers the application of ABSM in analysing system behaviour. In this research the expected emergent behaviour should be close to academic benchmarks.

2.5.1 Applications in operational behaviour

For the most part, the research that has been undertaken in the field of ABSM in operations research has been focused solely on the emergent behaviour seen in operations. Researchers focus on creating a realistic view of a current system to understand failures or inefficiency that arise due to the behaviour of the entities in the simulation. Once this has been uncovered they attempt to change the decision criteria of the agents (essentially designing a new system) to uncover a method of solving these problems.

In a study done by Knapen et al. [28], they undertook the task of discovering the incentives and inhibitors of carpooling using ABSM. The model tried to understand what incentives were required in order to encourage the use of carpooling. This was a classic example of how ABSM is traditionally implemented.

In another study done by Vanek [46], they sought to understand the behaviours of pirate ships in the Indian ocean and what strategies could be implemented by cargo ships in order to react to these threats. What can be seen in both of these cases was that the models uncovered the behaviours of the systems. In the carpooling case, they were trying to see how to encourage users to carpool and in the piracy case they tried to optimise the strategies implemented by the shipping companies.

The most pertinent study conducted, with respect to this research, was that by Archer et al. [1]. They conducted an analysis of an air taxi operation from a system of systems point of view, explored using ABSM. Their main focus was on aircraft selection and validated that the use of ABSM is suitable for industrial use and shows flexibility.

Grether et al. [19] developed an agent based simulation of air transport technology, the aim of the study was to evaluate the method's ability to forecast the behaviour of actors in the transport system. This speaks to the methods ability to study real systems. There have been other studies on collective transportation systems. Ciari [16] developed a multi agent traffic simulation framework to show how the method can be used to explore new solutions to urban traffic problems.

From the findings, different strategies can be tested and the optimal behaviours could be uncovered. This is different from finding optimal routes in a VRP, but the structures of the models and the approaches taken are of value because when a model is designed

there may be emergent optimal behaviour and this is important to understand when designing for optimality.

2.5.2 Applications in optimisation

In the field of vehicle routing the use of ant colony optimisation (ACO) has been abundant. This particular algorithm has been popular in solving problems sunch as the traditional travelling salesman problem (see Netlogo representation of this problem in Figure 2.8 [48]) and other VRPs. This algorithm can be considered a form of agent based simulation since the ants (effectively agents) in the system behave autonomously, but lack the full complement of ABSM characteristics such as memory. As with the test bed developed by Certicky et al. [47], the models developed are aimed at utilising ABSM with some algorithm built into the agent's decision engine [47]. Reed [39] illustrated the use of ACO in multi-compartment vehicle routing, essentially solving a capacitative vehicle routing problem. He found that this application of ACO was suited to the application. Thus, ACO in conjunction with ABSM could prove to be suitable for solving problems such as the DAFP and DARP.

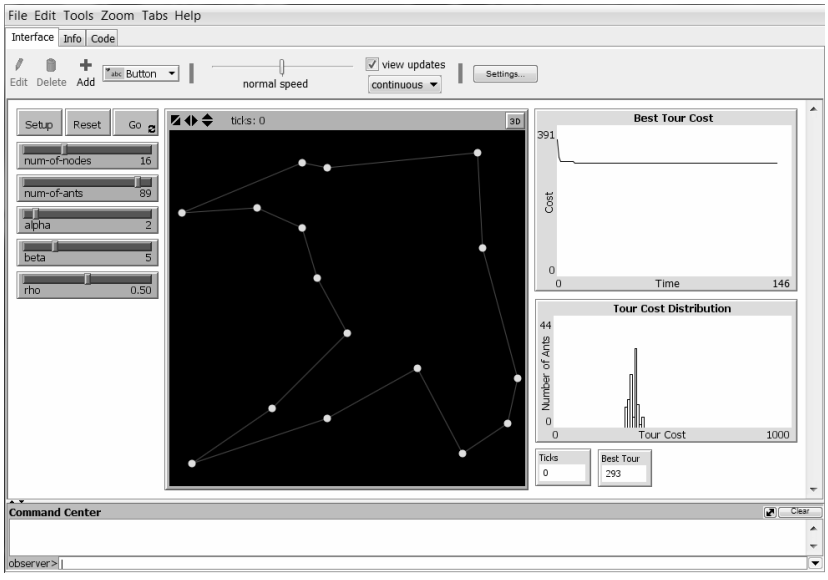

Figure 2.8: Netlogo representation of the ant colony optimisation by Jose Vidal [48].

Shah [44] gives an overview of the different approaches to multi-agent systems in the vehicle routing space. There have been three predominant architectures that have been used, MARS, JABAT and the ant system. The main reason for the use of agent systems in this field is because multi-agent systems behave autonomously and, therefore, will

provide better approaches to dynamic versions of the problems, but the most attractive feature of the method is that it flexible. Flexibility allows users to change aspects of the problems with ease.

The MARS system was developed for a shipping company called the Modelling Autonomous CoopeRating Shipping Companies (MARS). It was used predominantly for planning and scheduling dynamically. JABAT is middle-ware using multi-agents to solve VRPs. It is capable of solving combinatorial problems using a set of intelligent agents, each representing a set of improvement algorithms. The ant system developed by Dorigo [12], derived from the natural foraging behaviours of ants, uses swarm intelligence to solve VRPs see section 2.8.

Thangiah et al. [45] state that the use of ABSM to solve VRPs is a result of the need for a decentralised construction heuristic. They also states that there is a need for flexible models that allow for easy integration of traditional techniques. Dexterity of using the algorithms is also important so that they can solve many variations of the problem with minimal changes to the algorithm, thus making it a viable alternative to the Clark-Write method.

This report seeks to use traditional ABSM to solve VRP problems rather than implementing proto-agents to do so. Campbell [7] provides more depth into the use of ABSM to find feasible solutions to the DAFP. The method put forward resembles that of ACO, but lacks the pheromone update used for learning. The model relies on the fact that the dial-a-flight problem presented is highly constrained. Therefore, simple Monte Carlo techniques coupled with effective utility functions should provide good solutions. By applying an optimising linear program on the good nodes and legs of the route, optimal solutions to the problem are generated. The finding suggest the method is flexible and provides good quality solution for tightly constrained problems. In this report a utility function will be used to control the decision making of the agents. Campbell [7] proposed the use of an exponential distribution (see Equation 2.18) to determine the attractiveness of the next move.

$$u(s+1) = 1/e^{d/F_1} \tag{2.18}$$

Where d is the factor that influences decisions (such as distance or time urgency) and F_1 represents the weights that determine the importance of the factor. He also suggested

the exploration of linear and Gaussian function, when determining the utility of the next move.

Another approach taken by Ziddi et al.[53], was the use of a multi-agent system based on a multi-objective simulated annealing model to solve the DARP. The results were competitive, considering the approach taken was new. There were also considerable improvements in process times. The model created was versatile in that it could be adapted to solve other multi-vehicle routing problems.

The use of ABSM without the use of an assisting heuristic has also proven positive. Barbucha [3] proposed an agent based model to solve the VRP and concluded it was a viable method but produced inferior results to that of the tabu search algorithm. In this work Barbucha et al. [3] make use of the A-Team structure, which is essentially a collection of software agents that co-operate to solve a problem. The proposed algorithm has the following structure:

1. Generate the initial population of solutions,
2. apply a solution improvement algorithm from the initial population,
3. replace the selected solution with the improved solution, and
4. continue the cycle of reading - improving - replacing until some stopping criterion has been met.

The use of an improvement agent means that this method is different from that used in Campbells' research [7]. It resembles that of a traditional heuristic technique, and provides a more decentralised approach to solve the problem. The method does prove to be effective showing deviations from the optimal of around 5% [3].

In a study done by Vokrinek et al., ABSM was applied in solving the capacitative vehicle routing problem CVRP [49]. The results were optimistic, with an average of 19% error on 115 benchmarked problems. The architecture used for this problem was similar to that of Barbucha et al. [3] in that it makes use of improvement agents that had control of the population. Vokrinek et al. [49] suggests the use of three main agents types that make up the architecture of the program:

1. Task agent: process the demands of passengers and initiate the allocation of a customer.

2. Allocation agent: maintains the allocation of tasks to vehicles, and improves the overall solutions.

3. Vehicle agent: represents a single vehicle and is responsible for planing and optimising the routes.

The generalised architecture for this model can be seen in Figure 2.9.

The method allows for different strategies to be tried and tested and it successfully minimised the number of vehicles routed. This method was shown to be able to solve the capacitative vehicle routing problem with time windows (CVRPTW) by Kalina et al. [27]. The method proposed involved not only improvement agents but also negotiation strategies, and proved successful in achieving an average error of 3.2%. The negotiation takes place between the vehicle agent and the allocation agent and is broken up into four main processes; estimation of the cost of committing to a given task, estimating the gain from dropping some commitment, identifying the most costly task and committing or abstaining from a given task.

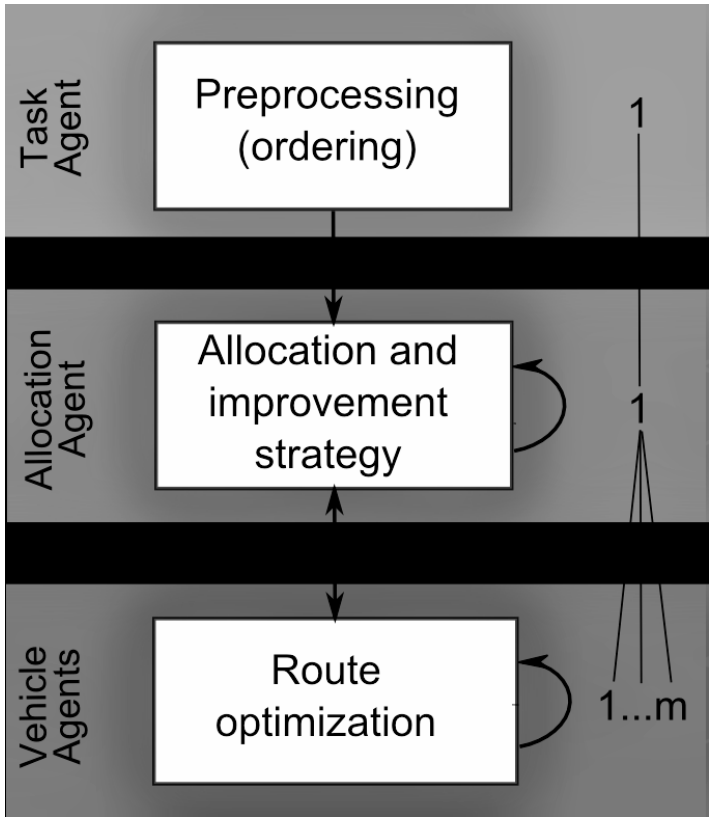

Figure 2.9: CVRP ABS program structure by Vokrinel et al. [49].

There have also been instances of using similar architectures for solving the dynamic versions of the CVRP. Maciejewski et al. [31] used the MatSim platform to simulate the dynamic vehicle routing problem (where passenger demand varies throughout the simulation) with the use of a multi-agent system. It is worth noting that the algorithm used incorporated evolutionary algorithms to aid the simulations. The approach taken takes factors that contribute to the search for optimality such as traffic in urban areas.

The real question that has to be asked is whether or not ABSM can compare favourably with the conventional techniques. There have been a few comparative studies, such as the comparison between ABSM and on-line optimisation for a drayage problem with uncertainty by Mahr et al. [32] The key finding was that the ABSM outperforms the conventional technique when uncertainty is high. This is probably due to the reactive behaviour of the agent, being able to react directly to changes in the problem. The study also suggests that ABSM can be very effective when applied to highly constrained problems.

What is noticeable from the literature was that the ABSM are suitable in solving constrained problems, or problems that contain some stochastic properties. However, there does not seem to be much literature on the use of the method to solve the DAFP other than the method proposed by Campbell [7]. There are new movements in ABSM. Researchers are now focused on the cooperative and learning aspects of ABSM [27], [2].

2.6 Current models and software

There are many open source software packages available to model systems using ABSM. NetLogo [37] is a robust package that allows the user to create an agent based simulation with ease however the reporting capabilities of the package are not up to the quality of conventional software packages. Designing complex custom models is tedious. Other programs have been developed to exploit this new field of research such as AnyLogic and SWARM.

Not many platforms have been developed to use the ABSM to solve VRP's. In 2014 Certicky et al. [47] developed a test bed for the analysis of the DARP using ABSM. Essentially the test bed allows the users to evaluate the performance of multi-agent on demand transport schemes for both static and dynamic cases. This is useful since the

test bed works in conjunction with Google Earth and can provide real test scenarios for the modellers. This design makes the implementation of agent based models easier for real world applications.

The majority of the models developed towards ABSM in solving VRPs have used standard programming languages such as Java, C++ or Matlab. These programs allow the users to fully customise and design the structure of the models as well as allow them to incorporate other heuristics methods into their simulations.

2.7 Agentology

The approach used in the development of the ABSM in this report is based on the generalised methodology put forward by Tomas Salamon [42]. Salamon suggests that due to the rise in demand for agent based models, a formalised method is required to develop these simulations. This methodology is called agentology. Agentology constitutes the steps in the development of such models. It is important to note that agentology has been predominantly developed for use in social sciences and applying this method to conventional operations research has rarely been done. This method was used to develop models to solve variations of the DAFP defined in subsection 3.1.1.

Salamon describes the development of agent models in a nine step process that can be grouped into four phases. These are defined as (also see Figure 2.10):

1. Task formulations: In this case before the model can be developed the DAFP must be formulated. The identification of the main objectives of the problem, the processes involved with the system and the identification of the agents are done at this phase.

2. Task evaluations: This can be seen as a critical stage in the development of the model. The understanding of the task has to be continuously assessed in order to develop a suitable model. Here the agents physiology is uncovered. In the case of the dial-a-flight problem the agents are hybrid, taking advantage of both reactive and utility based decision tools.

3. Conceptual modelling: This involves transforming the task into a conceptual model that consists of a set of understandable diagrams.

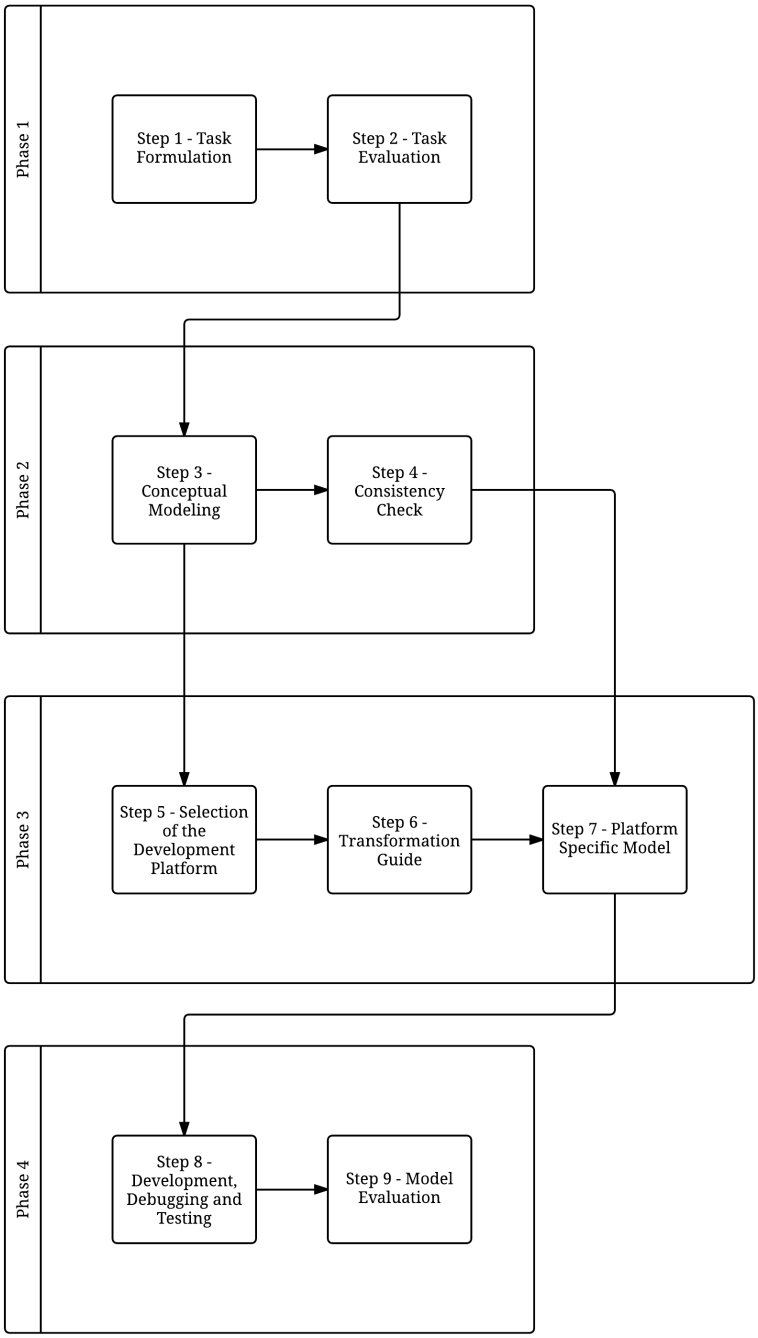

Figure 2.10: Agent model development method proposed by Salamon [42].

4. Consistency check: This is considered to be a recurring step since at every stage in the model development from this point the consistency of the model has to be evaluated, so that the accuracy of the results improve as the model is developed.

5. Selection of a development platform: In the case of this work Matlab was chosen as the development platform. It has extensive programming capabilities and heuristics can easily be incorporated.

6. Transformation: Transformation of the conceptual model to a working program.

7. Platform-specific model: For this report this step is irrelevant and only applies to models that are platform dependent. Because the conceptual model of the DAFP has been transformed directly into code the need to develop the platform is non-existent.

8. Development: This involves debugging and testing the models.

9. Model Evaluation: Here the results of the simulation will be evaluated against benchmarks to the problem.

2.8 Ant colony optimisation (ACO)

Ant colony optimisation (ACO) can be considered as one of the most successful vehicle routing algorithms. It is fast, accurate and can handle many different types of problems [5]. It is based on the self-organising principles of an ant colony. This allows for highly coordinated behaviour to occur, essentially using co-operative agents to solve complex problems. The main focus of research into ACO has been on foraging behaviour, the division of labour, brood sorting and co-operative transport of ants. These algorithms, like most algorithms for VRPs, deal with discreet problems [12].

ACO has the ability to solve travelling salesmen problems (TSP) quickly. It does this by mean of co-operative agents, that work in unison to achieve the target objectives. ACO is limited by the types of problems it can solve. This is what is relevant in terms of adapting this thinking towards agent based simulations. This is because the DAFP makes use of a heterogeneous fleet and thus one generic ant cannot be formulated. Thus an intelligent ant without the swarm intelligent component would be better suited to these types of problems.

The mechanics behind the ants co-operative behaviour is very simple (Figure 2.11) [11]:

1. Ants are sent out of the nest to search for food. They will choose routes at random. each ant lays down pheromone. This pheromone lets all the other ants know where it has been. The pheromone will evaporate after some period of time.

2. Ants populate the search space, and will most likely choose paths that have stronger pheromone trails. This is because as more ants use a specific path, the pheromone levels on that path will increase and attract more ants. The paths that are less frequently used will have less pheromone, because less ants will chose to use the path and the excess pheromone will eventually evaporate.

3. As the pheromone levels on the bad trails drop, due to less frequent visits by the ants, an optimal route through the environment will form.

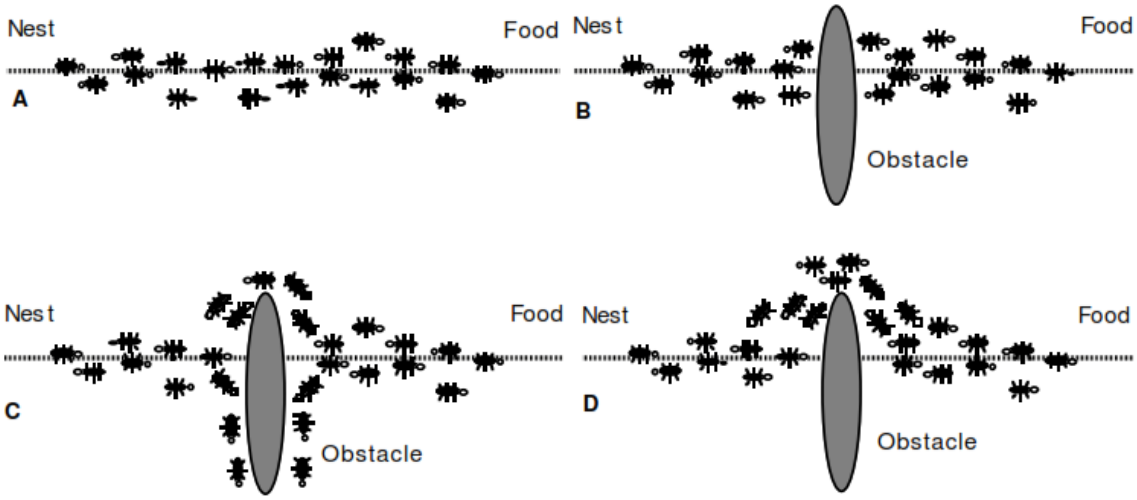

Figure 2.11: Co-operative behaviour in ACO [11].

The application of ant colony optimisation to the travelling salesman problem by Dorigo [11] in 1997 has opened the door for its use in more complex VRPs. Dorigo found that the algorithm is flexible and produced good solution quality when applied to symmetric problems [11]. Since then the algorithm has been adapted to solve other versions of the VRP, such as the multi-compartment vehicle routing problem [39], the vehicle allocation problem [36], the general vehicle routing problem [5] and the vehicle routing problem with time windows [12].

2.9 Genetic Algorithms (GA)

Genetic algorithms (GA) are a branch of heuristics that involve using an evolutionary approach to solve combinatorial problems. These problems generally require a method for searching through large amounts of solutions until an optimal or near optimal is found. The adaptive nature of GA means that it can be used to quickly find solutions without resorting to brute force [35] and thus provide an alternative method in solving large computational problems. The thinking behind this method comes from the process of natural selection, where, as generations of a population progress through time the genetic characteristics of the offspring resemble the most successful characteristics of the parents [21] (see Figure 2.12).

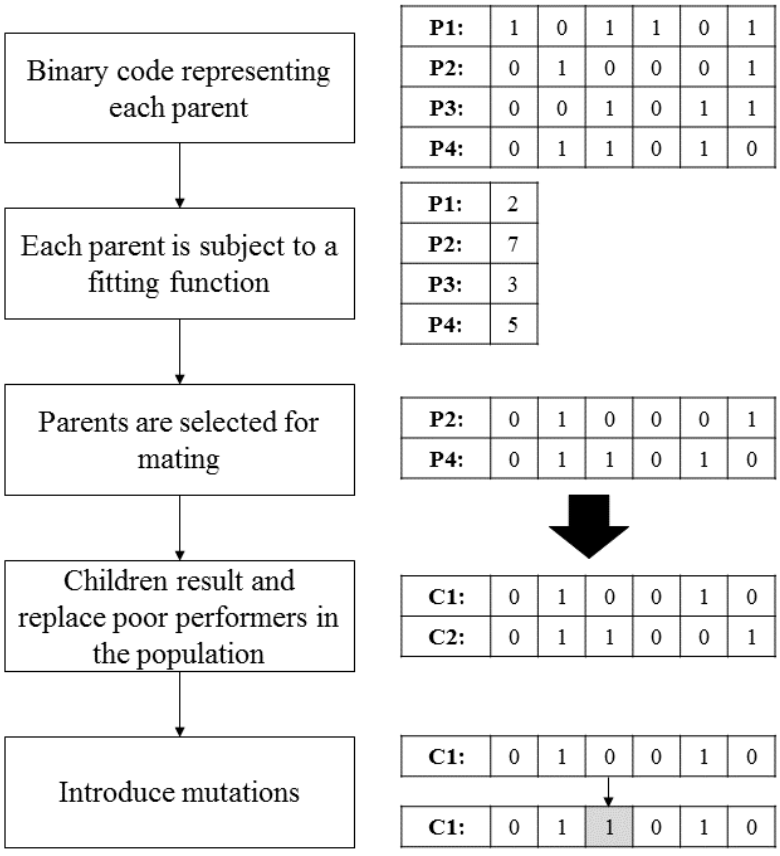

Figure 2.12: Analogy between a numerical genetic algorithm and biological genetics [21].

The genetic algorithm can be broken down into six steps:

1. Generation of the initial population
2. Find cost for each chromosome against some fitting function

3. Select mates
4. Mating of parent chromosomes
5. Mutation of children chromosomes
6. Convergence check or termination

The chromosome is a string of values that contain the decision variables for the function the user wishes to optimise. The model should start with an initial population of chromosomes. The population size should be carefully considered since a small population stands the risk of not effectively covering the search space and a large starting population may be a computational burden [40]. From the starting population, the best fit chromosomes are selected as parents and are mated using a crossover method. Each gene within the chromosome is subject to some mutation, defined by some mutation rate which states the probability of any gene being changed. Next is a convergence test or some stopping criteria. Here the solutions from the resulting children are checked against previous results. If the number of iterations has exceeded some predefined number, the algorithm is to be stopped, otherwise the process is restarted [23].

According to Reeves [40], there are five main advantages of using a genetic algorithm:

- Generality: this method can be formulated in a variety of programming languages and can be designed to solve a wide variety of problems.
- Non-linearity: the method does not rely on any assumptions of linearity, convexity or differentiability. All that is required is some method of calculating the performance which can be complex and non-linear.
- Robust: the models have been able to solve large computational problems. They can be applied to a wide variety of problems without major changes to the formulation of the algorithm.
- Ease of modification: GA's are easy to modify to solve any variations in the selected problem.
- Parallel nature: due to the repetitive nature of the model, this means that the selection and mating processes can be solved in parallel, increasing the speed of the algorithm.

2.10 Monte Carlo methods

Monte Carlo is a type of simulation that relies on selecting unknown outcomes using random sampling. The method resembles random experimentation where the outcomes will be unknown but can be represented by some distribution. The input data is modelled using some predefined distribution. The outcomes are then sampled from that distribution. Within the field of simulation the outcomes represent some step forward in the simulation, in the case of vehicle routing this could mean a choice to move to some location, drop off a passenger or pick up a passenger. The method is simple and provides an autonomous means of decision making [33].

The Monte Carlo method provides an easy way to sample from a probability distribution and can be used to design and test rudimentary simulations. Its ease of use makes it a good candidate for decision making. In ACO Monte Carlo sampling is utilised to assist ants in making choices in the graph. The pheromone update is used to update the probability distribution and Monte Carlo is used to sample this distribution [12]. This is technique can also be see in research conducted by Campbell [7], it is used to route aircraft through the day as well as assist the aircraft to choose passengers to pick up. This method has also been applied to local branching algorithm for the single vehicle routing problem with stochastic demand [41].

This method proves to be a suitable candidate to rout vehicles through a cost network.

2.11 Multi-criteria decision analysis (MCDA)

Multi-criteria decision analysis (MCDA) is a tool used by operations researchers to analyse different factors that contribute to making a decision. The method is highly dependent on the the context in which it is applied as well as the expertise of the people involved in designing the models. The method's popularity stems from the need to take into account multiple dimensions to a problem and consider each of the factors as a whole when making a final decision. The main steps of the multi-criteria decision making as defined by Ramon and Mateo [34] are as follows:

1. Defining the problem: Here the context of the problem should be uncovered. Clearly defining the parameters and the expectations of the stakeholders. Alternatives, objectives, constraints and any points of conflict should be examined.

2. Assigning weights: For each factor that influences a decision a weight should be assigned that represents the respective importance of the factor.

3. Construction of the evaluation matrix: The MCDA model can be expressed in matrix form:

$$\begin{aligned} &\text{Criteria:} && [C_1, C_2, \cdots, C_n] \\ &\text{Weights:} && [W_1, W_2, \cdots, W_n] \end{aligned}$$

$$\text{Alternatives:} \quad \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{pmatrix} \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$

   where x_{ij} is the numeric value given to the the ith alternative with respect to the jth criterion. w_j is the weight of criteria j, n is the number of criteria and m is the number of alternatives.

4. Selection: A method of selecting the alternatives must be chosen. The data and the level of uncertainty should be taken into consideration when deciding the method.

5. Ranking: The alternatives are ranked and the best one is proposed as the solution to the problem.

There are several advantages to using the MCDA method. It allows for investigation into multiple fields, it is capable of dealing with highly complex problems and is well known in the field of operations research [34]. Its most attractive feature is its ability to compare factors that have different units and it can deal with both qualitative and quantitative values [52]. It has been applied in manufacturing, environmental science, agriculture and medicine [22],[20],[29],[34].

3 Methodology

In chapter 2 the literature pertaining to multi-vehicle routing problems, ABSM and heuristic techniques were discussed. In this chapter the DAFP is defined, benchmarks to the instances tested are provided and the design of the ABSM model is presented.

3.1 The Dial-a-flight problem

This dissertation uses sets of instances (S10, S39, S40, S99, S102, S139 and S200) of the DAFP provided by Wilderness Air, these instances represent booking lists require scheduling. The data sets listed in the literature from Espinoza [15], [14] only provided aggregated results and so were not included or tested in this paper.

3.1.1 Wilderness air problem description

The version of the dial-a-flight problem in this work was defined using real data from an airline operating in the Okavango Delta shuttling tourists between 30 of 100 destinations. The airline is Wilderness Air services, the destination layout can be seen in Figure 3.1 [7]. Seven booking lists (see Appendix C) were supplied by Wilderness Air. Each booking list contains requests from groups of people. Each booking list is comprised of five dimensions. The earliest departure time and latest departure time are given in minutes and are essentially the time window that the aircraft must pick up and drop-off a passenger. The origin and destination gives the city reference number for which the passengers are requesting travel and people in the group states the number of passengers in the group. A booking list with 10 requests can be seen in Table 3.1. From this schedule one can see that there may be similar requests from

multiple customers. To reduce the amount of variables, an agent could combine the requests from group number two and seven to form one request that has the same time window and three passengers within the group. Doing this reduces the problem size making it easier for the program to find solutions, however optimality could be compromised because it eliminates possible solutions from the search space.

Table 3.1: An example of a 10 Request booking list.

Early departure time (min.)	Late arrival time (min.)	Origin	Dest.	No. in group
360	1080	6	8	1
360	1080	7	5	1
360	1080	4	7	1
360	1080	8	5	1
360	1080	7	8	1
360	1080	4	5	1
360	1080	7	5	2
660	930	5	8	2
660	900	5	6	5
815	870	21	20	11

The aircraft are currently manually scheduled to meet these requests and to minimise the overall cost of operations, as well as maximise customer satisfaction. These booking lists were supplied by Wilderness Air and are subsequently referred to as 'manual booking lists'. Customer satisfaction is defined as the aircraft being available for take off within the desired departure windows, minimise flight times and minimal number of flight legs per journey.

There are two types of aircraft that are used in order to service the requests, the Cessna 206 and Cessna 208. The specifications of the aircraft are given in Table 3.2. Each aircraft has a specified starting location, so regardless of the route the starting point of the routes must abide by this condition. An example of the fleet can be seen in Table 3.3. There are five type 1 aircraft and nine type 2 aircraft in the fleet.

Table 3.2: Set of aircraft specification.

Aircraft	**Fleet type**	**Speed (km/hr)**	**Seating capacity**	**Cost (Rands /hr)**
C208	1	260	11	R3910
C206	2	210	5	R1385

Table 3.3: Fleet used for booking lists S10, S40 and S102.

Plane No.	Type	Starting Location	Ending Location
1	1	7	7
2	1	7	7
3	1	7	7
4	1	4	17
5	1	18	3
6	2	3	7
7	2	7	7
8	2	7	7
9	2	7	7
10	2	7	17
11	2	17	3
12	2	3	3
13	2	8	12
14	2	19	16

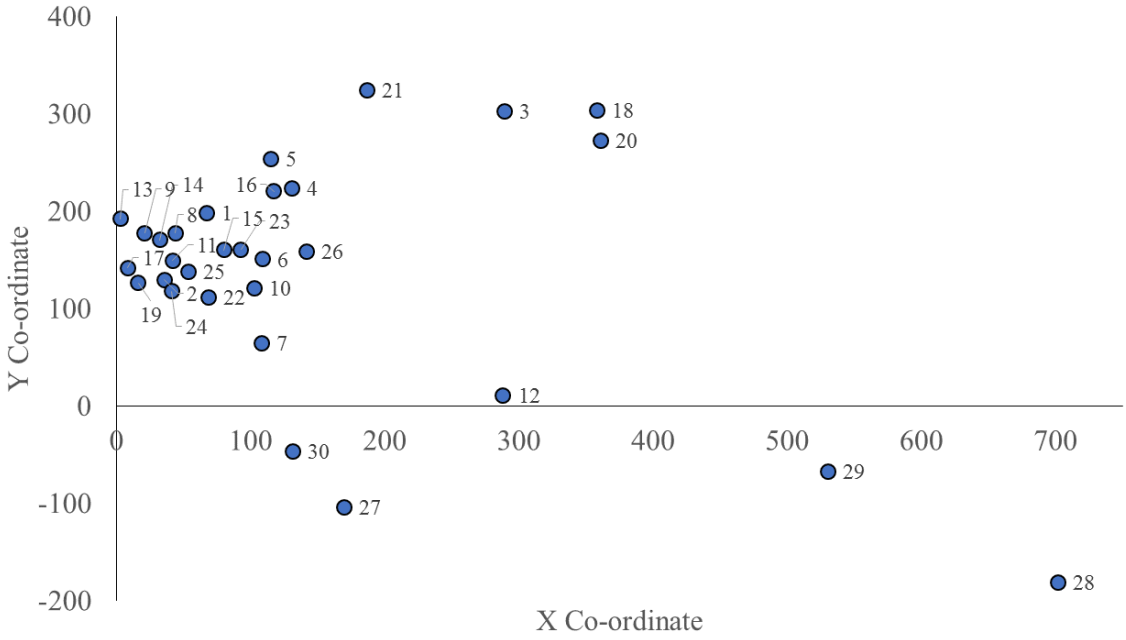

Figure 3.1: Destination layout of the cities visited by the aircraft in the DAFP.

The aircraft are then expected to navigate to origin nodes to pick up customers and deliver them to the destination nodes. Each node represents a destination in the

area and each has an x and a y coordinate that is located in Euclidean space. The parameters of the problem are as follows:

1. Turnaround time of the aircraft is 10 minutes. This value was chosen since it provides enough time for the aircraft to drop off its current passengers and fetch new passengers, and is the value currently used by Wilderness Air.
2. Turnaround times do not contribute to the overall cost of operations. The only cost that will be considered are costs incurred by aircraft in flight.
3. Aircraft fly at a constant speed throughout the flight leg. Take off, landing, boarding and disembarking are assumed to be included in the turnaround time.
4. The pick up convention was assumed to be that the aircraft would have to be able to service a full group of travellers at a time. This means that no group splitting can occur.

Two version of this problem are solved in this research. The differentiating factor are what constraints are applied to the aircraft.

Problem 1 constraints:

- Vehicles can only carry passengers equal to or less than their maximum capacity.
- Passengers can only be picked up and dropped off during their available time windows.
- No swapping of aircraft may occur. This is an inconvenience to the passenger and thus reduces customer satisfaction.
- All demand must be satisfied.

Problem 2 constraints:

- Vehicles can only carry passengers equal to or less than their maximum capacity.
- Passengers can only be picked up and dropped off during their available time windows.

- No swapping of aircraft may occur. This is an inconvenience to the passenger and thus reduces customer satisfaction.
- All demand must be satisfied.
- Passengers request to not stay on board the aircraft for longer than 3 flight legs.

In order to approach the problem, the network was visualised for both problems, including how the agents (in this case the aircraft) navigate the system. Both Figure 3.2 and Figure 3.3 show two aircraft with starting nodes at 1 and 8 and destination nodes at 5 and 9 respectively. Each aircraft can only carry their maximum capacities, and, therefore, the aircraft will have to pick up and drop off according to how many passengers are available. Pickups given by the positive letters represents the groups of people that require pickup and the negative letter represents the required drop off. Unlike in the dial-a-ride problem, nodes act as both pickup and drop off points, and aircraft do not have a centralised depot. Each aircraft has a local starting position, and may have an ending position too. In this illustration of the problem, each aircraft has predetermined starting locations at 1 and 8 respectively and no required end of day destination. Figure 3.2 shows passenger a able to stay on board the aircraft for more than three flight legs and Figure 3.3 shows that passengers a and b are only allowed to stay on board the aircraft for three flight legs.

Two models were developed, each with unique design to solve these problems respectively. Model A was designed to solve problem 1 and Model B to solve problem 2.

3.1.2 Benchmarking and manual schedules

There were seven booking lists supplied from Wilderness Air. They range in size from 10 to 200 requests. In order to measure the performance of the model, final costs are compared to given benchmarks. Some manual schedules were also supplied by Wilderness Air. For booking lists sized between 10 and 99 the results of the ABSM model were compared to the upper bound solutions from a mixed integer programme. The cost function used to calculate these costs are given in Equation 2.7 in subsection 2.1.4, the actual costs per aircraft are derived by multiplying the flight time of an aircraft by the cost per hour given in Table 3.2.

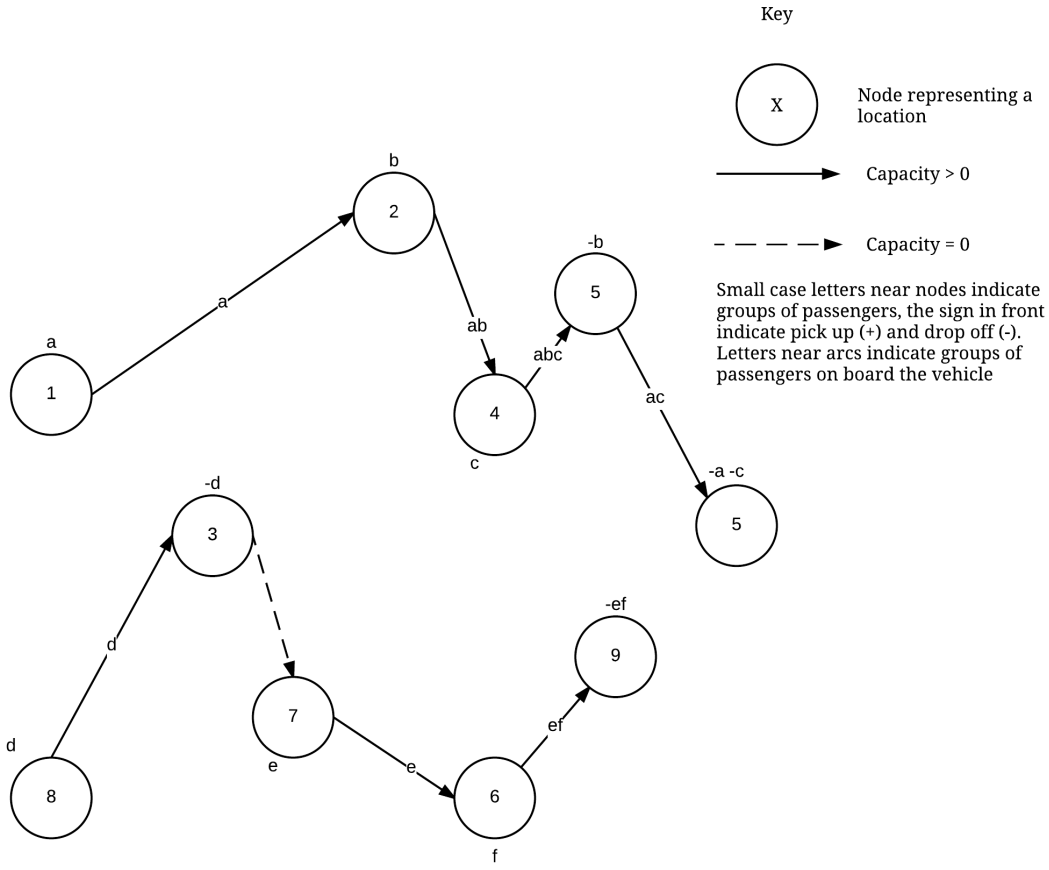

Figure 3.2: Problem 1 visualisation of a DAFP with two aircraft.

Booking lists 102 to 200 were manually generated solutions by Wilderness air. The benchmarks can be found in Table 3.4. Both problem 1 and 2 solutions were compared to these given benchmarks [8], [51]. This dissertation uses sets of instances (S10, S39, S40, S99, S102, S139 and S200) of the DAFP provided by Wilderness Air, these instances represent booking lists that require scheduling.

3.2 ABSM design

The agents in the DAFP will be referred to as aircraft. The aircaft required by both problems need to be able to decide the routes that will minimise their operational costs. They will seek to minimise the overall distance travelled and maximise customer satisfaction. This means that there will have to be a utility, which is the value of attractiveness, associated with each move. However aircraft can only make decisions

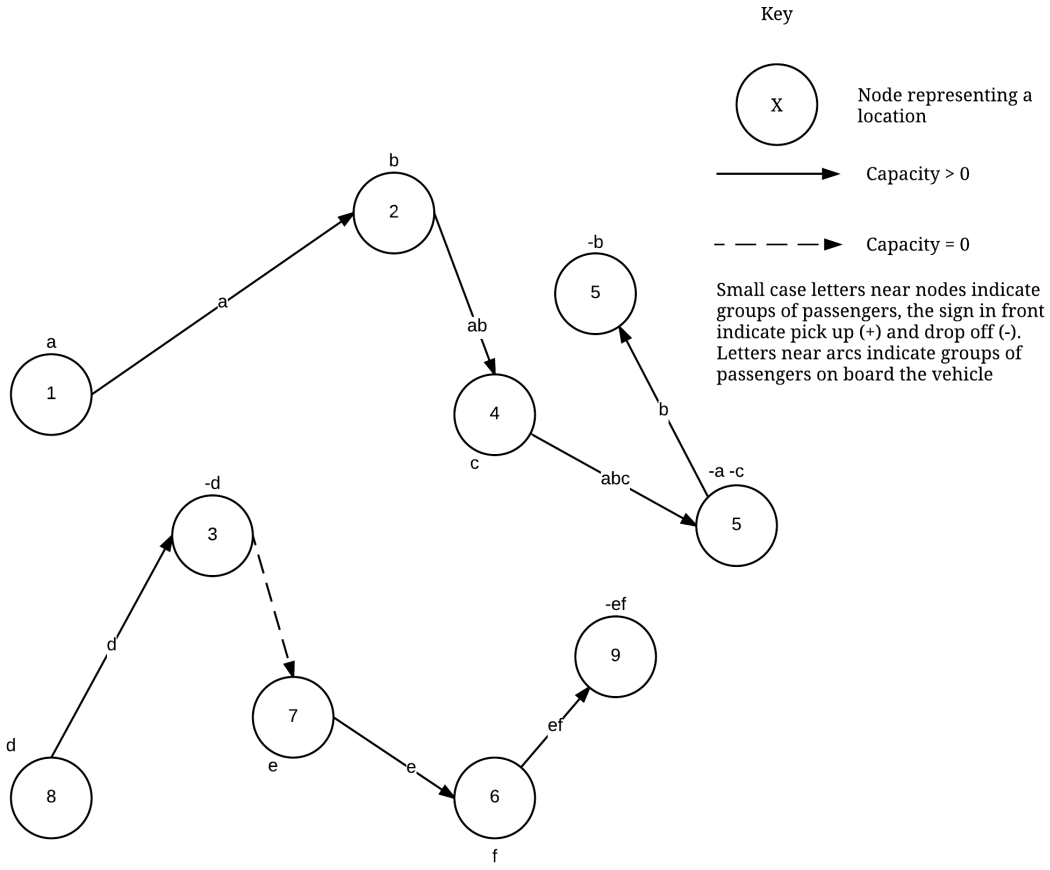

Figure 3.3: Problem 2 visualisation of a DAFP with two aircraft.

Table 3.4: The benchmarks to the DAFP.

	Lower Bound	**Upper Bound**	**Manual**
S10	-	R 12 826	-
S39	R 17 343	R 20 856	-
S40	R 25 035	R 34 531	-
S99	R 50 407	R 96 395	-
S102	-	-	R 63 270
S139	-	-	R 49 346
S200	-	-	R 81 740

based on the current information, such as their capacity and current demand. It was because of these two considerations that the model developed utilise the hybrid agent. Agents share characteristics of both reactive agents and of utility based agents. The

agent characteristics are given below in Table 3.5. The only characteristic that these aircraft do not possess the ability to modify behavioural characteristics.

Table 3.5: The DAFP aircraft characteristics.

Agent Characteristics
Attributes
Behavioral Rules
Memory
Resources
Decision Making Sophistication

3.2.1 Agent perception and action

The aircraft interact with the environment by means of three main functions:

- The drop off function, that allows passengers to disembark and records late arrivals and waiting times.
- The pickup function, that assesses all the available passengers at the aircraft's current location and decides which passengers should board the plane.
- Lastly the relocate function which decides where the aircraft should fly to next.

These functions have all the reactive rules built within them and are expected to produce emergent behaviour that produces good solutions without violating the problems constraints. These three functions are at the heart of the agent. After each aircraft has commenced, it will have produced one full flight leg, where drop offs and pick ups occur at the current location of the aircraft. The flight legs are defined by the relocation of an aircraft from one location to the next. A flight leg can be described using a time space diagram (see Figure 3.4). This figure shows the two states an aircraft can occupy; in flight, and idle. It also illustrates the ten minute turnaround time allocated to account for passenger boarding and disembarking. The dotted line shows how the turnaround times are included in the flight time. This becomes useful when checking if any given location is a viable option.

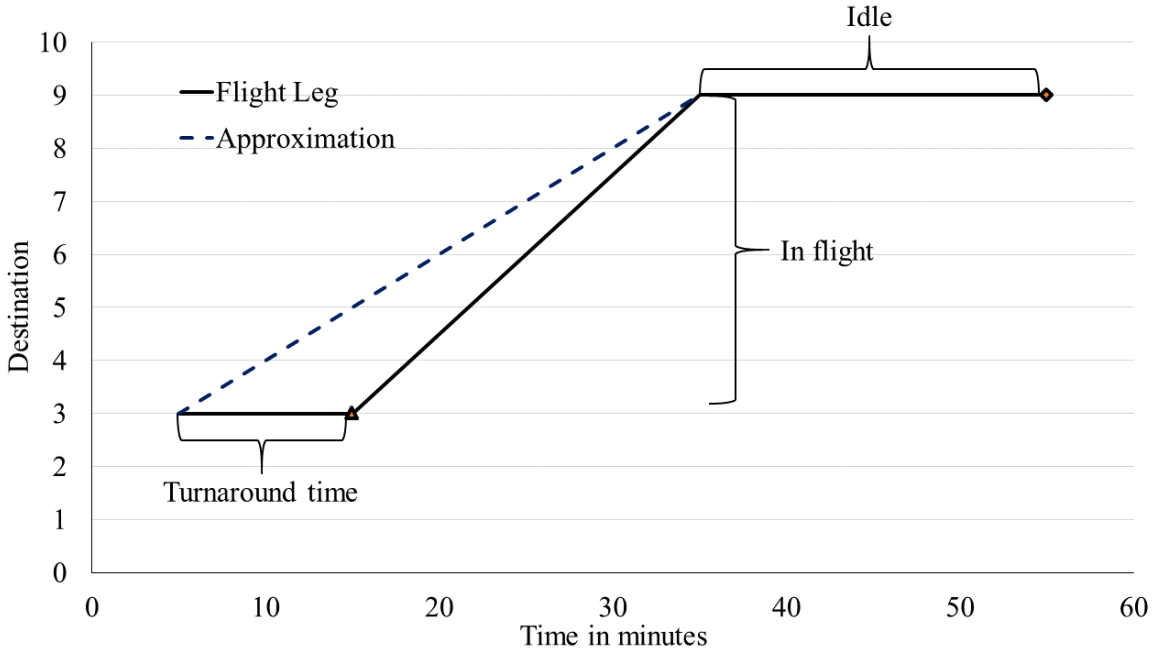

Figure 3.4: Flight leg description in a time space diagram.

3.2.2 Reactivity

Aircraft can do one of three things in the system. They can pick up a group of passengers, they can drop off a group of passengers or they can choose to remain idle (if the aircraft has passengers on board it can remain idle as long as it does not violate the late departure time constraint). This decision is based on the capacity of the aircraft at that point in time, the demand still left over in the system, the time difference between the aircraft and the passengers in the system and the passengers on the aircraft. These choices are then structured using the one pass vertical layering [42] see Figure 2.6. The reactive structure can be seen in Figure 3.5. This structure can be explained using five main reactive prompts the agents must be able to recognise:

1. Demand for pick ups are equal to zero and the capacity of the aircraft is greater than zero.
    - This means that there are no more passengers left in the system but there are passengers on board and they have to be dropped off.
2. Demand for pick ups are not equal to zero but the capacity of the aircraft is at its maximum.

- This means that no new passengers can be taken on board and so the agent will be forced to drop passengers off. In this case the aircraft should not remain idle.

3. Demand for pick ups are not equal to zero and the aircraft has capacity between zero and its maximum.

    - This means that the aircraft can either pick up more passengers, or drop off the passengers it has on board.

4. Demand for pick ups are not equal to zero but the capacity of the aircraft is equal to zero.

    - This means that the aircraft is free to fetch passengers in the system but cannot drop off passengers since there are no passengers on board.

5. Demand for pick ups are equal to zero and the capacity of the aircraft is equal to zero.

    - This means that all the demand has been satisfied. Therefore the aircraft should remain idle.

The aircraft should also have the choice to remain idle for each of the reactive statements above, except when the aircraft is full. There are two methods of ensuring that this reactive behaviour is met; the first is to hard code the reactive statements into the aircraft, and base all subsequent decisions on the capacity and the demand in the system, the second is to design the utility functions in such a way that the reactive statements are not broken, i.e. reduce the attractiveness of a move to zero if any of the capacity or demand constraints are broken. The first method is utilised in Model A and the second method in Model B.

3.2.3 Utility

Even though the agents have the reactive ability to gauge the demand in the system and compare that with the capacity of the aircraft, the aircraft still have to make individual choices about which passengers to on-board and which locations to fly to. Once the reactive function has been activated a set of available options are shown and the agent has to make a single choice from all of these available options. The choice

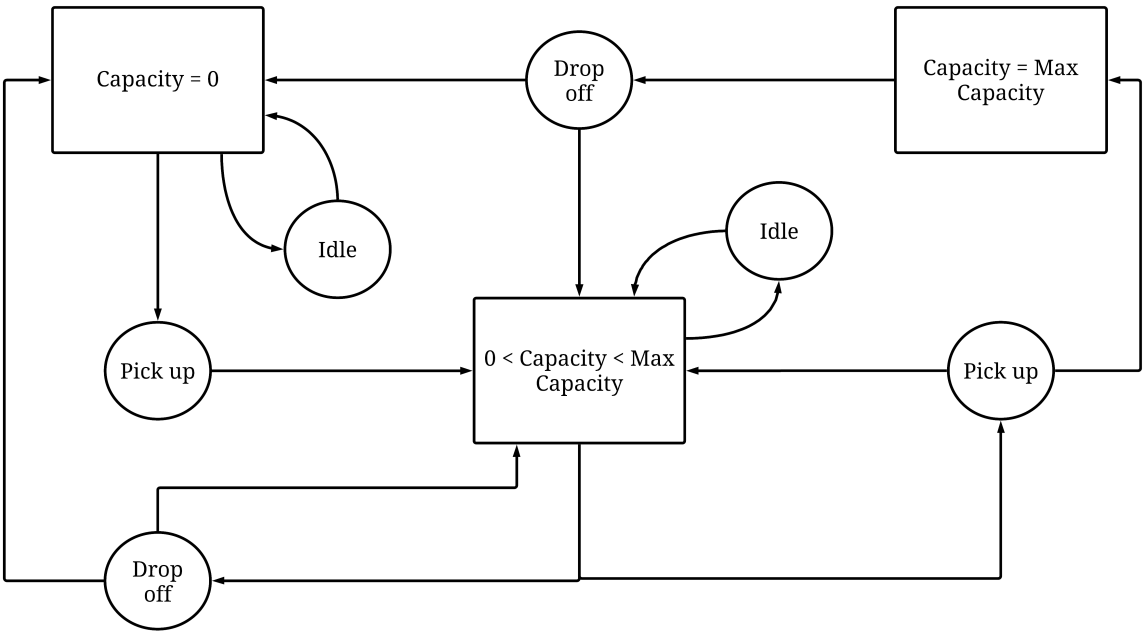

Figure 3.5: Reactive conceptualisation of DAFP agents.

selected will be made by means of Monte Carlo sampling. For aircraft to make a good choice there is a utility associated with each move.

At any given stage in the simulation, one of three outcomes can occur. Either the aircraft has to pick a group up, it has to drop off a group or chooses to remain idle. Only in one state can the aircraft decide between picking up people, dropping off people or remain idle. This is when its capacity is between zero and its maximum. In this case, agents can either pick up or drop off, depending on the outcome of the Monte Carlo selection procedure. When the aircraft has to fetch or drop off a group, the indicators taken into consideration are as follows:

- The distance between the agent current location and the candidate location.
- The early departure time.
- The late arrival time.
- the size of the group being collected.

In order for the agents to make a firm decision a utility function has to be implemented. Campbell suggests the use of exponential, linear and Gaussian functions as a extension to the research he conducted on ABSM[7]. This research tests all three methods to

determine if changing the utility function results in better decision making by the aircraft.

3.2.4 Environment of the DAFP

The agent environment is created by using the information from the booking list (see example in Table 3.1). At the beginning of the simulation, each aircraft has access to the information supplied by this environment and as each pickup is serviced the environment is updated sequentially. All aircraft can view this updated environment when they are about to make a decision. The environment contains information such as the pick up and drop off locations of groups, the number of passengers at those locations and the time windows associated with each of the groups. This means that the environment has some basic properties. These are as follows:

- The environment can be considered to be accessible since all the agents can gather information.
- The environment can be considered as deterministic. Aircraft can only service the pick ups and relocate for drop offs. Actions to the environment are always the same because only aircraft that have picked up a group of passengers affect the environment and aircraft that have relocated affect the environment.
- Since only agent behaviour influences the environment and demand is fixed at the start of the simulation, it can be considered to be static.
- The amount of possible actions that can be taken by the agents are always finite (it is completely dependent on the number of requests at the location of the aircraft) thus the environment is discrete.
- Agents operate using information from the entire environment, therefore the environment is non-episodic in nature.
- The environment contains the information about passengers at each location. Each aircraft has to be able to gauge where they are and where they are going. Therefore the environment has dimension.

3.3 Model design and global structure

This section covers the ABSM in detail. In order to design the different routing functions, a model was designed. The global structure in subsection 3.3.1 describes the general architecture of the model and shows how changes can be made to the model that effect the way aircraft make decisions and allows one to change the constraints of the problem. The structure of the model includes the design of the aircraft, environment and decision-making considerations. In subsection 3.3.2, the selection criteria of aircraft is discussed. The aircraft decisions will then be discussed in detail in sections subsection 3.3.3, subsection 3.3.4 and subsection 3.3.5. These cover how the aircraft chooses destinations and passengers. The Monte Carlo sampling procedure and utility function are discussed in subsection 3.3.6. Lastly the development of a genetic algorithm to optimise the weights of the decision factors is discussed in subsection 3.3.7. The code was written in MatLab 2014b and can be found in Appendix A.1.

3.3.1 Global structure

An ABSM was developed to test different pick up, drop off and relocate strategies. This design holds the general framework of the agent design. The model is illustrated in Figure 3.6. This diagram is a simplification of the complete model. It illustrates the most important features. These features can be described in five sections namely; initialisation, selection agent, routing decisions, environment update and stopping criteria.

The descriptions of each of the sections are as follows:

1. Initialisation: This involves reading the booking information and aircraft information from the data. This information is then used to set up the agent and environment variables in the program. The agent memory contains all the information about all the aircraft, such as the number of aircraft, the starting locations, the type, selection number, maximum speed and maximum capacity. Memory is initialised for each agent in the simulation. The environment contains information taken from the booking list, such as the group pick up points, drop off points and the number of passengers in each group.

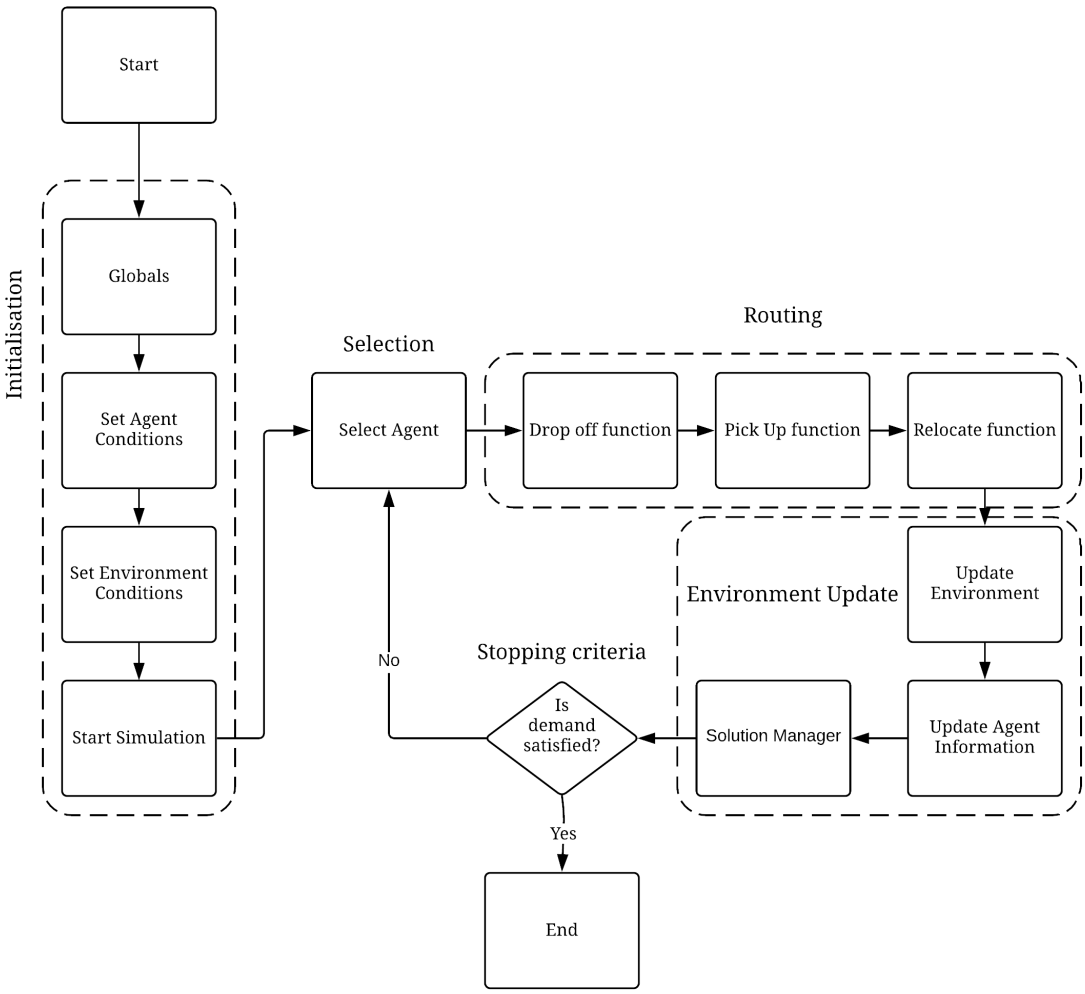

Figure 3.6: A illustration of a simplified program structure for the ABSM.

2. Selection Agent: Once the aircraft and environment have been defined by the program, the user will have to decide what routing method will be used. There are three main ideas for the routing method:

   (a) Vehicle by vehicle routing: Where a single aircraft is selected and is routed from the start of the day to the end of the day. After it has completed routing the next aircraft is selected.

   (b) Discreet event simulation: Here aircraft are routed in parallel by selecting the next event that occurs in time. Even though this was designed, it was not used in this research as preliminary testing showed poor results in comparison to the other selection mechanisms. This was potentially due to the poor utilisation of aircraft in comparison to the other methods.

(c) Monte Carlo: Where the aircraft to be repositioned next is selected based on a number of parameters such as the time the aircraft occupies, the number of groups on board and the type of aircraft.

These methods will be discussed further in subsection 3.3.2.

3. Routing decisions: After an aircraft has been selected it must be routed through one flight leg. The perspective of the aircraft was loosely based on the conceptual perspective put forward by Espinoza et al. [14] as seen in Figure 2.2. The decision is split into three parts, the drop off function that allows passengers to disembark, the pick up function that selects the passengers that will board and the relocate function that ultimately decides where the aircraft will fly to next. There were two methods used to model these routing functions. These methods were tested in this research. The models will be discussed in section 3.4 and functions will be discussed further in subsections 3.3.3, 3.3.4 and 3.3.5.

4. Environment update: This is the main means of communication between aircraft. After the aircraft has dropped off a group of passengers it will update the global demand letting all the other planes know that that group is no longer in the system. Likewise as a plane picks up a group of passengers, it must make it clear that no other plane can service that group. The aircraft will also communicate via the environment their current locations.

5. Stopping criteria: Lastly the simulation requires some stopping criteria. The aircraft are halted once there are no more passengers in the environment or on board any of the aircraft. Some passengers may be unserviced throughout the entire simulation. In this case if all the aircraft occupy a time that is later than the latest late arrival time then the simulation is stopped. There is no penalty associated with passengers that have not been serviced throughout the simulation.

3.3.2 Agent selection strategies

Aircraft in the simulation have to know when they are in a position to relocate or remain in the same location. This is achieved by means of a selection agent. The selection agent, as seen in Figure 3.6, is positioned before the relocate, pickup and drop off functions. Only an aircraft that has been chosen by the selection agent is then in a position to make decisions.

The selection agent assesses all the aircraft in the system, and chooses an aircraft that best suits the current state of the system. This agent decides the best aircraft using the following criteria; capacity, type of aircraft and the time that aircraft occupies. This information can be sampled using a Monte Carlo procedure. This makes the selection of aircraft a stochastic process. In this way more combinations of aircraft can be explored and thus the search space is covered more thoroughly. The weights of each parameter are shifted towards aircraft with lower capacities, cost and earlier times. Low capacity means an aircraft can house more passengers, with the aim to increase the utility of the aircraft. The same reasoning applies to the weights parameter of larger aircraft. Lastly the earlier the aircraft is the more potential it has to fetch passengers. The weights were chosen from preliminary testing.

This method is useful, since it can be easily modified so that users can implement different selection strategies. Apart from the method described above one could implement a vehicle by vehicle method, where each aircraft is routed from the start of the day till the end of the day. Only until an aircraft has completed a full day can the next aircraft be chosen and routed. One could also route the aircraft by only selecting an aircraft that has moved the least in time. This idea always ensures the next aircraft selected is in the most effective position to fetch passengers placed further in time, as well as reduce the number of aircraft required to solve the problem [7]. The downside to this method is that it may not always minimise the amount of aircraft used in the simulation. Preliminary testing of this method showed a increases the overall cost per hour. Because of these downsides the vehicle-by-vehicle selection was used in this research to test the DAFP. The alternate methods should be explored in more detail in further research.

3.3.3 Drop off function

The first major function the agent uses is the drop off function. It comes first so that the aircraft can always make space on board before any new passengers were taken on. As Figure 3.7 shows the aircraft needs to firstly assess all the passengers on board and determine who is due to disembark at the current location. Then the agent removes each passenger group one by one and adjusts the capacity. As a precaution, to ensure that all passenger groups are dropped off at their destinations before the late arrival time elapses, the agent will calculate how late the passengers are. If there are late passenger groups a penalty cost of R100 per minute is added and the group is then

recorded as being late. This penalty does not apply to passengers who have not been serviced at all, and only to passengers who have boarded the aircraft and are dropped off at their location after the late arrival time. Lastly the agent tells the environment that the group in question has been dropped off, thus communicating the fact to all other aircraft in the system.

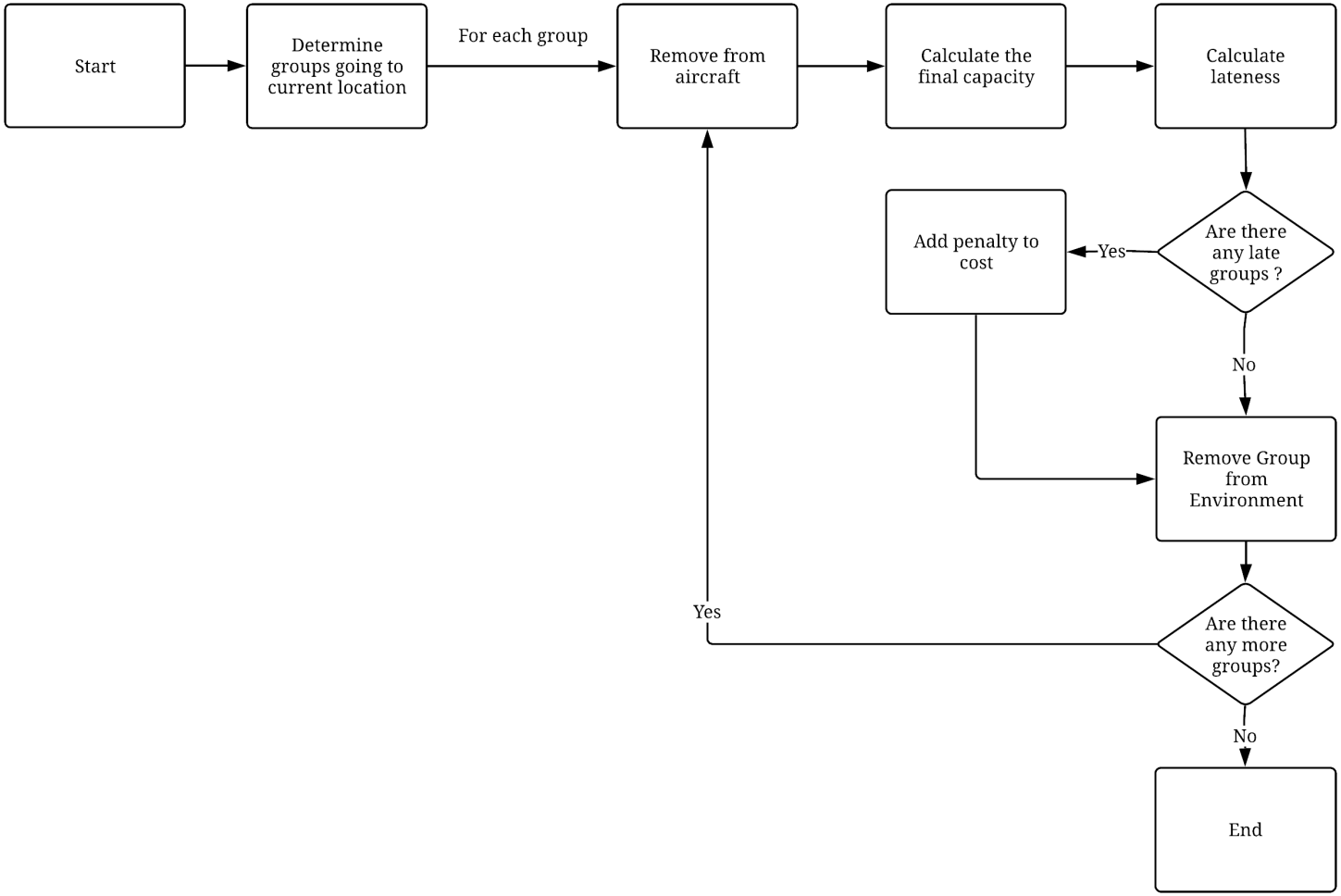

Figure 3.7: An illustration of a simplified drop off function.

3.3.4 Pick up function

The pick up function behaviour is similar to a bin packing algorithm (see Figure 3.8). For each group on the ground at the aircraft's location, it must decide who to take on board. It does this by firstly assessing how attractive each group is. Then each group must go through a series of reactive statements to remove attractiveness from choices that will result in infeasible solutions. The parameters used to asses the attractiveness of the groups waiting at the same location as the aircraft for pickup are the capacity of the groups C_w and the number of groups on board going to the same destination as the group under consideration M_2.

The reactive statements are then used to remove attractiveness from groups that will cause the problem to be infeasible, such as the overloading of the aircraft or if they cause other on-board groups to run late. The first is a checking function that ensures that the group in question can be loaded on board and the aircraft will still be able to deliver all the other groups it has on board without being late. If the group cannot be delivered or any of the on board groups cannot be delivered then the agent will reduce that group's attractiveness to zero. The second reactive statement asks if the group can fit on board. If it cannot then its attractiveness will also be reduced to zero. Next a group will be selected for pick up. In the pre-testing only the group with the highest attractiveness is picked up and the new capacity of the aircraft is calculated. Next there is a reactive statement that asks if the aircraft wishes to leave without fetching any more passengers, this is done by generating a random number and if that number is less than the parameter P_W for early departure (see subsection 2.1.5 factor six as per Campbells research [7]) then the aircraft leaves without assessing any more groups. This parameter is defined as:

$$P_W = \frac{1}{-e^{\frac{T_W}{F_W}}}$$

Where T_W is the amount of time the aircraft has to wait on the ground before it is able to take off. Otherwise, the aircraft will recalculate the attractiveness of the groups on the ground and continue the sampling process until it is told to leave, there are no more feasible groups at the given location or the aircraft is at capacity.

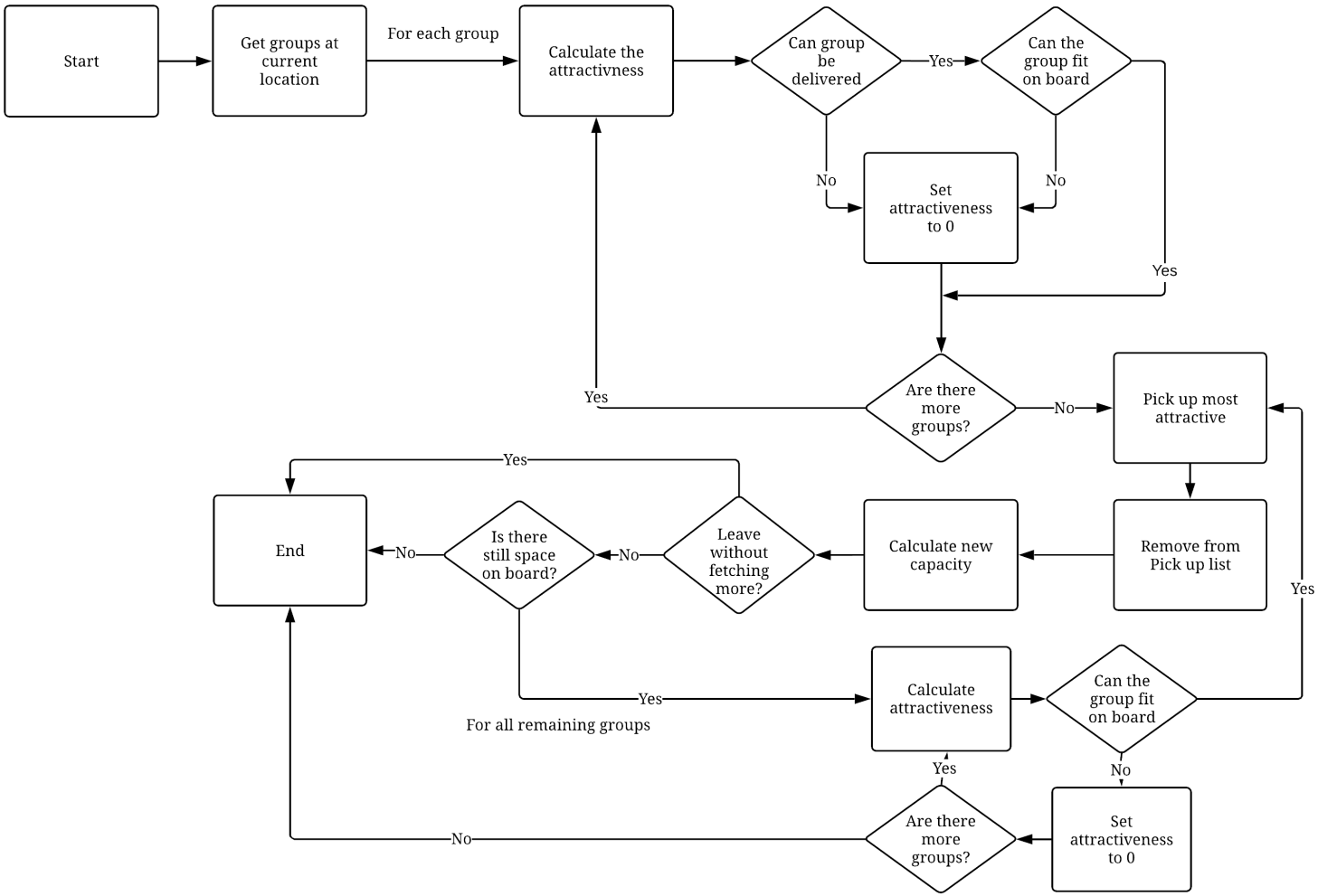

Figure 3.8: An illustration of a simplified pick up function.

3.3.5 Relocate Function

The relocate function decides where the aircraft will fly to next. The first thing the aircraft must check is if there are any passengers on board that need to be prioritised. If there is a group or groups that need to be delivered immediately the aircraft automatically routes itself to that location. If the groups need to be delivered immediately and have conflicting locations one of the groups will be delivered whilst violating the groups late arrival time constraint. It does this by calculating the urgency of each of the groups on board:

$$Tu = |L_w - Ta_w|$$

Where Tu is the time urgency of the group, L_w is the late arrival time of group w and Ta_w is the arrival time at the location of group w if the aircraft were to leave at its current time. If

$$Tu < Tu_1 \times Fu_2$$

were Tu_1 is the parameter for urgency, and Fu_2 is the factor associated with early departure. The group should be prioritised if this statement is true, the aircraft should fly directly to that groups destination of group w. This is to ensure that the group arrives at its desired location before the late arrival time expires.

If there are any groups on board that have been on board for two flight legs, that group is then prioritised and the aircraft will route itself to that group's location. This is to ensure no group is on board for more than three flight legs. If there are no priority groups, then the aircraft must assess all locations by assigning them levels of attractiveness (see subsection 3.3.6). The parameters used to determine the attractiveness of a given location are as follows; The capacity of the group C_w, the distance between the current location and the location under consideration D_{ij}, the number of passengers on board the aircraft going to the location j, the urgency of the groups $Tu1$ on board:

$$Tu1 = \frac{1}{e^{\frac{Tu}{Fu}}}$$

were, Fu is the weight of the urgency factor, the number of groups at the locations that have matching destinations M_2, the time closeness Tcl of the groups at the location:

$$Tcl = \frac{1}{e^{\frac{|E_i - T_l|}{F_{cl}}}}$$

were, F_{cl} is the factor associated with the time closeness parameter, T_l is the arrival time at the prospective location and E_i is the early departure time of the group under

consideration at the prospective location. These equations where developed from the equations governing decision making for agents by Campbell [7].

If a location has no pick ups or drop offs then it would make no sense for the aircraft to be routed there, so its attractiveness is set to zero, since the problem does not stipulate set end destinations for the aircraft. If the aircraft is full, then it should only be routed to destinations of the passengers on board. All locations that are still attractive must be checked to see if the aircraft can be routed there and still deliver all of the on board passengers on time. If going to a location means the aircraft cannot meet all of its engagements then that locations level of attractiveness is set to zero. The next destination is then chosen by means of Monte Carlo sampling (see subsection 3.3.6) and the selected location is the next destination. Note that the agent time changes with its change in location, as seen in Figure 3.4.

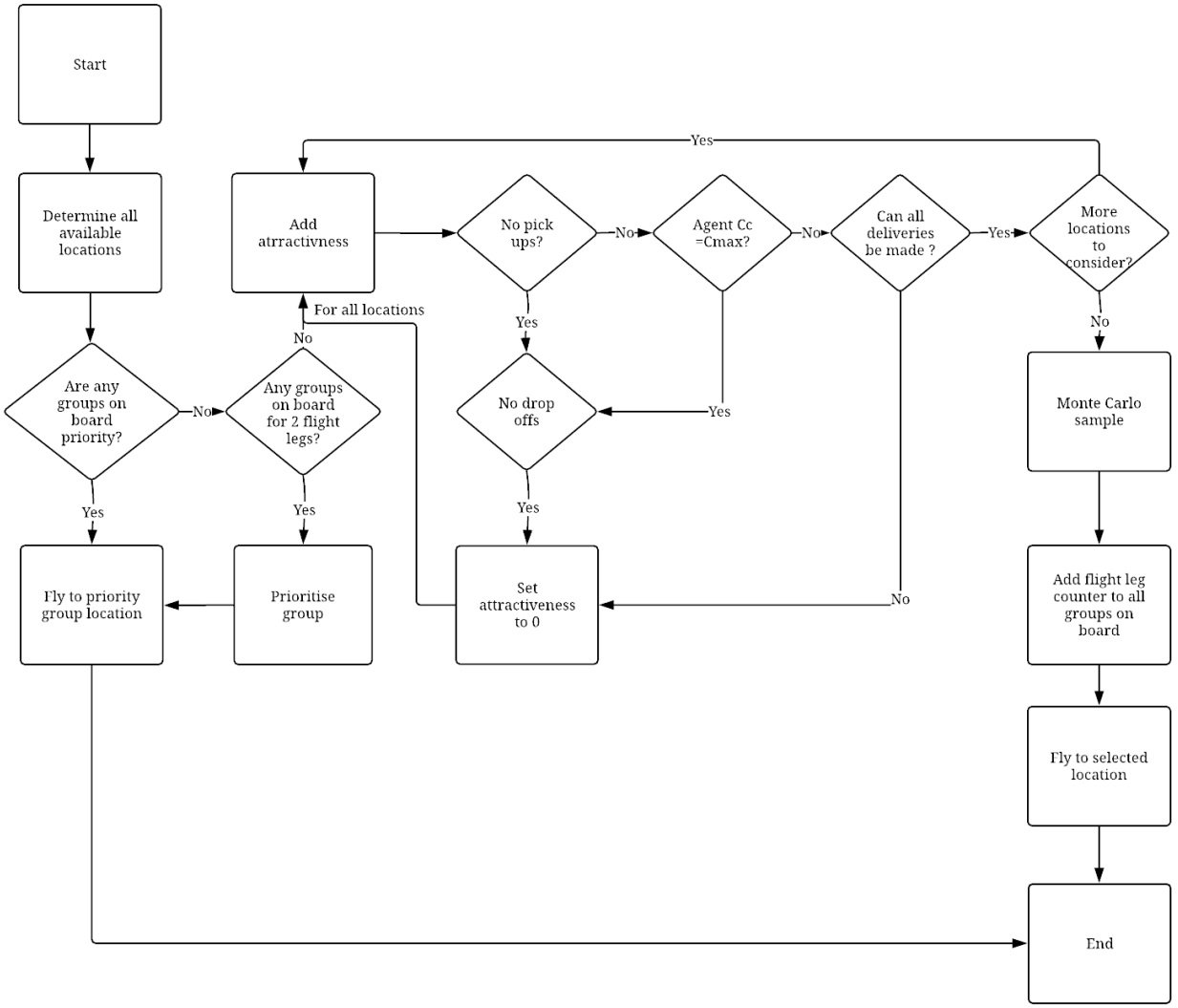

Figure 3.9: An illustration of a simplified relocate function.

3.3.6 Monte Carlo sampling and utility functions

Monte Carlo methods are used to allow agents to make decisions. The agents designed in this research can make sense of the environment by looking at the factors that influence their decisions. These factors can be aggregated into a numerical value that can then be sampled using a Monte Carlo sampling technique. A sample calculation is provided in Appendix B. This method relies on agents being able to effectively distinguish between good choices and bad choices. To aid in this process before any sampling takes place, the values of attractiveness are sent to a utility function that will boost the attractiveness of good choices and reduce the attractiveness of bad choices. The proposed utility function can be seen in Equation 3.1 and Equation 3.2. Equation 3.1 is the utility function that assigns attractiveness to groups waiting to be fetched and Equation 3.2 is used to assign attractiveness to locations considered by the aircraft. Once the attractiveness of all available moves is calculated the agent then makes a firm decision. $u(s \rightarrow a)$ is the attractiveness of the group under consideration from state s to state a and is calculated as follows:

$$u(s \rightarrow a) = C_w Fc + M_2 F_{M2} \tag{3.1}$$

Where:
s is the current state of the agent.
a is the next state.
Fc is the factor that adjusts the attractiveness of a groups capacity.
M_2 is the number of matching destinations between the groups on the aircraft and the groups on the ground.
F_{M2} is the factor that adjusts the matching groups attractiveness.

The attractiveness of a location is calculated as follows:

$$u(s \rightarrow a) = \frac{1}{e^{\frac{D_l}{F_d}}} + \frac{M_1}{F_{m1}} + \sum_{j=1}^{M_1} \frac{1}{e^{\frac{\left|L_j - Ta_j\right|}{F_u}}} + \sum_{i=1}^{n} \frac{M_{2i}}{F_{m3}} + \frac{1}{e^{\frac{\left|E_i - Ta_l\right|}{F_{cl}}}} + C_w F_c \tag{3.2}$$

Where:
D_l is the distance from the current location of the aircraft to the location under consideration.

F_d is the factor that influences the attractiveness of distance.
M_1 is the number of groups on board the aircraft going to the location under consideration.
F_{m1} influences the attractiveness of number of groups on board the aircraft going to the location under consideration.
L_j is the late arrival times of the groups on board going to the location under consideration.
F_u is the factor influencing how attractive urgent destinations are.
M_3 is the number of matching destinations between the on board passengers and the group under consideration at the location under consideration.
F_{m3} is the factor that influences the attractiveness of M_3.
E_i is the early departure time of a group under consideration at the location under consideration.
F_{cl} is the factor that influence how attractive a group is based on how close they are to the aircraft in terms of time.

It can be seen in Equation 2.18 that some of the components in Equation 3.2 make use of an exponential function proposed by Campbell [7]. This work will not only look at the use of exponential functions but also Gaussian and linear (see Figure 3.10). These functions all degrade at different rates and so should provide variations in results. This means that the level of attractiveness will vary for the same decision in different models and provide a view of which function generates the best overall solution. In order to test these different distributions, a second method of calculating utility values was devised making use of a multi-criteria decision method applied in Model A (the sample calculate can be found in section B.2). This method makes all factors that influence an agent's decision equal in weight by first scaling the values to an interval between 0 and 1, then summing the values and applying the values to each utility function. Lastly, these values make up the distribution representing the attractiveness of choosing a group to pick up or a destination to fly to. This distribution is then sampled using the Monte Carlo process given above. This test should provide enough information to decide which of the three utility functions perform best. Since the best factors that influence decisions are unknown, a genetic algorithm was developed to optimise the factors for each booking list. This algorithm will be covered in more detail in subsection 3.3.7.

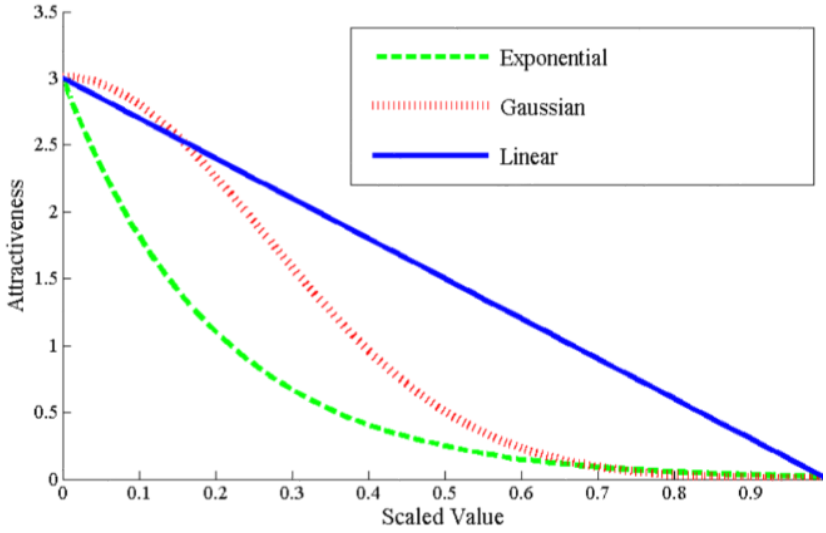

Figure 3.10: Utility Functions used to map attractiveness of pick ups and relocations.

3.3.7 GA weight factor optimisation

A genetic algorithm was devised in order to optimise the weights of each of the factors. The method takes the agent routing functions and uses them as a fitting function. The weights are then used as the genes within the chromosome (see Table 3.6). Each gene in the chromosome is a weight that is applied to each decision factor. These weights will determine what level of priority is given to each of the factors that influence a agents decisions. The GA principal of operation is shown in Figure 3.11. In the figure the algorithm starts by generating a starting population of ten random chromosomes, then uses the agent based routing functions as fitting functions and assesses each of the solutions. The seven best solutions are kept as part of the next generation's population. Two of the five worst and four of the best five chromosomes are sampled. These six new chromosomes are then sampled using Monte Carlo sampling and are paired in twos as parent chromosomes. These paired parents are crossed over at a random point on the chromosome producing three children (they can produce six children each but to keep the initial population constant the first child of each parent is chosen). Each gene in the three children's chromosome are then subject to mutation, defined by a mutation rate R_m. After this process, the children rejoin the main population and the cycle starts again until the number of predefined iterations has been met.

Because the routing functions are stochastic they would not be suitable as a fitness function for the genetic algorithm. In order to optimise the weights, all Monte Carlo sampling that had occurred in each of the functions had to be removed and the agent

is forced into selecting the best options only. In terms of the agent selection process, none of the strategies described in subsection 3.3.2 could be used. Instead, aircraft are selected one by one and routed from the start of the day to end of the day only selecting the best choice to the routing functions and their weights. This means that it could not be expected that the results would be of high quality, but they should produce good weights. Also, it is not expected that the final weights would be the same for each schedule. Different schedules will have different requests and thus agents may need to prioritise groups differently.

The genetic algorithm was used to optimise the weights of each of the factors of influence so as to reduce the cost of the schedules generated. The main aim here was to firstly determine the best weights, and secondly determine if they are the same for all booking lists. If they are the same then it means that no matter how the requests are structured within a booking list the level of attractiveness can always be described and calculated in one way.

Preliminary tests show that solutions would converge after three hundred iterations of the model and so this number of iterations is used to conduct the weight optimisation.

Table 3.6: Defined GA chromosome structure.

Factor	Description
F_u	Influences how attractive urgent destinations are
F_d	Attractiveness of distance
F_{cl}	Associated with the time closeness parameter
Fc	Attractiveness of a groups capacity
F_{m3}	Attractiveness of matching destinations between on board and ground groups
F_{m1}	Attractiveness of the number of groups on the aircraft going to a location
F_{M2}	Factor that adjusts the matching ground groups attractiveness
F_W	Influences the amount of time the aircraft waits on the ground before take off
F_{u2}	Associated with the attractiveness of early departure

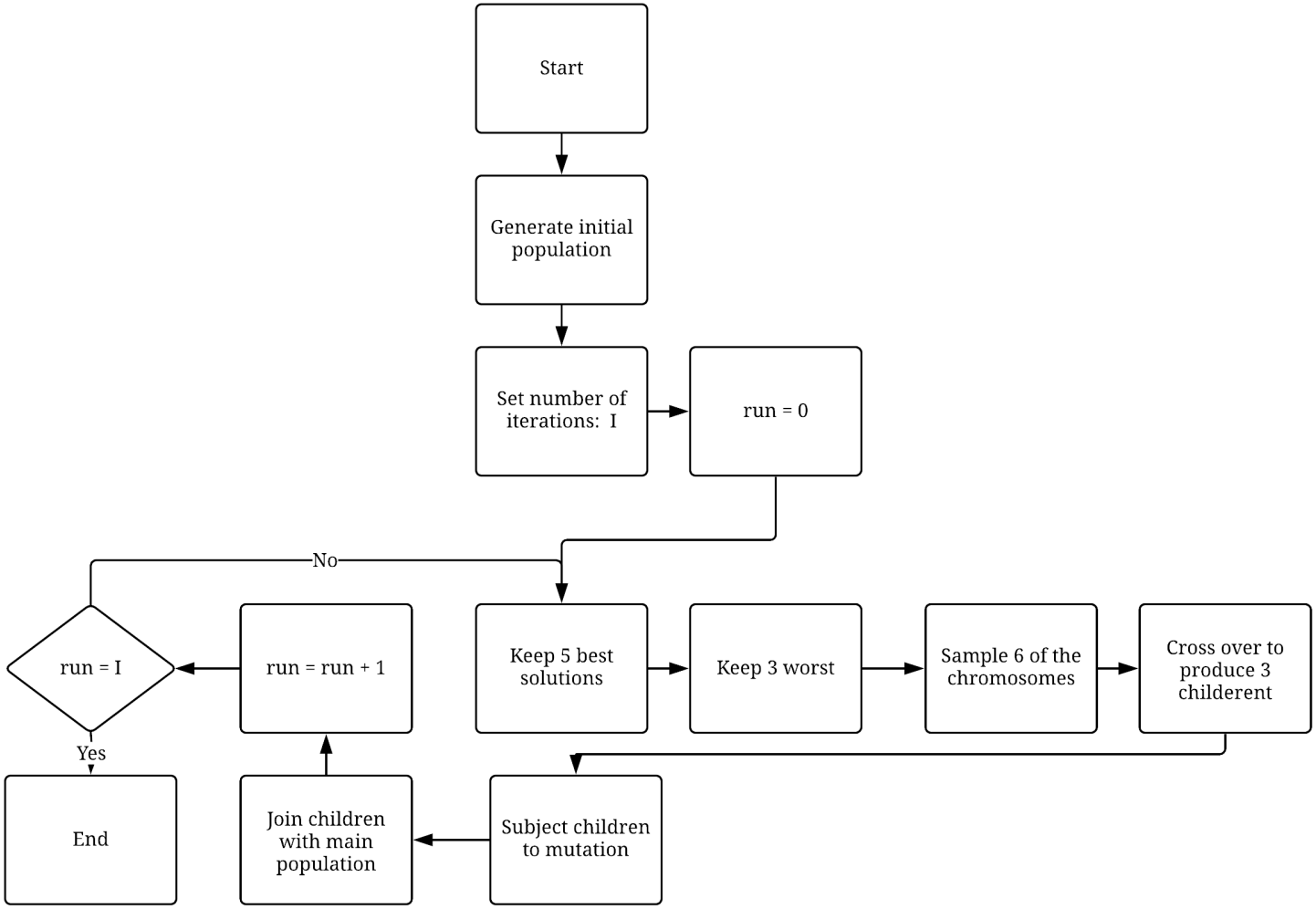

Figure 3.11: Genetic algorithm used to optimise model weights.

3.4 Model descriptions

There were two models designed and tested in this research. They both use the same general architecture as discussed in subsection 3.3.1, but make use of two different techniques to route the aircraft through the system. For reference in this paper they will be referred to as Model A and Model B.

Model A:

Model A, makes use of the multi criteria decision analysis (MCDA) as its means of making decisions. In this model, the agents are subject to tight reactive constraints. The aircraft make decisions, by assessing all options available, eliminating ones that violate the constraints of the DAFP, and then applying a MCDA to make a final decision. The model also allows both deterministic and stochastic decision making processes. This means that aircraft can be told to only make the best choices or they can apply a Monte Carlo process to select an option. In this research, Model A is only tested against the DAFP with constraints stated in subsection 3.1.1 however it can be applied to other versions of the problem.

The testing of Model A consist of short runs of 100 iterations and long runs with 1000 iterations each tested twice with and without Monte Carlo sampling. The model would also produce a frequency distribution (see Figure 4.8) to show the probability of a given output as well as generate a geographical plot (see Figure 4.9) and time-space network (see Figure 4.10) to illustrate the route the aircraft would take. Lastly, the models produce a schedule indicating the departure and arrival times see Table 4.7.

Model B:

Model B, is based on a method put forward by Campbell [7]. In this model, the reactive nature of the aircraft are slightly relaxed, this means that agents view all options available to them and then assign a value of attractiveness. If the option violates any of the DAFP constraints, then the aircraft will assign the option no attractiveness. These values of attractiveness are calculated by means of a utility function, and are discussed further in subsection 3.3.6. The model also allows both deterministic and stochastic decision making processes. This means that aircraft can be told to only make the best choices or they can apply a Monte Carlo process to select an option. In this research, Model A is only tested against the DAFP with constraints stated in

subsection 3.1.1, but a flight leg constraint is also included such that no passenger can stay on board the aircraft for more than three flight legs.

The model B tests take the standard DAFP and add a flight leg constraint. Passengers desire as few flight legs as possible so each aircraft prioritises a passengers when the number of legs travelled is one less than the maximum allowed. As with the unconstrained tests (see section 4.5), each booking list is tested on a short and long run with 100 and 1000 iterations respectively. They are tested with both Monte Carlo sampling and without. The same outputs are then produced by the model, frequency distribution, geographic illustration, time space network and final schedule.

Both models are compared to the academic benchmarks and manual schedules as stated in subsection 3.1.2.

The testing procedure will order the best utilities first and sample them by means of a Monte Carlo process. The best factor to use cannot be predicted and will be optimised by means of a genetic algorithm, the design of which is covered in subsection 3.3.7. These factors essentially provide the goal-orientated nature of the agents. They influence the agent's decisions towards good moves.

4 Observations and Results

The ABSM was designed and tested on the seven booking lists provided given in Appendix C. This section provides the results of the seven booking lists and also illustrates the effectiveness of the model by giving examples of the final schedules.

4.1 Testing procedure

The test procedure is broken up into five main parts. These are as follows:

1. GA tests, to minimise weights against the ABSM fitting function for all booking lists.
2. Utility function tests, using the un-weighted and weighted model to find the most effective utility function.
3. Repeatability test, for practical use of the model in industry one has to know whether the models can produce similar results consistently and so the repeatability of a test has to be examined.
4. Model A testing, using both the unweighted model and weighted model.
5. Model B testing, adding the flight leg constraint and testing all booking lists.

4.2 Genetic algorithm testing

The GA terminates after 300 iterations for each booking list. The cost vs. generation graph for the booking list with 99 requests can be seen in Figure 4.1. The graph

shows each generation of new chromosomes (across 300 iterations of the model) and the resulting final cost of the ABSM. From this figure it can be seen that the objective function is reduced from the first generation. It also shows variations from the minimised value due to the genetic variations induced by mutation. This variation is illustrated by Figure 4.2 which shows the best cost and mean cost over 300 iterations of the model. This figure shows that the function reduces costs and the mean cost fluctuates showing a continual search for better solutions. The results from this test shows an improvement in solution quality for the deterministic model. To show that the new factors have a significant effect on the stochastic model, further tests were conducted.

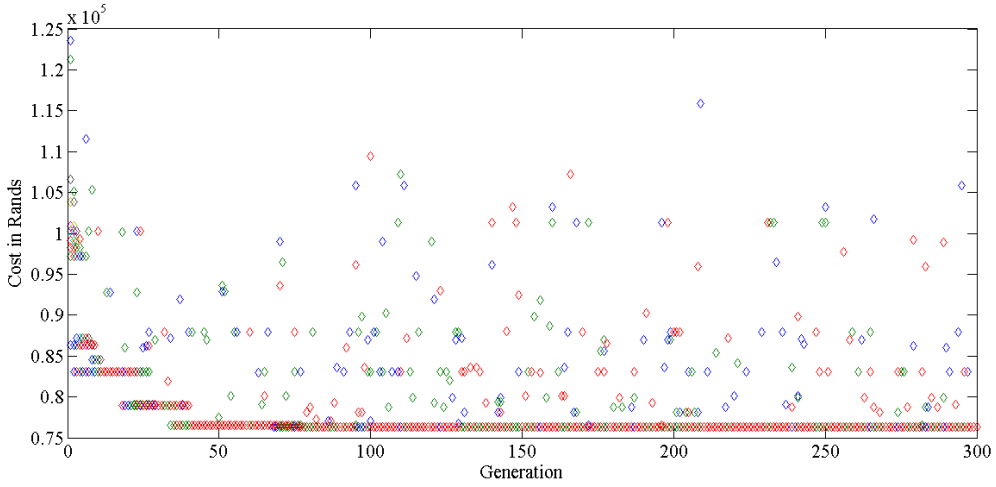

Figure 4.1: Costs generated by the GA for a booking list with 99 requests.

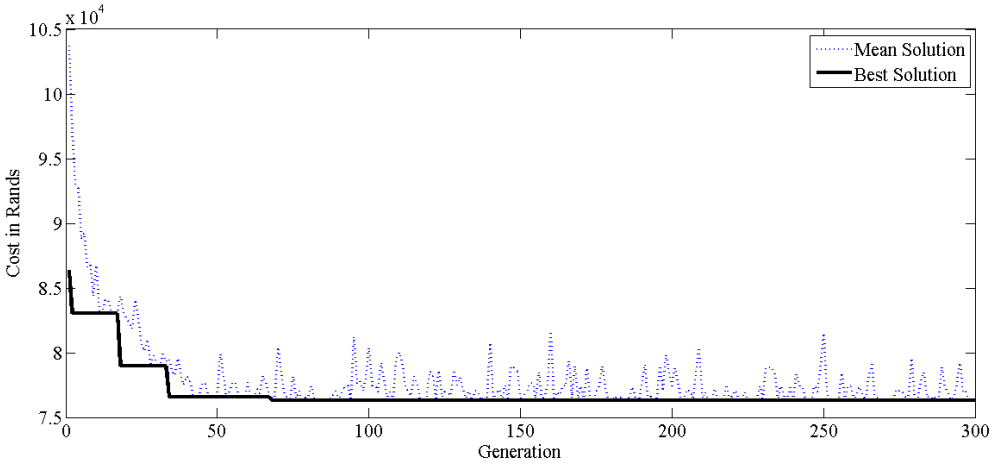

Figure 4.2: Mean solutions and best solutions for a booking list with 99 requests.

The resulting chromosomes for all booking lists can be seen in Table 4.2. These numbers represent the weights for the nine factors that influence an aircraft decision in the simulation.

A further test was conducted to see if the reduced weights had similar effects in reducing costs in other schedules. Table 4.1 shows the resulting weights applied to other booking lists, where 'X' indicates no improvement to the current best solution and '-' indicates the best solution produced. This was done to ensure the solutions produced by the GA did not find local solutions for each of the booking lists. There is only one instance where another booking lists weights (the final factors produced by S40 F) found the best solution when applied to a different booking list (S10). Other than this finding, there is no clear trend from the data so this means that the agents favour different parameters when approaching different booking lists. For example the booking list with 99 requests shows a greater emphasis on the size of the candidate groups than the booking list with 10 requests. The same variations can be seen for all other booking list sizes. This is an interesting phenomenon since it suggests a need for a more dynamic means of adjusting the weighted parameters.

Table 4.1: GA factors applied to different booking lists.

Weights	S10	S39	S40	S99	S102	S139	S200
S10 F	-	X	X	X	X	X	X
S39 F	X	-	X	X	X	X	X
S40 F	-	X	-	X	X	X	X
S99 F	X	X	X	-	X	X	X
S102 F	X	X	X	X	-	X	X
S139 F	X	X	X	X	X	-	X
S200 F	X	X	X	X	X	X	-

The time taken to solve each booking list is 85 minutes (see Table 4.3). The solution quality was comparable with other research conducted [47], with small to medium sized booking lists producing results within 16% and 42% percent of the upper bound. Larger booking sizes showed worst results with deviations within 36% and 57% of the upper bound.

Table 4.2: Final chromosomes for all booking lists.

Weights							
Weight	**S10**	**S39**	**S40**	**S99**	**S102**	**S139**	**S200**
F_u	8,52	23,32	5,41	39,73	17,33	44,14	24,46
F_d	24,75	32,29	28,11	90,64	71,95	24,19	73,17
F_{cl}	13,21	1,73	6,49	9,23	26,60	8,85	31,03
F_c	0,00	0,21	0,02	0,91	0,14	0,43	0,56
F_{m3}	3,16	9,24	1,37	6,39	1,39	6,76	8,03
F_{m1}	4,76	3,47	2,37	1,03	2,21	1,09	2,60
F_{m2}	0,42	0,13	0,80	0,96	0,54	0,67	0,54
F_w	45,23	44,18	7,49	36,67	29,56	13,77	40,28
F_{u2}	42,40	30,73	27,47	44,38	24,98	28,77	30,79

Table 4.3: Reduced cost and CPU time produced by the GA as applied to the ABSM fitting function.

Booking list	**Cost (R)**	**Benchmark (R)**	**Deviation**	**Run Time (Sec.)**
S10	15310,62	12826	19,37%	1116
S39	24100,90	20856	15,56%	1532
S40	60158,47	34531	74,22%	1998
S99	76300,53	96395	-20,85%	4709
S102	147730,35	63270	133,49%	6692
S139	110223,11	49346	123,37%	10834
S200	128060,54	81740	56,67%	8511

4.3 Utility function tests

The utility function determines the final attractiveness of all the available choices the aircraft can make given in Equation 3.2. It is, therefore, imperative to discover which type of function performs the best so that it can be incorporated into the final model. As stated in subsection 3.3.6, three functions could be used, namely Gaussian, linear and exponential functions. In Campbell's [7] work, it has been seen that the exponential function performs well when applied to this method.

The agent routing function is run for 100 iterations using the exponential function, then the Gaussian and lastly the linear function. The results are then arranged from highest to lowest forming a profile. These profiles are then easily comparable and one can determine visually which of the functions performed the best. Model A and Model B were tested in this sequence. For Model B, were the weights of each of the functions had been optimised using the GA in subsection 3.3.7.

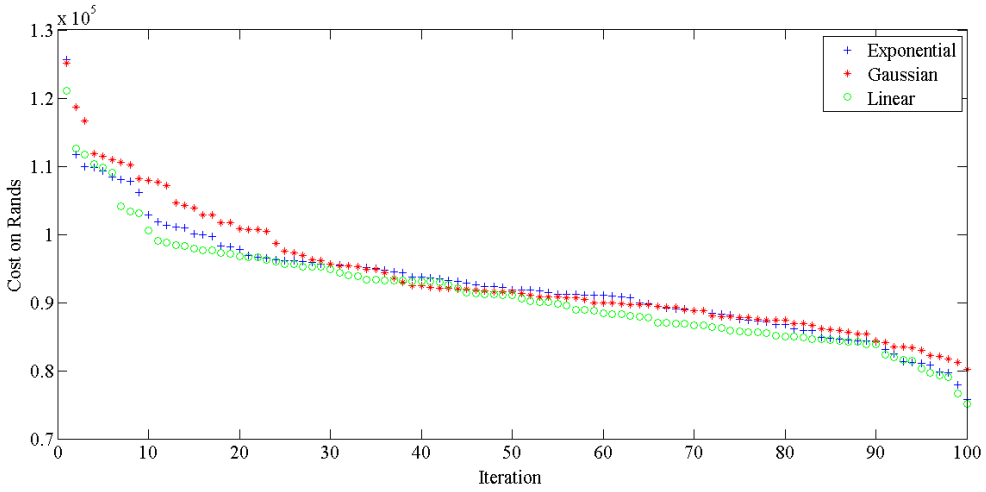

Figure 4.3: Results profile of model A on a booking list with 99 requests.

From Figure 4.3 it can be seen that the utility functions have no effect on the result profile of the Model A. This is true for all the booking lists tested. The reason for this is that since all the factors are scaled between zero and one, before they are sent to the utility function they are already ordered from best to worst. This means that the functions have little to no effect on the outcomes. From Figure 4.4 it is clear that, for Model B, the linear function performed the worst and the exponential and Gaussian functions are on par with one another. This is true for all the booking lists tested.

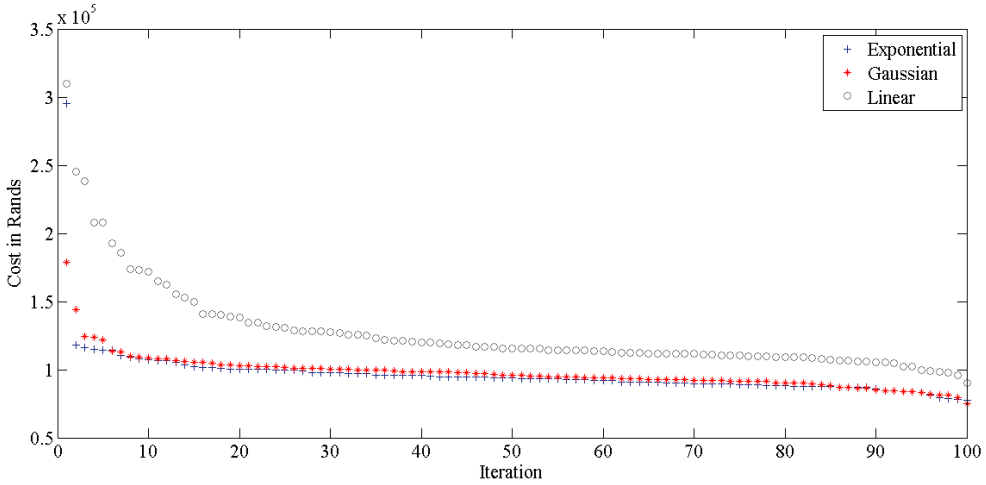

Figure 4.4: Results profile of model B on a booking list with 99 requests.

From these results, it was decided that for all further tests (in section 4.5 and section 4.6) the exponential function will be used since the result profile showed the most promising results in terms of solution quality.

Table 4.4: Average cost generated through testing of each utility function for each model.

	Model A Cost (R)			**Model B Cost (R)**		
Booking list	**Linear**	**Gaussian**	**Exp.**	**Linear**	**Gaussian**	**Exp.**
S10	14726,06	14444,67	14553,74	16745,62	13382,26	13277,12
S39	26869,10	27050,09	27322,59	36496,39	33122,97	33030,12
S40	43927,61	44521,01	43030,28	62016,21	56266,38	55218,54
S99	91446,88	93726,71	92764,86	127366,11	97736,47	96530,09
S102	106485,64	105180,32	99063,05	190981,85	150182,59	154816,59
S139	107820,92	108646,88	106802,51	189972,69	142236,14	131147,30
S200	187225,63	185223,39	189460,07	215793,92	176384,80	188084,84

Table 4.5: Performance of utility functions for each model.

Number of times lowest result reached					
Model A			**Model B**		
Linear	**Gaussian**	**Exp.**	**Linear**	**Gaussian**	**Exp.**
2	2	3	0	2	5

4.4 Repeatability

Due to the fact that this method has no improvement mechanism it is of importance to see if the models results can be repeated. To do this, each of the booking lists are tested repeatedly over short runs (test 1 - 100 iterations) and long runs (test 2 - 500 iterations). The normal distributions are then compared to see if they differ.

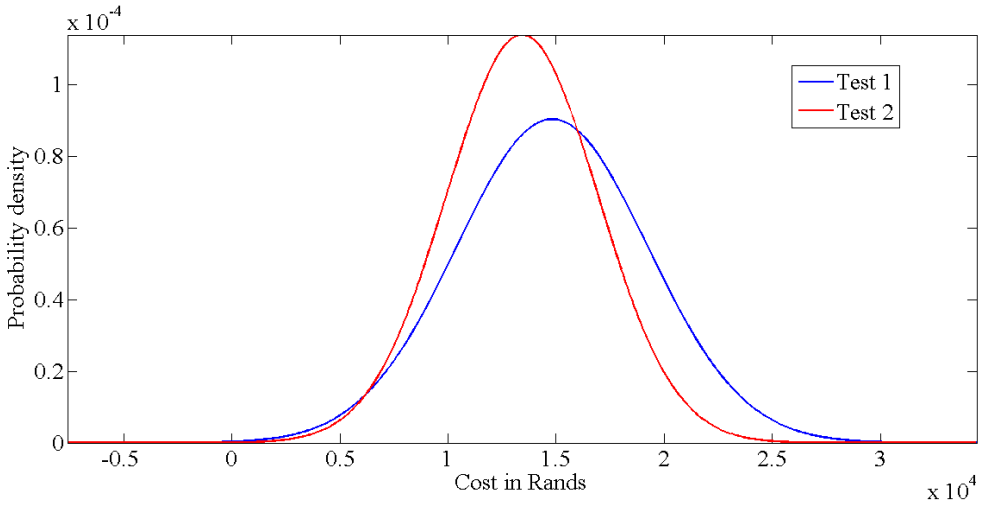

Figure 4.5: Short run normal distributions of ABS model testing a booking list of 10 requests.

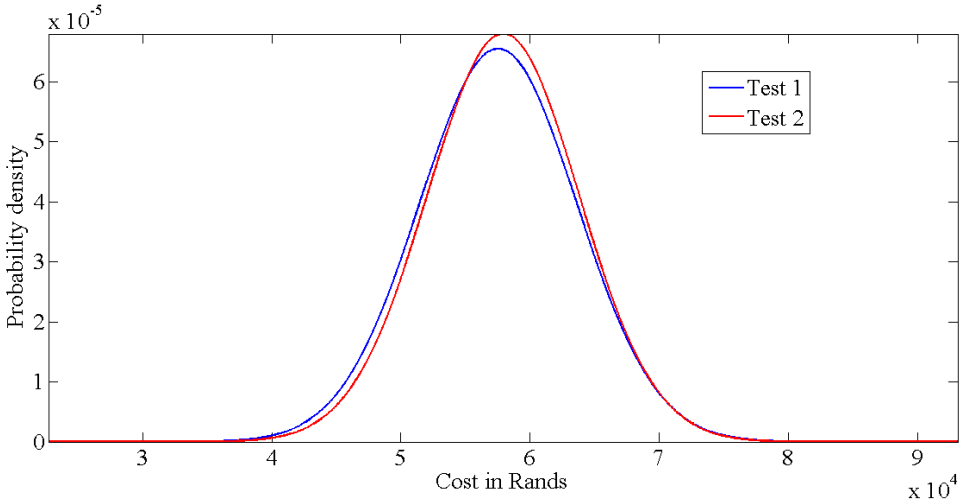

Figure 4.6: Long run normal distributions of ABS model testing a booking list of 40 requests.

From Figure 4.6 and Figure 4.7 it can be seen that over long runs the model produces a normal distribution that is almost identical to that of the previous runs. This means that when the chance of the best result being repeated when the model is repeated is

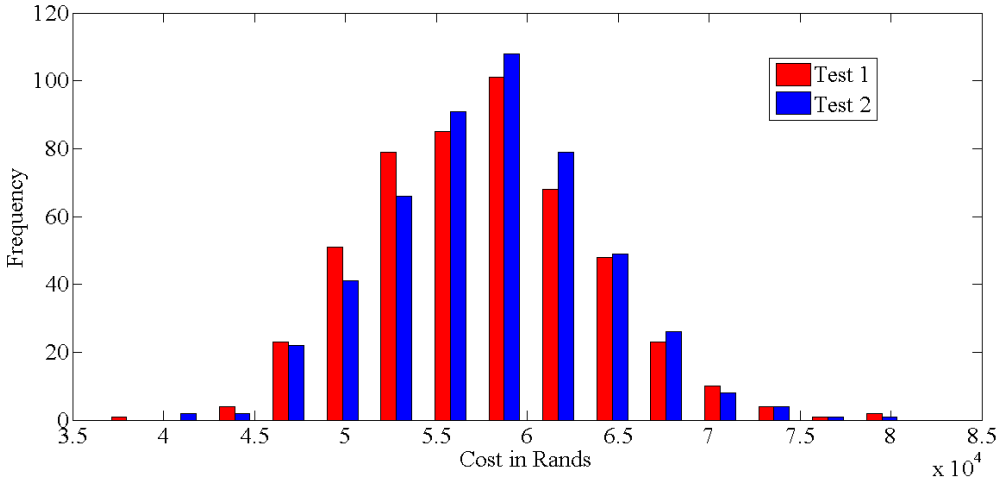

Figure 4.7: Long run frequency distributions of ABS model testing a booking list of 40 requests.

very high. This result can be seen when the experiment is repeated using all seven booking lists. Under short runs the result varies. As seen in Figure 4.5, on shorter runs the chance of repeating the results are significantly less as seen in the change in shape of the normal distributions in subsequent tests. This means that the only way to ensure the repeatability of any given result is to run the models for longer periods of time.

4.5 Model A tests

This section covers the use of ABSM when applied to the DAFP without the flight leg constraints. Model A was applied to all seven booking lists and the full set of results can be seen in Table 4.6.

All final costs were compared to both the respective upper bound and manual schedule costs Table 3.4. However, the solution quality significantly reduced as the size of the schedule increased. For the smaller schedules (size 10 to 99) the solutions were, in the best cases, far below the upper bound of the fully constrained models and in the worst cases only deviated from the upper bounds by 5%. This shows promise ,in small booking lists, for the method and since the model relies on the constraints to formulate decisions, it is expected that by adding more constraints the solution quality should

Table 4.6: Final Costs for Model A.

	Short Run		Long Run	
Schedule	**Monte Carlo**	**Deterministic**	**Monte Carlo**	**Deterministic**
S10	R 11 452,77	R 11 420,99	R 10 732,37	R 10 369,94
S39	R 23 873,60	R 23 663,26	R 22 676,56	R 22 077,02
S40	R 35 074,25	R 33 945,87	R 32 801,76	R 32 349,11
S99	R 79 758,02	R 78 001,81	R 77 705,33	R 74 630,80
S102	R 73 059,51	R 74 150,45	R 73 092,28	R 70 241,03
S139	R 125 929,77	R 109 453,49	R 124 632,26	R 104 822,03
S200	R 133 433,60	R 112 164,08	R 130 983,24	R 112 008,26

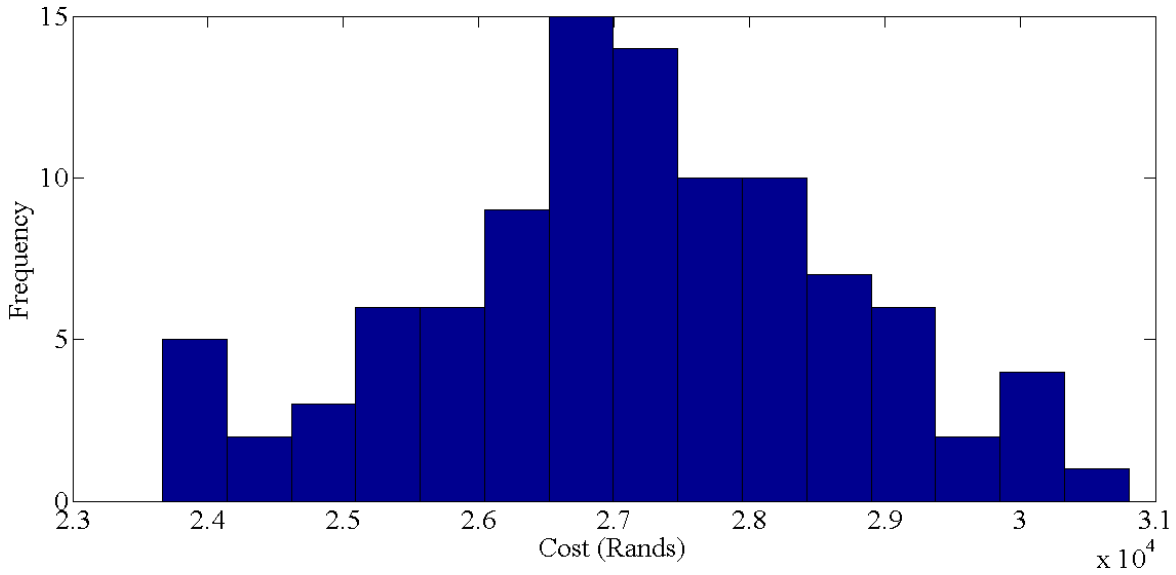

Figure 4.8: Frequency distribution of short run Model A test of a booking list with 39 requests.

increase, this is because this will allow the aircraft to eliminate more options that will violate constraints, effectively reducing the search area.

Table 4.8 shows the CPU run time to execute the agent based model. From this it can be said that the time taken is far longer than other heuristic techniques [3] [53] [47]. Table 4.9 shows the percentage deviation from the benchmarks. Positive deviations mean that the result was worst than the academic benchmark and negative deviations mean the result was an improvement on the academic benchmark (see Table 3.4).

Table 4.7: Final schedule of a booking list with 39 requests using Model A.

Type	**Origin**	**Departure**	**Dest.**	**Arrival**	**Cap. at Origin**	**Groups**
2	7	390	10	416	5	10 8
	10	660	7	686	3	8 19
	7	686	11	727	4	8 11
	11	727	8	745	1	8
2	11	390	17	410	0	
	17	660	4	712	2	22
1	7	390	26	423	0	
	26	423	3	481	8	23
1	7	870	8	910	7	9
2	8	660	17	684	5	16 14 15
	17	684	11	704	4	16 14
	11	704	10	733	3	16 20
	10	733	7	759	1	20
	7	865	4	921	4	7
	4	921	10	961	0	
	10	961	7	987	5	18
2	4	390	3	451	3	3
	3	610	11	703	4	2 1
	11	703	7	744	2	1
	7	744	3	839	2	4
1	7	870	17	909	6	12
2	4	390	8	428	0	
	8	775	11	793	1	17
	11	793	4	836	4	17 21
2	7	390	4	446	5	5 6 13
	4	446	26	475	1	13

Table 4.8: CPU time for all test problems in seconds using Model A.

	Short Run		**Long Run**	
Schedule	**Monte Carlo**	**Deterministic**	**Monte Carlo**	**Deterministic**
S10	264,13	152,28	2388,86	2222,49
S39	215,43	187,36	2154,10	1846,67
S40	236,75	244,12	2603,75	2256,94
S99	386,59	383,89	1017,87	961,23
S102	674,73	545,89	1838,57	1732,91
S139	751,48	1242,51	2589,64	3753,58
S200	764,92	893,74	2145,17	2797,08

Table 4.9: Percentage improvements and deviations from respective benchmarks using Model A.

	Short Run		**Long Run**		
Schedule	**Monte Carlo**	**Deterministic**	**Monte Carlo**	**Deterministic**	**Benchmark**
S10	-10,7	-11,0	-16,3	-19,1	Upper bound
S39	14,5	13,5	8,7	5,9	Upper Bound
S40	1,6	-1,7	-5,0	-6,3	Upper Bound
S99	-17,3	-19,1	-19,4	-22,6	Upper Bound
S102	15,5	17,2	15,5	11,0	Upper Bound
S139	155,2	121,8	152,6	112,4	Manual
S200	63,2	37,2	60,2	37,0	Manual

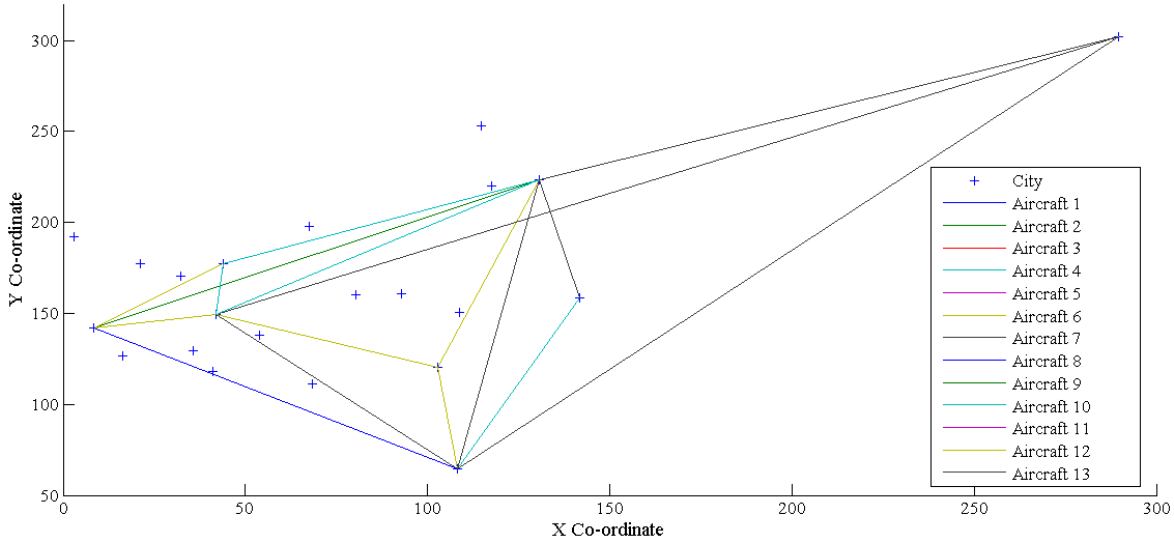

Figure 4.9: Geographic illustration of short run Model A test of a booking list with 39 requests.

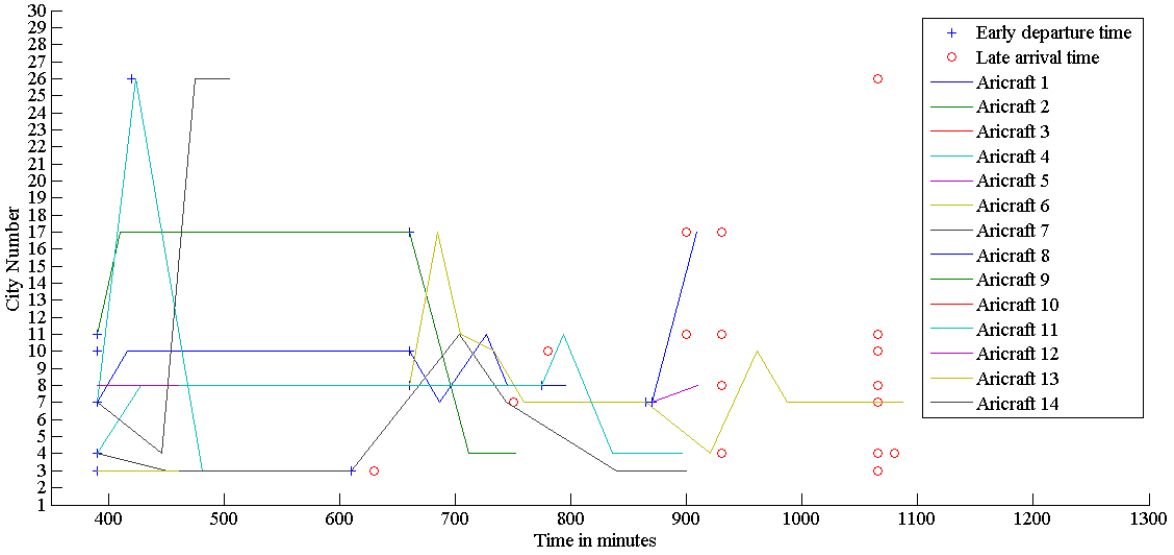

Figure 4.10: Time space network of short run Model A test of a booking list with 39 requests.

4.6 Model B tests

The final cost from these schedules is then compared to the respective benchmarks to determine the success of the method. As per the results from section 4.2 and section 4.3, each schedule was tested using the exponential utility function and the weights from the respective GA test. The final costs can be seen in Table 4.10. The deviations from the respective upper bound and manual schedules can be seen in Table 4.11. The CPU

time can be seen in Table 4.12. The average wait time can be seen in Table 4.13. Lastly the average utility of each aircraft for each booking list can be seen in Table 4.15.

The full schedules can be found in Appendix D. An example of a schedule produced by this method can be seen in Table 4.14.

Table 4.10: Final Costs for constrained tests using Model B.

	Short Run		**Long Run**	
Schedule	**Monte Carlo**	**Deterministic**	**Monte Carlo**	**Deterministic**
S10	R 10 279,43	R 9 942,19	R 9 942,19	R 9 942,19
S39	R 25 733,83	R 25 355,67	R 24 142,95	R 23 671,74
S40	R 46 824,43	R 42 780,11	R 37 080,99	R 40 352,49
S99	R 86 820,71	R 79 577,76	R 78 810,76	R 74 804,91
S102	R 90 946,13	R 74 416,20	R 76 423,58	R 73 359,65
S139	R 108 975,96	R 108 482,41	R 107 230,09	R 102 267,51
S200	R 128 980,51	R 123 074,85	R 125 125,73	R 114 259,34

Table 4.11: Percentage improvements and deviations from respective benchmarks using Model B.

	Short Run		**Long Run**		
Schedule	**Monte Carlo**	**Deterministic**	**Monte Carlo**	**Deterministic**	**Benchmark**
S10	-19,9	-22,5	-22,5	-22,5	Upper Bound
S39	23,4	21,6	15,8	13,5	Upper Bound
S40	35,6	23,9	7,4	16,9	Upper Bound
S99	-9,9	-17,4	-18,2	-22,4	Upper Bound
S102	43,7	17,6	20,8	15,9	Upper Bound
S139	120,8	119,8	117,3	107,2	Manual
S200	57,8	50,6	53,1	39,8	Manual

4.7 Analysis of results

This section covers the analysis of the test results seen in section 4.5 and section 4.6. The performance of the model was based on the percentage error from the respective

Table 4.12: CPU time for all test problems in seconds using Model B.

	Short Run		**Long Run**	
Schedule	**Monte Carlo**	**Deterministic**	**Monte Carlo**	**Deterministic**
S10	107,2	127,5	653,5	1031,3
S39	109,1	77,7	1017,1	691,6
S40	137,7	194,3	1728,4	2083,5
S99	347,7	340,7	3634,7	3634,2
S102	608,6	551,3	2969,3	7951,8
S139	204,3	204,3	5357,0	4922,1
S200	848,6	648,0	6976,0	6577,1

Table 4.13: Average customer wait times using Model B.

	Short Run		**Long Run**	
Schedule	**Monte Carlo**	**Deterministic**	**Monte Carlo**	**Deterministic**
S10	0,45	0,51	0,51	0,51
S39	2,09	1,11	2,30	1,47
S40	0,48	0,36	0,38	0,61
S99	1,74	1,54	1,87	1,71
S102	0,77	0,71	0,78	0,63
S139	1,94	1,57	2,04	1,64
S200	2,10	2,15	1,76	1,93

Table 4.14: Final schedule of a booking list with 39 requests using Model B.

Type	**Origin**	**Departure**	**Dest.**	**Arrival**	**Cap. at Origin**	**Groups**	
2	11	390	7	431	4	20	21
	7	431	4	487	5	4	21
	4	487	3	547	5	3	4
	3	610	7	705	4	2	1
	7	705	11	746	5	11	2
	11	746	7	787	0		
	7	865	4	921	5	5	7
	4	921	7	977	0		
	7	977	8	1024	5	8	10
	8	1024	10	1057	4	10	
1	7	870	17	909	6	12	
2	8	775	4	813	1	17	
2	4	390	10	430	0		
	10	430	7	457	5	18	
1	7	870	8	910	7	9	
2	7	390	26	423	0		
	26	423	3	481	8	23	
2	4	390	8	428	0		
	8	660	17	684	1	15	
2	8	660	8	670	2	14	
	8	670	11	688	4	16	14
	11	688	10	717	2	16	
	10	717	7	743	2	19	
2	7	390	26	429	4	13	6
	26	429	17	477	3	6	
	17	660	4	712	5	22	6

Table 4.15: Utility of each aircraft under deterministic long run tests using Model B.

	S10	S39	S40	S99	S102	S139	S200
Utility	56,0%	61,5%	45,3%	65,5%	56,0%	66,8%	62,3%

benchmarks, the mean results the models produced and the time taken to solve the problems. The average waiting time of customers is examined and lastly the effect of the Monte Carlo processes on the model performance is evaluated.

Figure 4.11 shows the percentage deviation from the respective benchmarks. What is interesting to note is that the ABSM performed well in small instances of the DAFP. It showed below benchmarked results for the booking lists containing 10 requests by 22%, the only way this can be explained is that the ABSM uses time as a continuous variable and does not include time discretions that the mixed integer program used by Campbell. This means that better time nodes were found during the simulation [7].

These results are impressive and hold weight when arguing for the method's use in practical instances. Even though the models did not produce results that were optimal the results are within a range that could be considered acceptable from the point of view of the business. The model did not perform as well when applied to the larger instances of the problem, booking list 139 was the worst performer only managing a solution error of 107%. Booking list 139 shows there is a need for a more dynamic method of routing, possibly one that includes learning mechanisms.

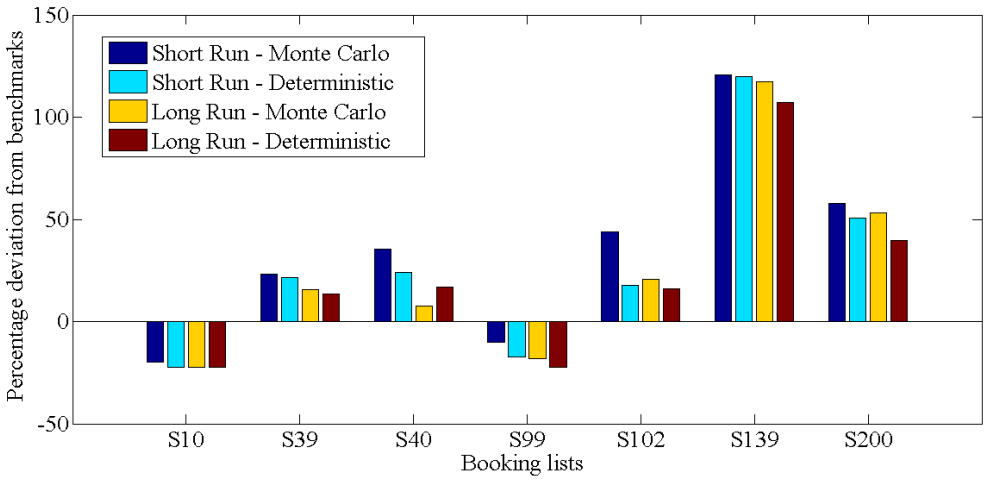

Figure 4.11: Percentage error of respective schedules against benchmarks.

In light of the best solutions produced by the models in this research, a reduction in the number of individual requests reduced the mean solutions. This means that in future work, the aircraft should seek to amalgamate groups more effectively, shortening respective time windows if necessary in order to reduce the complexity of the problem.

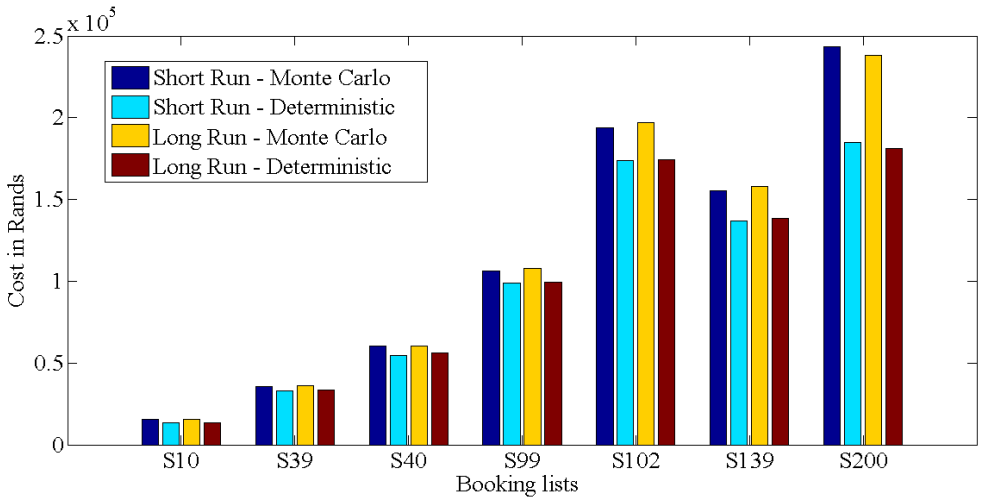

Figure 4.12: Mean result of each schedule under each test.

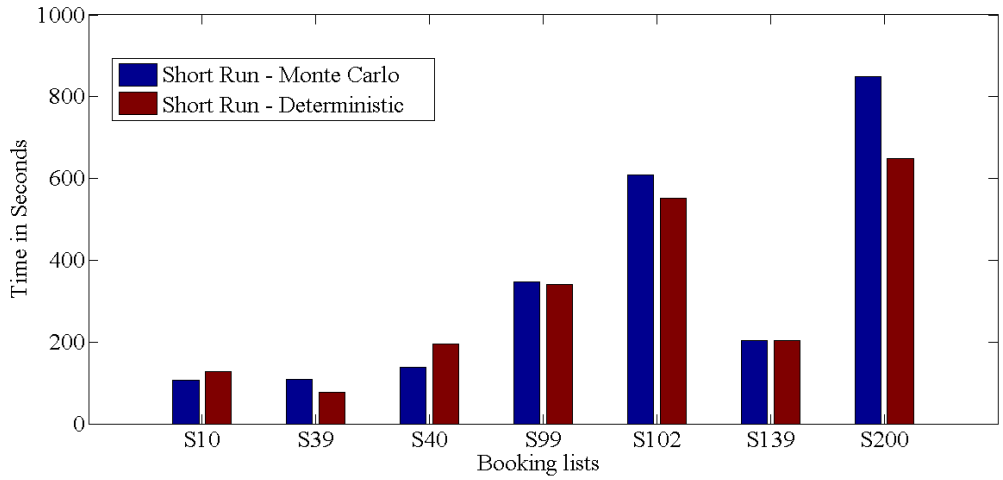

Figure 4.13: CPU time of each short run test.

The CPU times for each test are presented in Figure 4.13 and Figure 4.14. Both figures show that the time taken to solve each problem increases with the size of the schedule, with the exception of S139. S139 results in a lower final cost, suggesting that the customer requests are similar to each other meaning that more groups can be amalgamated into one effectively making the problem size smaller than S102. Since the problem can be reduced in size the solving time should be less than S102.

Even though customer waiting times did not form part of the cost function for the model it was examined. The amount of time a customer waits on the ground for an aircraft (from the early departure time till the time the aircraft picks up the passenger) is a factor that contributes to the quality of service. These values were measured for all tests and the resulting waiting times were graphed in Figure 4.15. The best schedules

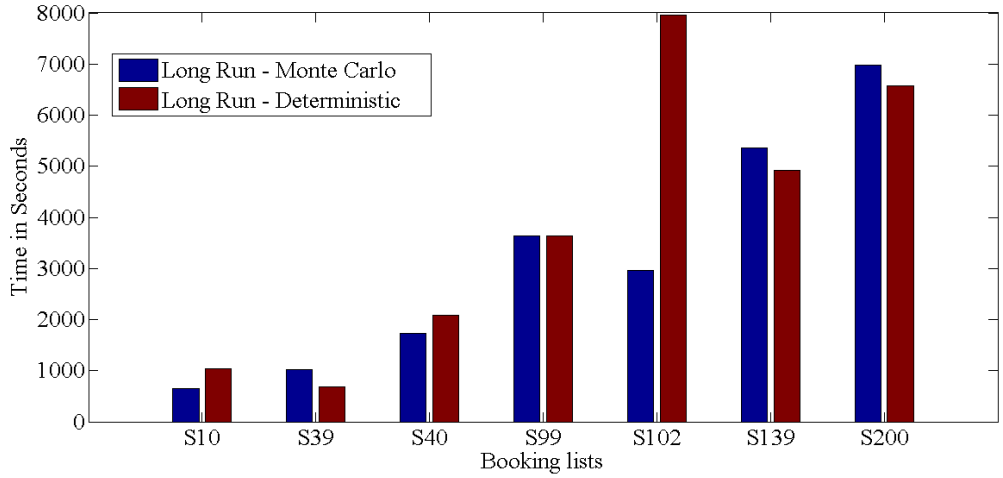

Figure 4.14: CPU time of each long run test.

for all tests produced similar waiting times in this research. The models developed in this research did not focus on minimising these waiting times. A reduction in waiting times could result in a rise in operational costs. This is because if every passenger transit has to be minimised, some of the groups will not be able to be merged due to their differences in early departure times and late arrival times. This means the aircraft will have to make more flights in order to service all of the customers.

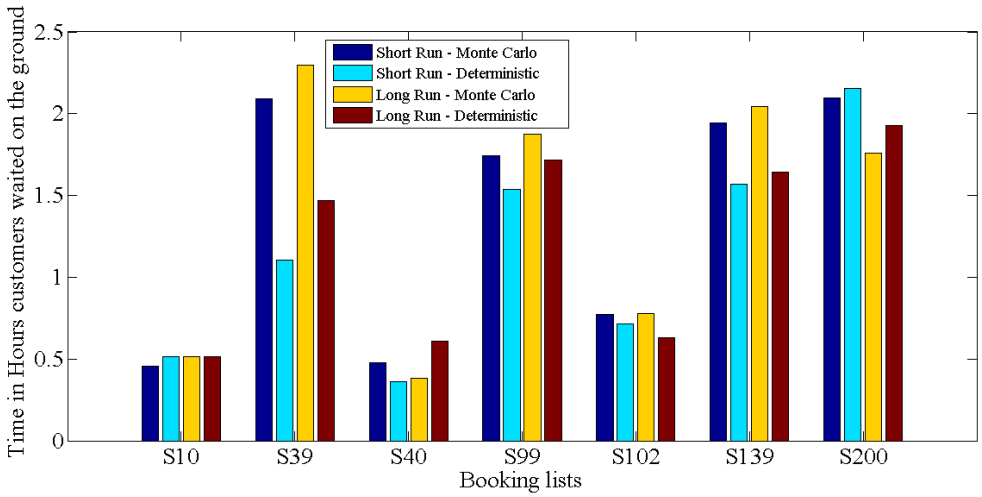

Figure 4.15: Customer wait time for each schedule.

Figure 4.16 and Figure 4.17 shows the normal distributions for schedules 10, 139 and 200 respectively. It shows the distributions of the short and long runs with and without Monte Carlo simulation. They all show a similar trend. The tests that included Monte Carlo simulations showed that the models search the space more comprehensively, since the range of results the model returned were much wider. However, they have a lower

probability of finding lower costs, this is not what is expected when searching for low costs. Whereas the tests without Monte Carlo produce a better chance of finding low-cost results as the range of results are smaller and have higher probabilities of returning lower costs. This result is also evident in Figure 4.11. The long run tests without the use of Monte Carlo simulation produced the best results, the longer run time with the greater chance of finding lower costs all indicate the success of the method.

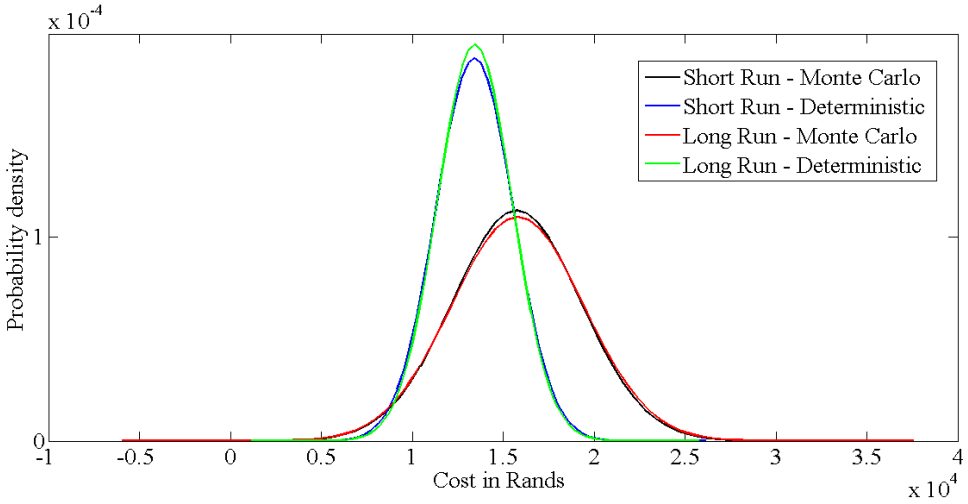

Figure 4.16: Normal distribution of all tests for schedule 10.

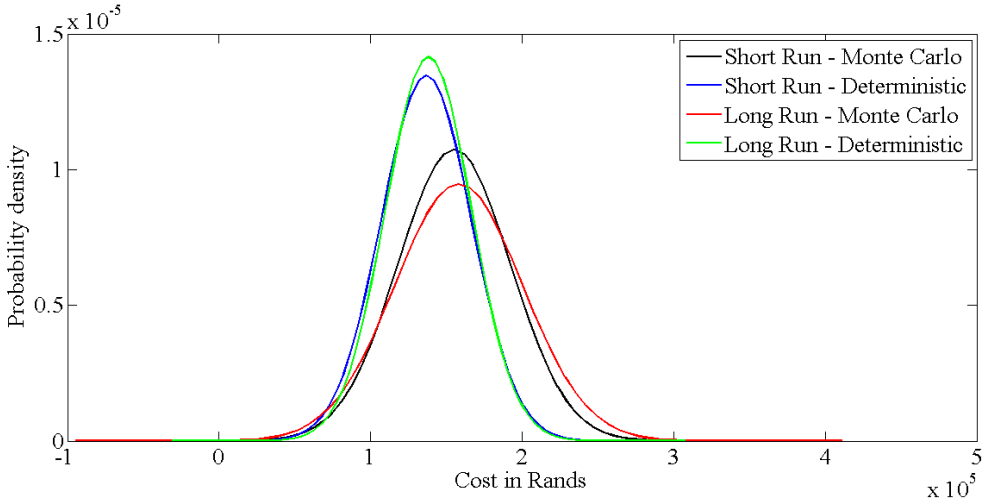

Figure 4.17: Normal distribution of all tests for schedule 139.

4.8 Summary of results

Table 4.16: Summary of results of Model A and Model B.

Model	Schedule	Benchmark (R)	Short Run Results		Long Run Results		Best Result (R)	Deviation %
			Monte Carlo (R)	Deterministic (R)	Monte Carlo (R)	Deterministic (R)		
A	S10	12826,00	11452,77	11420,99	10732,37	**10369,94**	10369,94	-19,15
	S39	20856,00	23873,60	23663,26	22676,56	**22077,02**	22077,02	5,85
	S40	34531,00	35074,25	33945,87	32801,76	**32349,11**	32349,11	-6,32
	S99	96395,00	79758,02	78001,81	77705,33	**74630,80**	74630,80	-22,58
	S102	63270,00	73059,51	74150,45	73092,28	**70241,03**	70241,03	11,02
	S139	49346,00	125929,77	109453,49	124632,26	**104822,03**	104822,03	112,42
	S200	81740,00	133433,60	112164,08	130983,24	**112008,26**	112008,26	37,03
B	S10	12826,00	10279,43	9942,19	9942,19	**9942,19**	9942,19	-22,48
	S39	20856,00	25733,83	25355,67	24142,95	**23671,74**	23671,74	13,50
	S40	34531,00	46824,43	42780,11	37080,99	40352,49	37080,99	7,38
	S99	96395,00	86820,71	79577,76	78810,76	**74804,91**	74804,91	-22,40
	S102	63270,00	90946,13	74416,20	76423,58	**73359,65**	73359,65	15,95
	S139	49346,00	108975,96	108482,41	107230,09	**102267,51**	102267,51	107,25
	S200	81740,00	128980,51	123074,85	125125,73	**114259,34**	114259,34	39,78

5 Discussion

The ABSM design principals used to model the DAFP proved to be effective. The aircraft in the system where able to service all the demand from all of the booking lists. One pass vertical layering was the obvious choice in that it could incorporate the defined rules of the DAFP, such as never exceeding the capacities of the different aircraft and selection of suitable destinations. It also allowed the integration of utility functions to further assist the aircraft in deciding which of the available destinations to fly to as well as which of the available passengers to pick up. The down side of using this method is that the aircraft have no hind sight, so that any decision they make is final for the duration of the simulation. Weighted utility functions have to be optimised in order to ensure the best choices can be made. These are two drawbacks to the model structure, and could possibly be dealt with using two pass vertical layering. This means every decision the aircraft makes will have to be checked and the aircraft can take a retrospective approach to decision making. This way before an aircraft makes a final decision, it can contemplate the effectiveness of the choice.

In this research the program architecture only makes use of two types of routing agent, effectively the aircraft and a selection agent, used to assign which aircraft makes the next move. One of the problems encountered in the simulations was the long waiting times passengers endured on the ground (see Table 4.13 and Figure 4.15). An idea to mitigate this is to design a passenger agent that can track and monitor the amount of time the passenger waits. These agents should be able to communicate with both the selection agent and the aircraft in the system to ensure that they are fetched between the earliest departure time and latest arrival time. Even though the current model was able to fetch and drop off all passengers within the given time windows it is important to note that some passengers may require urgent pick ups and drop off. These are especially true for passengers travelling directly to an airport, to catch a connecting flight. The airlines staff members on the other hand would not require an urgent pick

up or drop off, unless they are needed on another flight, and so a passenger agent could build all of these characteristics within their framework.

MatLab provided an adequate platform for the models, however the programming itself could be optimised to reduce the run times of the simulations. What was noted was that as the problem sizes increase there was also an increase in CPU time (see Figure 4.13 Figure 4.14). The solving times for the booking lists larger than 99 requests are still higher than that of other heuristic techniques applied to VRP's taking between one and two hours to solve where literature suggests a solving time of between half an hour one hour for 50 requests [18]. The main contributor is that the models require multiple iterations to reach good solutions. This can only be resolved by implementing optimisation heuristics within the agent structure, such as the models developed by Barbucha [3] [2] and Certicky [47]. A more practical method would be to modify the current technique in such a way that it can take advantage of a swarm optimisation technique such as ant colony optimisation [11].

Model B's performance is dependent on the weights of the decision parameters. Because of this, the use of the GA in conjunction with the utility testing were crucial aspects of the research. This method proved to be effective in setting the weighted parameters for solving the DAFP using ABSM. It provided insight into how the problems parameters change for different booking lists and suggests that more research has to be conducted to further optimise the weights. The results showed that when the models are adjusted to behave deterministically, the solution qualities are competitive compared to models developed by Barbucha which was 5% mean relative error when applying ABSM to the VRP [3]. This method has also opened the door to research into full optimisation using the method to dynamically change the factors of influence as the agent based model solves the DAFP. This was not done as it would mean the general structure of both models would have to be changed and was out of the scope of this research. This method could potentially be used to not only affect the way the aircraft picks up passengers and flies to new locations but also influence the way aircraft are selected. This, coupled with the agent model ability to solve problems in a distributive manner, could prove to be valuable. Other methods such as simulated annealing and tabu search could also be incorporated and could be used in further research to solve the DAFP. This could be achieved by setting the decision weights as the objective function, similar to the GA, and allowing the heuristic to solve for the minimised weights using the agent model as a fitting function.

Once the GA minimised the deterministic model, each of the utility functions were tested using the optimised weights. From these tests the most successful function was the exponential function. This conclusion, similar to that found in research done by Campbell is that the exponential function outperformed the Gaussian and linear functions in most test cases [7]. Figure 4.4 shows the results profile using Model B to solve for a booking list S99. The exponential and Gaussian function outperform the linear function in Model B testing Table 4.5. What is interesting to note was that when the same booking list was applied to the MCDA method in Model A, all the utility functions performed equally well with the linear function outperforming the other functions on two occasions, the Gaussian on two occasions and the exponential on three occasions (see Table 4.5). The reason the Gaussian and exponential functions outperformed the linear function could be due to the fact that they degrade at a faster rate, where the linear function degrades at a constant rate. This fact means that even when prospective passengers or locations are similar in attractiveness, they can be distinguished more effectively based on the more attractive parameter of the decision.

The time-space network (Figure 4.10) produced for schedule 39 gives an indication whether Model A was successful in finding a feasible solution to the problem. Each coloured line in the network represents a different aircraft, the blue plus markers ('+') indicate early departure times and the red circles indicate late arrival times. These graphs show that no aircraft exceeds the amount of time given in the day to complete all the requests. They are also useful when comparing the manual schedules to the schedules generated by the agent based model. One can also see that the aircraft do not only have the option to pick up and drop off passengers but can also idle on the ground if necessary. This means that the models are performing according to their designed reactivity (Figure 3.5). The geographic illustration of the flight paths (Figure 4.9) can be used to determine the cities any given aircraft will service for that schedule (see Figure 4.9). In this plot each aircraft is assigned a colour and the plus ('+') markers represent cities. The frequency distribution gives the users an idea of how successful the model was in solving the problem. In Figure 4.8 there is only a 5 out of 100 chance of the model producing the best solution of between R23600 and R24000. The figure shows that there is only a 10% chance that the model will beat the upper bound solution to the problem. This result is completely dependent on the size of the booking list. Larger booking lists produce frequency distributions with low probabilities for results near the upper bound of the problem.

Model B produces competitive results and can generate schedules with the flight leg constraint with reasonable costs see Table 4.10. The percentage deviation from the benchmarks (see Table 3.4) can be seen in Table 4.11, where the model outperforms the upper bounds for S10 and S99, while coming close to the upper bounds for booking lists S39, S40 and S102. The reason the ABSM outperforms the benchmark solution for the booking list with 10 requests, is that the mixed integer program relies on 10 minute time discretions and so there may be better solutions when time is continuous variable. The larger schedules produce poor results compared to the manual schedules provided by Wilderness air see Table 4.16. There is a negative correlation between the size of the booking list and the quality of the result. CPU times are generally slow taking at lowest 107.2 seconds for booking list 10 and 648.0 seconds for booking list 200 (see Table 4.12) this could be due to unoptimised programming and could be improved by more effective coding techniques.

In some cases Model A outperformed Model B. The direct comparison between these methods is slightly naive since Model B solved the DAFP with flight leg constraints, and model A relaxed this constraint. Both methods produced results that ranged between 22,6% improvement from the upper bounds and 112% deterioration from the manual solutions for all booking lists tested see Table 4.16. These are too inconsistent to be implemented in any real world application unless the booking lists are smaller than 39 requests. The models produce inconsistent results for larger booking lists however they can be applied as a substitute for any manual schedulers. From looking at the schedules produced, four main routing issues can be seen (see Appendix D). These issues can be directly related to the performance of the models.

1. The first is that the aircraft are duplicating flight legs. This is when an aircraft leaves destination A, lands at B, and then flies back to destination A. Unless constrained by the group's time windows this is unnecessary, especially if the aircraft is flying back to the original destination to fetch more passengers.

2. Secondly more than one aircraft will fly the same flight leg, within the same time period. This is not optimal, since instead of boarding all passengers going to a specific destinations, aircraft will split these similar passengers amongst themselves.

3. Third is that the overall productivity of the aircraft are not maximised. This means that the aircraft do not service as many passengers as they could. From Table 4.15 the productivity of each aircraft in each of the tests never breach

66.8%, this means the aircraft on average is only 66.8% full. It suggests that passengers are not being effectively loaded onto each of the aircraft.

4. Lastly, there needs to be a more effective way to minimise the number of aircraft routed, so as to reduce the overall operational cost.

Table 4.9 and Table 4.11 show that the deterministic models produced the best results in all cases except booking list S40 under the constrained testing. The only stochastic element in those tests was the method in which aircraft were selected, therefore making the best local decision at each move makes sense. This eliminated any outcomes where bad local moves lead to further bad moves further in the simulation. The normal distributions seen in Figure 4.16 and Figure 4.17 show that when the Monte Carlo method is applied the search space increases and the aircraft go through a wider selection of solutions. When the models are deterministic however, the normal distribution narrows and the mean result is lessened i.e the result is more effective. This means that the probability of finding better results in the Monte Carlo method is less than in the deterministic version of the problem. This finding is logical as the deterministic model focuses on better decisions and does not allow any poor choices to be made, and has implications on further work in the field.

There are no visible relationships between the best results produced by the models (see Figure 4.11). The results for booking lists of size 10 to 102 produced a range of results between an improvement of 22.6% and deterioration of 16.9% from their respective benchmarks. The booking lists larger than 102 requests produced results 39.78% deterioration from the benchmarks, see Figure 4.11. This speaks to some fundamental issue with the models routing protocols since it is incapable of finding good solutions to the larger instances of the problem. This suggests the search space for larger instances of the problem are not constrained sufficiently by the routing functions. Therefore in order to improve the result for larger schedules, the routing functions have to be more selective of locations and passengers that will produce better results.

The amount of time taken to solve each problem is important to assess its viability as a routing tool. As shown in Figure 4.13 and Figure 4.14 the time taken to solve any given problem is proportional to the number of requests. Zidi [53]found a CPU time of 840 seconds using a MOSA algorithm, 15180 seconds using a GA and 3077 seconds using a tabu search algorithm for the DARP wich consists of 8 vehicles and 108 customer requests. The closest problem to compare these times to was the DAFP

with 102 requests and 14 vehicles. The ABSM models in this dissertation solved the problems in a time of 545.89 seconds for Model A and 551.3 seconds for Model B under short run conditions. This shows that the method is competitive with conventional routing algorithms for schedules that are smaller than 102 requests. The CPU times do deteriorate with the size of the problem, taking up to 1242 seconds to solve S139 and 893 seconds to solve S200 in short run conditions. In long run testing, where the most promising results are found, Model A took 2222,49 seconds to solve the S10 problem and Model B took 1031.3 seconds to solve the S10 problem. The long run testing proved to be uncompetitive with the results presented by Zidi as the algorithm takes longer to reach better solutions than the conventional techniques [53].

6 Conclusions and Recommendation

This dissertation has looked at the application of ABSM in solving variations of the DAFP. Based on the results and discussion from chapter 4 and chapter 5, conclusions are made with regards to the objectives set out in section 1.5. Next a list of possible recommendations for future research are provided.

6.1 Conclusions

Based on the observations and results presented in chapter 4 as well as the discussion from chapter 5, the following conclusions were found:

1. The solutions ranged from a 22,6% improvement to the benchmark to 107.2% above the benchmark. The models were able to meet the objective of the research for booking lists that have between 10 and 102 requests. The larger instances of the problem could therefore not produce results within the expected range of a maximum of 20% as specified in the objectives. This could be due to:
    - The utility functions not being able to make clear distinctions between good choices since there are more options available for each aircraft.
    - Issues identified such as duplicated flight legs, poor utilisation and minimisation of the number of aircraft are exacerbated by the sizes of the booking list.
2. This research shows that significant results can be achieved using ABSM without the use of a complementary heuristic. However there is room for improvement in that expected results for booking list sizes over 102 were not in the expected range.

3. One pass vertical layering proved to be an effective method in structuring the agent model. The structure is easy to code and can accommodate hybrid agency.

4. The most effective utility function was the exponential distribution. Functions that degrade at a fast rate prove to be most useful, but when applied to MCDA models they had little to no effect on the outcomes.

5. Deterministic models produced the best results under long run testing. The testing showed that deterministic models have a higher probability of achieving solutions close to the given benchmarks.

6. In comparison to similar vehicle routing problems with the same number of requests (such as the DARP) the ABSM method when applied to the DAFP proved to have competitive CPU times for problems under short run testing.

7. This research has introduced seven instances of the DAFP (S10, S39, S40, S99, S102, S139 and S200), making use of a heterogeneous fleet between 14 and 18 aircraft and has listed best known solutions in section 4.8.

6.2 Recommendations

One of the major improvement points is that the aircraft in the simulations only make decisions based on the outcomes of the utility functions. A learning function that could assist in the decision making process could be of great value. There are a few options to consider, the first is the use of neural networks. A neural network could take the place of the Monte Carlo simulations within each of the routing functions. The network should take in each of the decision parameters and output the option the aircraft will action. The only issue with implementing this method would be to develop a training set, this could be achieved by setting up test scenarios that the agents could learn from. These test scenarios would have to be MIP solutions to the DAFP. The second option is the use of ant colony optimisation (ACO). ACO is essentially an agent based system, the current model can already deal with the nature of the DAFP in terms of the heterogeneous feet and capacity constraint. The models should be re-designed in such a way that for each simulation the aircraft lay pheromone down and increase the attractiveness of each route. From this we expect to see solutions improve on each iteration. This implementation would not be easy since the DAFP has both routing constraints for each aircraft and aircraft allocation problem, in selecting which type of

aircraft is used. There is still a large possibility for improvement in determining which passengers board which aircraft.

To resolve the issue of passenger waiting time there are a few approaches one could take. The first would be to include the minimisation of the amount of time a passenger takes from the time his or her time window opens to the time he lands at his destination as part of the objectives of the model. This can only be achieved by introducing a new type of agent in the simulation. This agent should represent the groups of passengers on the ground. These groups would have properties such as, if they are actively looking for pick up, if they are running late, how urgent they are and if they are tourists or Wilderness staff. The aircraft and passenger agents would communicate and help the aircraft make better routing decisions based on negotiations between the two parties. This method should improve the way passengers are grouped on board the aircraft as well as reduce the amount of poor solutions.

Apart from the passenger agent, one could also investigate the use of other program structures. Having a two pass vertical layering [42] instead of one pass vertical layering could be useful. This structure involves allowing an agent to make decisions and allowing the agent to reflect on its choices. This retrospective approach will allow an agent to compare solutions in previous moves / simulations and make better choices in context of what the other agents are doing in the system.

References

[1] Archer, JR, Black, AW, and Roy, S (2012) "Analyzing Air Taxi Operations from a System-of-Systems perspective using Agent-based Modeling." West Lafayette: Purdue University pp. 1–15.

[2] Barbucha, D (2012) "Agent-based guided local search." Expert Systems with Applications vol. 39, no. 15, pp. 12032–12045.

[3] Barbucha, D and Je, P (2007) "An Agent-Based Approach to Vehicle Routing Problem." International Journal of Applied Mathematics and Computer Science 4.1 vol. 1, no. 2, pp. 36–41.

[4] Barnhart, C (2006) Handbooks in Operation Research and Management Science: Transportation, 1st Edition. Elsevier Science & Technology.

[5] Bell, JE and Griffis, SE (2010) "Swarm Intelligence: Application of the Ant Colony Optimization Algorithm To Logistics-Oriented Vehicle Routing Problems." Journal of Business Logistics vol. 31, no. 2, pp. 157–175.

[6] Berbeglia, G, Cordeau, JF, Gribkovskaia, I, and Laporte, G (2007) "Static pickup and delivery problems: a classification scheme and survey." Top vol. 15, no. 1, pp. 1–31.

[7] Campbell, I (2013) Construction Heuristics for the Airline Taxi Problem. Phd thesis, University of the Witwatersrand.

[8] Campbell, I (2015) "Mixed interger solutions to dial a flight problem." Email correspondence from the University of the Witwatersrand containing the solutions to booking list S10, S39, S40, S99 and S102.

[9] Cordeau, JF and Laporte, G (2003) "The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms." 4OR - Quarterly Journal of Belgian, French and Italian Operations Research Societies vol. 1, no. 2, pp. 89–101.

[10] Cordeau, JF and Laporte, G (2007) "The dial-a-ride problem: Models and algorithms." Annals of Operations Research vol. 153, no. 1, pp. 29–46.

[11] Dorigo, M and Gambardella, LM (1997) "Ant colonies for the travelling salesman problem." Biosystems vol. 43, no. 2, pp. 73–81.

[12] Dorigo, M and Stützle, T (2004) Ant Colony Optimization. Scituate, MA, USA: Bradford Company.

[13] Dyson, E (2006) Visible Demand: The New Air-Taxi Market, Esther Dyson quarterly report, vol. 2, pp. 1-44. Tech. Rep. 2.

[14] Espinoza, D, Garcia, R, Goycoolea, M, Nemhauser, GL, and Savelsbergh, MWP (2008) "Per-Seat, On-Demand Air Transportation Part I: Problem Description and an Integer Multicommodity Flow Model." Transportation Science vol. 42, no. 3, pp. 263–278.

[15] Espinoza, D, Garcia, R, Goycoolea, M, Nemhauser, GL, and Savelsbergh, MWP (2008) "Per-Seat, On-Demand Air Transportation Part II: Parallel Local Search." Transportation Science vol. 42, no. 3, pp. 279–291.

[16] Francesco, C (2009) "Simulation Framework." In "Modeling collective taxis in a multi-agent traffic simulation framework," vol. 9. Zürich: Swiss Transport Research Conference.

[17] Franklin, S and Graesser, A (1997) "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents." Intelligent agents III agent theories, architectures, and languages vol. 1193, pp. 21–35.

[18] Golden, B, Raghavan, S, and Wasil, E (2008) The Vehicle Routing Problem: Latest Advances and New Challenges, vol. 43 Information Systems Journal. Springer.

[19] Grether, D, Fürbas, S, and Nagel, K (2013) "Agent-based Modelling and Simulation of Air Transport Technology." Procedia Computer Science vol. 19, pp. 821–828.

[20] Harpa, R (2008) "Applications of Multicriteria Analysis in Cotton Mill." Textile Research Journal vol. 78, no. 5, pp. 421–426.

[21] Haupt, RL and Haupt, SE (1998) Practical Genetic Algorithms. 2nd Edition. New York: Wiley.

[22] Hayashi, K (2000) "Multicriteria analysis for agricultural resource management: A critical survey and future perspectives." European Journal of Operational Research vol. 122, pp. 486–500.

[23] Hillier, F and Lieberman, G (2010) Introduction to Operations Research. 9th Edition. McGraw-Hill Higher Education.

[24] Ho, SC and Haugland, D (2009) "Local search heuristics for the probabilistic dial-a-ride problem." OR Spectrum vol. 33, no. 4, pp. 961–988.

[25] Jennings, N and Wooldridge, M (1998) "Applications of intelligent agents." Agent technology pp. 3–28, Springer, Berlin, Heidelberg.

[26] Jennings, NR, Sycara, K, and Wooldridge, M (1998) "A Roadmap of Agent Research and Development." Autonomous agents and multi-agent systems vol. 38, pp. 7–38.

[27] Kalina, P (2012) "Algorithm for Vehicle Routing Problem with Time Windows Based on Agent Negotiation." Proceedings of the 7th Workshop on Agents In Traffic and Transportation, AAMAS .

[28] Knapen, L, Keren, D, Yasar, AUH, Cho, S, Bellemans, T, Janssens, D, and Wets, G (2012) "Analysis of the Co-routing Problem in Agent-based Carpooling Simulation." Procedia Computer Science vol. 10, no. 270833, pp. 821–826.

[29] Kou, G and Wu, W (2014) "Multi-criteria decision analysis for emergency medical service assessment." Annals of Operations Research vol. 223, no. 1, pp. 239–254.

[30] Macal, CM and North, MJ (2008) "Agent-based modeling and simulation: ABMS examples." In "Proceedings of the 2008 Winter Simulation Conference," pp. 101–112.

[31] Maciejewski, M and Nagel, K (2012) "Towards Multi-Agent Simulation of the Dynamic Vehicle Routing Problem in MATSim." Parallel Processing and Applied Mathematics vol. 7204, pp. 551–560.

[32] Máhr, T, Srour, J, de Weerdt, M, and Zuidwijk, R (2010) "Can agents measure up? A comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty." Transportation Research Part C: Emerging Technologies vol. 18, no. 1, pp. 99–119.

[33] Mason, SJ, Hill, RR, Mönch, L, Rose, O, Jefferson, T, and Fowler, JW (2008) "Introduction to monte carlo simulation." In "Proceedings of the 2008 Winter Simulation Conference," pp. 91–100.

[34] Mateo, JRSC (2012) Multi criteria analysis in the renewable energy industry. Springer Science and Business Media.

[35] Mitchell, M (1998) An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press.

[36] Negulescu, SC, Kifor, CV, and Oprean, C (2008) "Ant colony solving multiple constrains problem: Vehicle route allocation." International Journal of Computers, Communications and Control vol. 3, no. 4, pp. 366–373.

[37] NetLogo (2011) "Netlogo." URL `https://github.com/NetLogo/NetLogo, Accessed[9May2014]`.

[38] Pillac, V, Gendreau, M, Guéret, C, and Medaglia, AL (2013) "A review of dynamic vehicle routing problems." European Journal of Operational Research vol. 225, no. 1, pp. 1–11.

[39] Reed, M, Yiannakou, A, and Evering, R (2014) "An ant colony algorithm for the multi-compartment vehicle routing problem." Applied Soft Computing vol. 15, pp. 169–176.

[40] Reeves, CR (1997) "Genetic Algorithms for the Operations Researcher." INFORMS Journal of Computing vol. 9, pp. 231–250.

[41] Rei, W, Gendreau, M, and Soriano, P (2010) "A Hybrid Monte Carlo Local Branching Algorithm for the Single Vehicle Routing Problem with Stochastic Demands." Transportation Science vol. 44, no. December 2014, pp. 136–146.

[42] Salamon, T (2011) Design of Agent-Based Models : Developing Computer Simulations for a Better Understanding of Social Processes. Academic series. Repin, Czech Republic: Bruckner Publishing.

[43] Savelsbergh, MWP and Sol, M (1995) "The general pickup and delivery problem." Transportation Science vol. 29, no. 1, pp. 17–29.

[44] Shah, MM (2012) "Artificial Intelligence : Vehicle Routing Problem and Multi Agent System." International Journal of Computer Applications (IJCA) Artificial pp. 1–3.

[45] Thangiah, SR, Shmygelsaka, O, and Mennell, W (2001) "An agent architecture for Vehicle routing problems." In "Proc 2001 ACM Symposium on Applied Computing," pp. 517–521.

[46] Van**e**k, O, Jakob, M, Hrstka, O, and P**e**chou**c**ek, M (2013) "Agent-based model of maritime traffic in piracy-affected waters." Transportation Research Part C: Emerging Technologies vol. 36, pp. 157–176.

[47] **C**ertický, M, Jakob, M, Píbil, R, and Moler, Z (2014) "Agent-based Simulation Testbed for On-demand Mobility Services." Procedia Computer Science vol. 32, pp. 808–815.

[48] Vidal, J (2014) "Multi-agent models." URL `http://jmvidal.cse.sc.edu/netlogomas/,Accessed[3September2014]`.

[49] Vokínek, J, Antonín, K, and Michal, P (2010) "Agents Towards Vehicle Routing Problems." Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems vol. 1, no. 1, pp. 773–780.

[50] wilderness (2014) "Wilderness air." URL `http://www.wilderness-air.com/about.html,Accessed[2October2015]`.

[51] Wildernesss-Air (2015) "Manual schedules of booking list s139 and s200." Spread sheets provided through email correspondence from the University of the Witwatersrand containing schedule data.

[52] Xu, L and Yang, Jb (2001) "Introduction to Multi-Criteria Decision Making and the Evidential Reasoning Approach." Technical Working Paper , no. 0106, pp. 1–21.

[53] Zidi, I, Zidi, K, Mesghouni, K, and Ghedira, K (2011) "A Multi-Agent System based on the Multi-Objective Simulated Annealing Algorithm for the Static Dial a Ride Problem." Proceedings of the 18th IFAC World Congress, 2011 vol. 18, no. 1, pp. 2172–2177.

Appendix A MatLab Code

A.1 Main code

```
% Date: 26 August 2015
% Author: Daniel Reddy
% Agent Based Simulation of the Dial-a-Flight problem
% Version 9.1 - Constrained Model
%-------------------------------------------------------------------------%
clear
clc
%% Globals
global schedule Factz1 citys fleettype fleet cost LateCost UndelCost LateGroups Per
schedule = xlsread('schedule R200.xlsx',1);
citys = xlsread('data.xlsx',1);
fleettype = xlsread('data.xlsx',2);
fleet = xlsread('Fleet200.xlsx',1);
Factz = xlsread('GeneticR.xlsx',1);
SolnKeeper = []; % Store all solutions here
[SS1,SS2] = size(schedule);
% Store the idle time of each group in this variable
Gidle = zeros(SS1,1);
%% 1st stratergy variables
% set the correct factors from the genetic algoritm
G = 7; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for ff = 1:8
   Factz1(ff,1) = Factz(ff,G);
end
%% 2nd stratergy variables
SFac = [3/3 3/3 3/3]; % Weights for selection
PFac = [5/5 5/5 5/5 5/5 5/5]; % Weights for pick up
FFac = [5/5 5/5 5/5 5/5 5/5]; %Weights for flying
```

```
% Set the nest solution to be infinity
BestSoln = inf;
FRout = {};
%% Set the number of runs
tic
disp('Begining Simulation...')
for run = 1:1000
    %% Set Environment
    [Environment] = PopMyEnvironment;
    %% Set up Agents/Aircraft
    [Agents] = PopMyAgent;
    [~,S1] = size(Agents);
    %% Set Data storage variables
    cost = 0; % Starting cost of the simulation
    Pen = 0; % Starting penelty cost = 0 rands
    LateCost = 100; % Lateness cost of 100 rand per minute
    UndelCost = 10000; % Cost of every undelivered group is 10000 rand
    UndelGroups = []; % All undelivered groups (i.e. not been fetched)
    LateGroups = []; % All the late groups (i.e dropped off late)
    LGTimes = [];
    AgentData = {}; % This is used to store the final routes
    Xkeeper = []; % This is used to store the x and y locations for plotting the ag
    Ykeeper = [];
    %% Set the demand stopping criteria && LAT stopping criteria
    stop = sum(schedule(:,6)); % This is the demand in the system
    MaxLAT = max(schedule(:,3)); %  This is the latest LAT of all the groups
    %% Set General stopping criteria
    simrun = 1; % The simulation runs untill simrum = 0;
    %% Start Simulation
    while simrun ~= 0
        %% Select Agent
        [i] = select(Agents);
        % Update the selection value
        Agents(i).Selection = Agents(i).Selection + 1;
        MoveNum = Agents(i).Selection;
        %% Routing
        if MoveNum == 1
            % Pick up only
            [Agents,Environment] = Pickup2(Agents,Environment,i);
            %[Agents,Environment] = pickup(Agents,Environment,i);
            capin = Agents(i).cap;
            if capin == 0
                % If no passengers are collected then the aircraft should
                % move to a new location
```

```
            Agents(i).Selection = Agents(i).Selection + 1;
            MoveNum = MoveNum + 1;
        end
    end
    if MoveNum > 1
        % F--->P--->D
        % Record the data at the origin node
        Nc = Agents(i).location;
        XX = citys(Nc,2);
        YY = citys(Nc,3);
        Td = Agents(i).Time;
        Cf = Agents(i).cap;
        Gob = Agents(i).gob;
        % Fly
        [Agents,Environment] = fly2(Agents,Environment,i);
        %[Agents,Environment] = fly(Agents,Environment,i);
        % Record the data at the destination
        Nn = Agents(i).location;
        Ta = Agents(i).Time;
        % Join the data
        Route = horzcat(Nc,Td,Nn,Ta,Cf,Gob');
        % Drop off
        [Agents,Environment,stop] = Dropoff2(Agents,Environment,i,stop);
        % Pick up
        [Agents,Environment] = Pickup2(Agents,Environment,i);
        %[Agents,Environment] = pickup(Agents,Environment,i);
        % Store the Leg
        AgentData{MoveNum-1,i} = Route;
        Xkeeper(MoveNum-1,i) = XX;
        Ykeeper(MoveNum-1,i) = YY;
    end
    %% Check the Stopping conditions for Agent
    TAgent = Agents(i).Time;
    if TAgent >= MaxLAT
        Agents(i).Stop = 1;
    end
    %% Check the stopping condition for the Simulation
    % Stop the simulation if all passengers have been droped off
    if stop == 0
        simrun = 0;
    end
    % Stop the simulation if all the Agents occupie a time that is
    % greater than the latest LAT
    for j = 1:S1
```

```
            Timecheck(j,1) = Agents(j).Time;
        end
        FTime = min(Timecheck);
        if FTime > MaxLAT
            simrun = 0;
        end
    end
    %% END simulation
    % Store the cost
    SolnKeeper{run,1} = cost;
    SolnKeeper{run,2} = stop;
    SolnKeeper{run,3} = LateGroups;
    SolnKeeper{run,4} = LGTimes;
    SolnKeeper{run,5} = Pen;
    SolnKeeper{run,6} = stop*UndelCost;
    SolnKeeper{run,7} = SolnKeeper{run,1} + SolnKeeper{run,5} +SolnKeeper{run,6};
    FCost = SolnKeeper{run,1} + SolnKeeper{run,5} +SolnKeeper{run,6};
    FCostKeeper(run,1) = SolnKeeper{run,1} + SolnKeeper{run,5} +SolnKeeper{run,6};
    % Get the Best solution
    if FCost < BestSoln
        BestSoln = FCost;
        FRout = {};
        FRout = AgentData;
        Fx = [];
        Fx = Xkeeper;
        Fy = [];
        Fy = Ykeeper;
        FGidle = [];
        FGidle = Gidle/60;
    end
end
%% Analysis
disp('Disply graphs...')
% Plot the Time space Network
plotmyagents2(FRout)
%plotmyagents(FRout)
% Plot the route taken geographically
Geoplot(Fx,Fy)
% Box and wisker plot of the final costs
figure(3)
subplot(2,2,1)
boxplot(FCostKeeper)
title('Box plot of the output values')
xlabel('Data Set')
```

```
ylabel('Cost in Rands')
% Scatter plot of all the solutions
subplot(2,2,2)
plot(FCostKeeper,'+b')
title('Scatter plot of the output solutions')
xlabel('Itteration')
ylabel('Cost in Rands')
% Pareto of the solutions
Sorted = sort(FCostKeeper,'descend');
subplot(2,2,3)
bar(Sorted)
axis([0 run 0 inf])
title('Pareto chart of output solutions')
xlabel('Itteration')
ylabel('Cost on Rands')
% Plot the normal distribution to the set of data
subplot(2,2,4)
pd = fitdist(FCostKeeper,'Normal');
minval = pd.mu - 6*(pd.sigma);
maxval = pd.mu + 6*(pd.sigma);
Xval = minval:100:maxval;
Y = pdf(pd,Xval);
plot(Xval,Y,'LineWidth',2)
axis([minval maxval 0 inf])
title('Normal distribution of output values')
xlabel('Cost in Rands')
% Frequency distribution
figure (4)
hist(FCostKeeper,15)
%% Report
disp('Report...')
% Schedule number
disp('Schedule R200') % User input %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% time elapsed
toc
% Show the number of itterations
disp('Itterations:')
disp(run)
% Mean Waiting time of passengers
GMEAN = mean(FGidle);
disp('Mean passenger waiting times:')
disp(GMEAN)
% Best solution
disp('Best Solution:')
```

```
disp(BestSoln)
% Display the cost per group
CPG = BestSoln/200; % User input %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('Cost per group:')
disp(CPG)
% Mean solution
disp('Mean:')
disp(pd.mu)
disp('Standard deviation:')
disp(pd.sigma)
% The deviation
disp('Deviation from Optimal:')
Best = 81740; % This must be specified by the user %%%%%%%%%%%%%%%%%%%%%%%%%%
E1 = ((BestSoln-Best)/Best)*100;
disp(E1)
%% Write to Excel
disp('Writing results to file...')
% Create a structured array of the results
Results.Size = 'Schedule R200'; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Results.RunTime = toc;
Results.Itterations = run;
Results.Z = Best;
Results.CPG = CPG;
Results.BestSoln = BestSoln;
Results.Mean = pd.mu;
Results.Sigma = pd.sigma;
Results.Deviation = E1;
names = fieldnames(Results);
for ij = 1:length(names)
    tempData{ij,1} = Results.(names{ij});
end
filename = 'Results.xlsx';
xlswrite(filename,names,1,'A1:A9')
xlswrite(filename,tempData,1,'B1:B9')

T1 = {'cost','stop','Late Groups','Late Time','Late Pens','Undel. Pens','Final Cost
xlswrite(filename,T1,2,'A1:G1')
xlswrite(filename,SolnKeeper,2,'A2')

disp('Process Completed...')
```

A.2 Drop off function

```
% We need a drop off function to ensure the timing of the program is
% correct.

function [Agents,Environment,stop] = Dropoff2(Agents,Environment,i,stop)
global schedule Pen LateGroups LateCost LGTimes
% Agent properties
GOB = Agents(i).gob;
Legs = Agents(i).legs;
Nn = Agents(i).location;
Cc = Agents(i).cap;
Tc = Agents(i).Time;
[s1,~] = size(GOB);
% We have the groups that are going to depart
G2D = Agents(i).depart;
DropoffG = Environment(Nn).DropoffG;
[s2,~] = size(G2D);
if s2 > 0
    for i1 = 1:s2
        g1 = G2D(i1,1);
        LAT = schedule(g1,3);
        c1 = schedule(g1,6);
        Cc = Cc - c1; % Calculate final capacity
        stop = stop - c1; % Remve the groups from the stop counter
        mx1 = find(GOB(:,1) == g1);
        [GOB,~] = removerows(GOB,'ind',mx1); % Remove the group from the aircraft
        [Legs,~] = removerows(Legs,'ind',mx1);
        % Remove the group from the drop off points
        mx2 = find(DropoffG(:,1)==g1);
        [DropoffG,~] = removerows(DropoffG,'ind',mx2);
        % Calculate the penalties for late delivery
        if Tc > LAT + 10
            lateness = Tc - LAT;
            Pen = Pen + lateness*LateCost;
            [s3,~] = size(LateGroups);
            LateGroups(s3+1,1) = g1;
            LGTimes(s3+1,1) = lateness;
        end
    end
end
Agents(i).gob = GOB;
```

```
Agents(i).legs = Legs;
Agents(i).cap = Cc;
Agents(i).depart = [];
end
```

A.3 Fly function

```
% Here we are introducting a new fly function that will allow an agent to
% remain idle

function [Agents,Environment] = fly2(Agents,Environment,i)
global schedule citys cost Factz1
% The current node of the agent i  s given by
Nc = Agents(i).location;
x1 = citys(Nc,2);
y1 = citys(Nc,3);
speed = Agents(i).speed;
% The agents capcity is given as
Cc = Agents(i).cap;
% The maximum capcity of the agent is given as
Cmax = Agents(i).maxcap;
% The current time of the agent is given as
Tc = Agents(i).Time;
% The groups on board the agent are given as
GOB = Agents(i).gob;
[s1,~] = size(citys);
Attract = zeros(s1,6);
Legs = Agents(i).legs;
[SLeg,~] = size(Legs);
if SLeg > 0
    MaxLeg = max(Legs);
else
    MaxLeg = 0;
end
[s3,~] =size(GOB);
mustgo = zeros(s1,1); % This is the variable that tells us if we have to go to a sp
Destgob = [];
for i3 = 1:s3
    g2 = GOB(i3,1); % This looks at all of the groups on board
    Destgob(i3,1) = schedule(g2,5); %List of destinations of the groups
    dest1 = Destgob(i3,1);
```

```
    x2 = citys(dest1,2);
    y2 = citys(dest1,3);
    distance  = pdist([x1 y1;x2 y2]);
    ArrivalTime1 = Tc + (distance/speed)*60;
    TU = abs(schedule(g2,3) - ArrivalTime1); %Gives the time urgencgy of the groups
    TU1 = 1/exp(TU/Factz1(1,1));
    Attract(dest1,2) = Attract(dest1,2) + TU1;
    % Calculate if the group must be delivered
    % Mustgo(Dest,1)
    if TU<TU1*Factz1(9,1);
        mustgo(dest1,1) = 1;
    end
end

% Add attractivness for each move
for i1 = 1:s1
    Nn = i1; % Nn = location under examination
    % Check if we can go to this destination and still drop all our
    % passengers off.
    x2 = citys(Nn,2);
    y2 = citys(Nn,3);
    distance = pdist([x1 y1;x2 y2]);
    Attract(i1,1) = 1/exp(distance/Factz1(2,1));
    ArrivalTime = Tc + (distance/speed)*60;
    % Get the groups on ground for each reqquest
    GOG = Environment(Nn).PickupG;
    [s2,~] = size(GOG);

    for i2 = 1:s2
        g1 = GOG(i2,1); % This will look at all the groups on the ground
        dest1 = schedule(g1,5);
        Destgog(i2,1) = dest1;
        % Calculate the time closeness of the groups
        TC = abs(schedule(g1,2) - ArrivalTime);
        TC1 = 1/exp(TC/Factz1(3,1));
        Attract(Nn,3) = Attract(Nn,3) + TC1;
        % can the group fit on the aircraft
        Cg = schedule(g1,6);
        if Cc + Cg <= Cmax
            Cap = Cg*Factz1(4,1);
            Attract(Nn,4) = Attract(Nn,4)+Cap;
        end
        % Number of OB passengers sharing desinations with OG passengers
        if s3 > 0
```

```
            mx2 = find(Destgob(:,1) == dest1);
            [s5,~] = size(mx2);
            Attract(Nn,6) = Attract(Nn,6) + s5/Factz1(5,1);
        end
    end
    if s3 > 0
        % Number of passengers going to Nn
        mx1 = find(Destgob(:,1) == Nn);
        [s4,~] = size(mx1);
        Attract(Nn,5) = s4/Factz1(6,1);
    end
end
Fac = [1;1;1;1;1;1];
% Atrract table - [distance, TUgob,TCgog,Cap, Gob2Dest, A2Gmatch]
Attract1 = Attract*Fac;
% Attract =========> AttractF
% Apply logical filter
% Give zero attractivness to no PU and no DO points
for i4 =1:s1
    PU = Environment(i4).PickupG;
    DO = Environment(i4).DropoffG;
    [pu,~] = size(PU);
    [do,~] = size(DO);
    if pu == 0 && do==0
        % If there are no pick ups or drop offs set attractivness to zero
        Attract1(i4,1) = 0;
    end
    true = 0;
    for i6 = 1:s3
        d1 = Destgob(i6,1);
        if d1 == i4
            true = 1;
        end
    end
    if true == 0 && pu == 0
       Attract1(i4,1) = 0;
    end
end
% The agent should  drop off if Cc == Cmax only look at the DO points
if Cc == Cmax
    for city = 1:s1
        true = 0;
        for i5= 1:s3
            d1 = Destgob(i5,1);
```

```
            if d1 == city
                true = 1; % Do not remove the attractivness
            end
        end
        if true == 0
            Attract1(city,1) = 0;
        end
    end
end
% Can we fly to each of the remaining locations and still drop off the
% groups on board before their LATs?
mx3 = find(Attract1 > 0);
[s6,~] = size(mx3);
for i6 =1:s6
    c1 = mx3(i6,1); % The city under consideration
    pu1 = Environment(c1).PickupG;
    [SJ1,~] = size(Destgob);
    if SJ1>0
        mxd = find(Destgob(:,1) == c1);
        [sj,~] = size(mxd);
    else
        sj = 0;
    end
    [s7,~] = size(pu1);
    y = 0;
    true = 0;
%       true2 = 0;
    for i7 = 1:s7
        Gn = pu1(i7,1); % Group under consideration
        % Calculate the arrival time at this location
        Di1 = schedule(Gn,4);
        xi1 = citys(Di1,2);
        yi1 = citys(Di1,3);
        DIS = pdist([x1 y1;xi1 yi1]);
        Tarr = Tc + (DIS/speed)*60;
        % This constraint stops the plane from fetching a group of
        % passengers that are greater than 30min in the future.
%         if Tarr <= 30 + schedule(Gn,2)
%            true2 = 1;
%         end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [y] = checkdrop2(Tarr,GOB,c1,Gn,speed);
        if y == 1
            true = 1;
```

```
        end
    end
    if s7 == 0
        true = 1;
%         true2 = 1; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
    if sj > 0
        true = 1;
%         true2 = 1; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
    if true == 0
        Attract1(c1,1) = 0;
    end
%     if true2 == 0
%         Attract1(c1,1) = 0; % Apply the 30min delay constraint %%%%%%%%%%%%
%     end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% If the max Leg counter has been activated then the aircraft must
% priorities that specific request.
% This prevents any passenger stayig on the aircraft for longer than 3
% flight legs.
if MaxLeg >= 2
   ml1 = find(Legs(:,1) == 3);
   ml11 = max(ml1);
   gl1 = GOB(ml11,1);
   destgl1 = schedule(gl1,5);
   mustgo(destgl1,1) = 1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check the must go variable
mg1 = find(mustgo(:,1) == 1);
[s8,~] = size(mg1);
if s8 == 0
    on = 0; % 0 for no monte carlo, 1 to activate the monte carlo %%%%%%%%%
    [Nn]=MonteCarlo(Attract1,on);
    x2 = citys(Nn,2);
    y2 = citys(Nn,3);
    distance  = pdist([x1 y1;x2 y2]);
    ArrivalTime1 = Tc + (distance/speed)*60 + 10;
else
    Nn = mg1(1,1); % has to fly to this destination
    x2 = citys(Nn,2);
    y2 = citys(Nn,3);
```

```
    distance  = pdist([x1 y1;x2 y2]);
    ArrivalTime1 = Tc + (distance/speed)*60 + 10;
end

som = sum(Attract1);
if som == 0
    Nn = Nc;
    x2 = citys(Nn,2);
    y2 = citys(Nn,3);
    distance  = pdist([x1 y1;x2 y2]);
    ArrivalTime1 = Tc + (distance/speed)*60 + 10;
end
% Update the next destination
Agents(i).location = Nn;
% Update the x and y coordinates
Agents(i).x = citys(Nn,2);
Agents(i).y = citys(Nn,3);
% Update the time of arrival and add the TOT
Agents(i).Time = ArrivalTime1;
% Update the departure box
if s3 > 0
    gdo = find(Destgob(:,1) == Nn);
    [s9,~] = size(gdo);
    for i8 = 1:s9
        index = gdo(i8,1);
        gdp = GOB(index,1);
        depart(i8,1) = gdp;
    end
    if s9 > 0
        Agents(i).depart = depart;
    end
end
% Calculate the cost of this move and add it to the cost counter
IFC = Agents(i).cost;
COM = (distance/speed)*IFC ;
cost = cost + COM;

% Add a leg on each of the groups on board
if Nn ~= Nc
    [s100,~] = size(Legs);
    for j1 = 1:s100
        Legs(j1,1) = Legs(j1,1) + 1;
    end
    Agents(i).legs = Legs;
```

```
end
end
```

A.4 Pick up function

```
% This is the pick up function

function [Agents,Environment] = Pickup2(Agents,Environment,i)
global schedule Gidle Factz1
Nc = Agents(i).location;
Tc = Agents(i).Time;
Cc = Agents(i).cap;
GOB = Agents(i).gob;
GOG = Environment(Nc).PickupG;
Cmax = Agents(i).maxcap;
speed = Agents(i).speed;
Legs = Agents(i).legs;
[a1,~] = size(GOG);
booked = [];
TOT = Tc;
while a1 ~= 0

    % Get the destinations of the groups on board
    [s2,~] = size(GOB);
    Destgob = [];
    for i2 =1:s2
        g2 = GOB(i2,1);
        Destgob(i2,1) = schedule(g2,5);
    end
    [s1,~] = size(GOG);
    Attract = zeros(s1,2);
    mintot = inf;
    for i1 = 1:s1
        g1 = GOG(i1,1);
        d1 = schedule(g1,5);
        Cg = schedule(g1,6);
        edt = schedule(g1,2);
        if edt < inf
            mintot = edt;
        end
        if s2>0
```

```
            mx1 = find(Destgob(:,1) == d1);
            [s3,~] = size(mx1);
            Attract(i1,1) = s3*Factz1(7,1); % Factor
        end
        Attract(i1,2) = Cg/Factz1(4,1); % Factor
        y1 = 1;
        if Cg + Cc > Cmax
            Attract(i1,1) = 0;
            Attract(i1,2) = 0;
            y1 = 0;
        end
        % This constraint stops the plane from fetching a group of
        % passengers that are greater than 30min in the future.
%         if TOT <= 30 + schedule(g1,2)
%             y1 = 0;
%         end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if y1 == 1
            [y] = checkdrop2(TOT,GOB,Nc,g1,speed);
        else
            y = 0;
        end
        if y==0
            Attract(i1,1) = 0;
            Attract(i1,2) = 0;
        end
    end
    Attract1 = Attract*[1;1];
    isit = sum(Attract1);
    % Calculate the chance of leaving
    Twait = mintot-TOT;
    x =1-1/exp(Twait/Factz1(8,1)); % Factor
    if x < 0.01
        x = 0.01;
    end
    true = 0;
    if isit > 0 %Picking group up
        on = 0;
        [choice]=MonteCarlo(Attract1,on);
        Gc = GOG(choice,1);
        % Add the group to the booked vector
        booked = vertcat(Gc,booked);
        % Remove the chosen group from GOG
        mx1 = find(GOG(:,1) == Gc);
```

```
            [GOG,~] = removerows(GOG,'ind',mx1);
            % Add the chosen group to GOB
            GOB = vertcat(Gc,GOB);
            Legs = vertcat(0,Legs);
            % add to the capcity of the craft
            Cc = Cc + schedule(Gc,6);
            true = 1;
            edt1 = schedule(Gc,2);
            % Move the take off time for each of the groups collected
            if edt1>TOT
                TOT = edt1;
            end
            Gidle(Gc,1) = Tc - edt1; % Stores the idel times of each of the passengers
            if Gidle(Gc,1) < 0
                Gidle(Gc,1) = 0;
            end
        end
        a1 = a1 - 1;
        % Ask to leave
        r = rand;
        if r<x && true ==1
            a1 = 0;
        end
    end

    % Update the variables
    Agents(i).gob = GOB;
    Agents(i).legs = Legs;
    Agents(i).booked = booked;
    Agents(i).cap = Cc;
    Agents(i).Time = TOT;
    Environment(Nc).PickupG = GOG;

    end
```

A.5 GA Code

```
% This is the genetic algorithm that will support the optimisation process.
% It will take the initial set of weights for each of the 8 factors of
% infuence and manipulate them to give an optimal set of results.
% Author: Daniel Reddy
% Date: 11 October 2015
%%
clc
clear
global Facz schedule citys fleettype fleet
schedule = xlsread('schedule R200.xlsx',1);
citys = xlsread('data.xlsx',1);
fleettype = xlsread('data.xlsx',2);
fleet = xlsread('Fleet200.xlsx',1);
Zf = inf;
Fxii = 3;
%% Generate the initial population
for i = 1:10 % Calculate for 10 chromosomes
    C1(1,1) = 1 + (100-1)*rand;
    C1(2,1) = 1 + (100-1)*rand;
    C1(3,1) = 1 + (100-1)*rand;
    C1(4,1) = rand;
    C1(5,1) = 1 + (10-1)*rand;
    C1(6,1) = 1 + (10-1)*rand;
    C1(7,1) = rand;
    C1(8,1) = 1 + (50-1)*rand;
    C1(9,1) = 15 + (45-15)*rand;
    Facz = C1;
    CKeep(:,i) = C1;
    [ZLow,FCostKeeper,SolnKeeper] = ABSM(Fxii);
    ZmKeep(1,i) = ZLow;
end
%% Main Algorithm
% Select the five best and sample 4 of the five
disp('Begining Main Algorithm...')
tic
for run = 1:300
    ZmKeep1 = ZmKeep';
    ZmKeep1  = sort(ZmKeep1,'ascend');
    % Store the initial 7 best solutions and chromosomes
    for i1 = 1:7
```

```
    mxi = find(ZmKeep1(i1,1) == ZmKeep(1,:));
    mx1 = min(mxi);
    ZmKeep2(1,i1) = ZmKeep(1,mx1);
    Chromo1(:,i1) = CKeep(:,mx1);
end

% Get 5 best and 5 worst
for i2 = 1:5
    BestKeep(i2,1) =  ZmKeep1(i2,1); % Best solutions
end
for i3 = 6:10
    WorstKeep(i3-5,1) = ZmKeep1(i3,1); % Worst solutions
end

on = 1; % Turns the monte carlo function on

% Sample 4 of the 5 best solutions
p(1,1) = MonteCarlo(BestKeep,on);
p(1,2) = MonteCarlo(BestKeep,on);
while p(1,1) == p(1,2)
    p(1,2) = MonteCarlo(BestKeep,on);
end
p(1,3) = MonteCarlo(BestKeep,on);
while p(1,1) == p(1,3)
    p(1,3) = MonteCarlo(BestKeep,on);
end
while p(1,2) == p(1,3)
    p(1,3) = MonteCarlo(BestKeep,on);
end
p(1,4) = MonteCarlo(BestKeep,on);
while p(1,1) == p(1,4)
    p(1,4) = MonteCarlo(BestKeep,on);
end
while p(1,2) == p(1,4)
    p(1,4) = MonteCarlo(BestKeep,on);
end
while p(1,3) == p(1,4)
    p(1,4) = MonteCarlo(BestKeep,on);
end
% Sample 2 of the 5 worst solutions
p(1,5) = MonteCarlo(BestKeep,on);
p(1,6) = MonteCarlo(BestKeep,on);
while p(1,5) == p(1,6)
    p(1,6) = MonteCarlo(WorstKeep,on);
```

```
end
% pare the parents up
Pmatch = randperm(6);
for i4 = 1:6
   t1 = Pmatch(1,i4);
   if t1>=5
        s1 = p(1,t1);
        Zm1 = WorstKeep(s1,1);
        Zmx = find(ZmKeep(1,:) == Zm1);
        Zmx1 = min(Zmx);
        PerantM(:,i4) = CKeep(:,Zmx1);
   else
        s1 = p(1,t1);
        Zm1 = BestKeep(s1,1);
        Zmx = find(ZmKeep(1,:) == Zm1);
        Zmx1 = min(Zmx);
        PerantM(:,i4) = CKeep(:,Zmx1);
   end
end
% Make 3 babies
for i5 = 1:2:6
    % Cross over point
    rc = round(1+(8-1)*rand);
    % Cross Over
    for i6 = rc:8
        PerantM(i6,i5) = PerantM(i6,i5+1);
    end
end
% Here are the babies
MainChild(:,1) = PerantM(:,1);
MainChild(:,2) = PerantM(:,3);
MainChild(:,3) = PerantM(:,5);

% Mutation
% Mutation of the main Child
Rx =0.05; % Mutation variable
for i = 1:3
    % For child one
    for j = 1:9
        ri = rand;
        if Rx > ri
           if j==1
               MainChild(j,i) = 1 + (100-1)*rand;
           end
```

```
            if j==2
                MainChild(j,i) = 1 + (100-1)*rand;
            end
            if j==3
                MainChild(j,i) = 1 + (100-1)*rand;
            end
            if j==4
                MainChild(j,i) = rand;
            end
            if j==5
                MainChild(j,i) = 1 + (10-1)*rand;
            end
            if j==6
                MainChild(j,i) = 1 + (10-1)*rand;
            end
            if j==7
                MainChild(j,i) = rand;
            end
            if j==8
                MainChild(j,i) = 1 + (50-1)*rand;
            end
            if j==9
                MainChild(j,i) = 15 + (45-15)*rand;
            end
        end
    end
end
% Store the three babies in the main population
CKeep = [];
CKeep = horzcat(Chromo1,MainChild);
for i = 1:3 % Calculate for 10 chromosomes
    Facz = MainChild(:,i);
    [ZLow,FCostKeeper,SolnKeeper] = ABSM(Fxii);
    ZmKeepC(1,i) = ZLow;
end
ZmKeep = [];
ZmKeep = horzcat(ZmKeep2,ZmKeepC);
Zmean = mean(ZmKeep);
MKeep(run,1) = Zmean;
LKeep(run,1) = min(ZmKeep);
Gkeep(run,:) = ZmKeep(1,:);

ZZlow = min(ZmKeep);
mxi = find(ZmKeep(1,:) == ZZlow);
```

```
    mx1 = min(mxi);
    if ZZlow < Zf
        Zf = ZZlow;
        CHrF = CKeep(:,mx1);
    end
end
toc
disp('Generating Graphs...')
figure (1)
plot(Gkeep,'d')
title('Generations Z value')
xlabel('Generation')
ylabel('Cost in Rands')

X = 1:1:run;
figure (2)
plot(X,MKeep,':',X,LKeep,'-')
legend('Mean Solution','Best Solution')
xlabel('Generation')
ylabel('Cost in Rands')

disp('Writing results to file...')
filename = 'GeneticR3.xlsx';
T1 = {'Factors 40'};
T2 = {'Z Value'};
T3 = {'Run Time'};
T4 = {'Iterations'};
xlswrite(filename,T1,1,'B1:B1')
xlswrite(filename,CHrF,1,'B2:B10')
xlswrite(filename,T2,1,'B12:B12')
xlswrite(filename,Zf,1,'B13:B13')
xlswrite(filename,T3,1,'B14:B14')
xlswrite(filename,toc,1,'B15:B15')
xlswrite(filename,T4,1,'B16:B16')
xlswrite(filename,run,1,'B17:B17')
disp('Process Completed...')
```

A.6 Repeatability Code

```
% Repeatability experiment
```

```
clc
clear
global Facz schedule citys fleettype fleet
schedule = xlsread('schedule R40.xlsx',1);
citys = xlsread('data.xlsx',1);
fleettype = xlsread('data.xlsx',2);
fleet = xlsread('fleet.xlsx',1);
Factor = xlsread('GeneticR.xlsx',1);

G = 3; % For the schedule you want to check

for i = 1:8
   C1(i,1) = Factor(i,G);
end

Facz = C1;

NR = 2; % This is the number of comparisons we are making

% Make sure the monte carlos are turned on so that the results are not
% determinisic

for run = 1:NR
    [ZLow,FCostKeeper,SolnKeeper] = ABSM;
    ComCost(:,run) = FCostKeeper;
end

figure (1)
hist(ComCost(:,1));
hold on
hist(ComCost(:,2));

figure(2)
hist(ComCost,15);

% Plot the normal distribution to the set of data
figure (3)
pd = fitdist(ComCost(:,1),'Normal');
minval = pd.mu - 6*(pd.sigma);
maxval = pd.mu + 6*(pd.sigma);
Xval = minval:100:maxval;
Y = pdf(pd,Xval);
plot(Xval,Y,'LineWidth',2)
hold on
```

```
pd = fitdist(ComCost(:,2),'Normal');
minval = pd.mu - 6*(pd.sigma);
maxval = pd.mu + 6*(pd.sigma);
Xval = minval:100:maxval;
Y = pdf(pd,Xval);
plot(Xval,Y,'LineWidth',2)
axis([minval maxval 0 inf])
title('Normal distribution of output values')
xlabel('Cost in Rands')
```

A.7 Utility test Code

```
% Utility test model for Campbells method
clc
clear
%%
global Facz schedule citys fleettype fleet
schedule = xlsread('schedule R200.xlsx',1);
citys = xlsread('data.xlsx',1);
fleettype = xlsread('data.xlsx',2);
fleet = xlsread('Fleet200.xlsx',1);
Factz = xlsread('GeneticR.xlsx',1);
Factz2 = xlsread('GeneticR2.xlsx',1);
%% 1st stratergy variables
% set the correct factors from the genetic algoritm
G = 7; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for ff = 1:9
   Facz(ff,1) = Factz(ff,G);
   Facz2(ff,1) = Factz2(ff,1);
end
% Functions:
% Exponential = 1
% Gaussian = 2
% Linear = 3

for i = 1:3
    Fxii = i;
    if Fxii == 3
        Facz(:,1) = Facz2(:,1);
    end
    [ZLow,FCostKeeper,SolnKeeper] = ABSM(Fxii);
```

```
    FCost(:,i) = FCostKeeper;
end
for i = 1:3
    BestSoln(1,i) = min(FCost(:,i));
end

figure(1)
Sorted = sort(FCost(:,1),'descend');
Sorted2 = sort(FCost(:,2),'descend');
Sorted3 = sort(FCost(:,3),'descend');
plot(Sorted,'+b')
hold on
plot(Sorted2,'*r')
hold on
plot(Sorted3,'og')
legend('Exponential','Gaussian','Linear')
title('Pareto chart of output solutions')
xlabel('Itteration')
ylabel('Cost on Rands')
savefig('UT2S200.fig')
```

Appendix B Sample calculations

B.1 Monte Carlo methods

This section will provide a sample calculation to illustrate the use of the Monte Carlo sampling as well as the use of linear, Gaussian and exponential utility functions as applied in both the method proposed by Campbell [7] and a multi criteria decision analysis. The calculation is as follows:

Let the environment conditions be:

Let the set of locations be: $\begin{bmatrix} L_1 & L_2 & L_3 & L_4 \\ Ta_1 & Ta_2 & Ta_3 & Ta_4 \\ Dist_1 & Dist_2 & Dist_3 & Dist_4 \end{bmatrix}$

Where:
L_i represents the set of locations.
Ta_i is the calculated time of arrival of the aircraft.
$Dist_i$ is the calculated distance from the current location to locations in question.

Let the agents initial conditions be:

Aircraft current location: $N_c = L_4$

and on board group details are: $\begin{bmatrix} g_1 & g_2 & g_3 & g_4 \\ d_1 & d_2 & d_3 & d_4 \\ LAT_1 & LAT_2 & LAT_3 & LAT_4 \end{bmatrix}$

Where:
N_c is the current location of the aircraft.

g_i is the group in question.
d_i is the associated destination.
LAT_i is the late arrival time of the associated group.

Let the groups on the ground at L_4 be: $\begin{bmatrix} w_1 & w_2 & w_3 \\ d_{w1} & d_{w2} & d_{w3} \\ EDT_{w1} & EDT_{w2} & EDT_{w3} \\ LAT_{w1} & LAT_{w2} & LAT_{w3} \\ C_{w1} & C_{w2} & C_{w3} \end{bmatrix}$

Where:
w_i are the groups on the ground.
d_{wi} is the destination of the associated group w_i.
EDT_{wi} is the early departure time
LAT_{wi} is the late arrival time.
C_{wi} is the number of people in the group.

To then calculate the attractiveness of each group on the ground Equation B.1 is used.

$$u(s \to a) = C_{wi}F_c + M_2F_{M2} \tag{B.1}$$

Where:
$u(s \to a)$ is the attractiveness of the group under consideration.
F_c is the factor that adjusts the attractiveness of capacity.
M_2 is the number of matching destinations between the groups on the aircraft and the groups on the ground.
F_{M2} is the factor that adjusts the matching groups attractiveness.

Let:
$d_{w1} = d_1 = d_2 = d_4 = d_{w1} = L_1$
$d_{w2} = d_{w3} = d_3 = L_3$
$C_{w1} = 1$
$C_{w2} = 3$
$C_{w3} = 2$

and the factors are set to:
$F_c = 1/20$
$F_{M2} = 1/2$

It is now possible to calculate the attractiveness of the groups on the ground.

Table B.1: Example Data

	M_{2i}	C_{wi}
w_1	1	1
w_2	2	3
w_3	2	2

Using Equation B.1 its possible to calculate the atractivness of each group (see Table B.2).

Table B.2: Values of attractiveness for each group

	$u(s \rightarrow a)$
w_1	0,55
w_2	1,15
w_3	1,1

A probability distribution can then be formulated (see Table B.3) and sampled from using a Monte Carlo method.

Table B.3: Monte Carlo Process

Group	$u(s \rightarrow a)$	**Cum. Utility**	**Cum. Probability**	**Range**
w_1	0,55	0,55	0,20	0 - 0.19
w_2	1,15	1,7	0,61	0.2 - 0.6
w_3	1,1	2,8	0,22	0.61 -0,99

If the random number produced falls within the range of a specific group that group is then selected for pick up.

B.2 Multi criteria decision analysis sample calculation

Let P be the matrix containing the criteria for identifying candidate passengers.

$$P = \begin{bmatrix} Tw_1 & Tu_1 & G2G_1 & Dist_1 \\ Tw_2 & Tu_2 & G2G_2 & Dist_2 \\ \vdots & \vdots & \vdots & \vdots \\ Tw_n & Tu_n & G2G_n & Dist_n \end{bmatrix}$$

Let F be the vector containing the weights for each of the factors of influence.

$$F = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

For each value of P_{ij}, scale the values between 1 and 0 using Equation B.2.

$$x' = x'_{min} + \frac{x - x_{min}}{x_{range}} \cdot x'_{range} \tag{B.2}$$

The initial utility's are then calculated:

$$U = \begin{bmatrix} Tw_1 & Tu_1 & G2G_1 & Dist_1 \\ Tw_2 & Tu_2 & G2G_2 & Dist_2 \\ \vdots & \vdots & \vdots & \vdots \\ Tw_n & Tu_n & G2G_n & Dist_n \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{B.3}$$

Each value of U is sent to a utility function, either exponential, Gaussian or linear, to calculate the final utility of each of the potential moves.

$$u(s \to a) = \frac{1}{e^{u_i}} \tag{B.4}$$

These values are then sampled using Monte Carlo sampling and the selected passenger or destination is chosen.

Appendix C Booking lists

Table C.1: S10 booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
1	360	1080	4	7	1
2	360	1080	4	5	1
3	660	930	5	8	2
4	660	900	5	6	5
5	360	1080	6	8	1
6	360	1080	7	5	1
7	360	1080	7	8	1
8	660	930	7	5	2
9	360	1080	8	5	1
10	815	870	21	20	11

Table C.2: S39 booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
1	390	1065	10	7	1
2	390	1065	10	7	1
3	390	1065	7	3	1
4	390	1065	7	10	1
5	390	1065	7	3	1
6	390	1065	7	10	1
7	390	1065	7	10	1
8	390	1065	10	7	1
9	390	1065	10	7	1
10	390	1065	7	10	1

Table C.2: S39 booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
11	390	1065	10	7	1
12	390	1065	7	26	1
13	390	1065	4	3	1
14	390	1065	4	3	2
15	420	630	26	3	8
16	390	1065	11	7	1
17	390	1065	7	11	1
18	390	1065	11	4	3
19	390	1065	7	8	1
20	390	1065	3	7	1
21	390	1065	3	7	1
22	390	1065	7	11	2
23	610	930	3	11	2
24	390	930	7	4	1
25	390	1065	7	4	1
26	390	1065	7	4	1
27	390	1065	7	4	1
28	390	1065	7	7	1
29	865	930	7	4	4
30	660	900	8	11	2
31	660	900	8	17	1
32	660	750	10	7	2
33	660	780	8	10	2
34	775	825	8	4	1
35	870	930	7	17	6
36	870	930	7	8	2
37	775	825	17	4	2
38	870	930	7	8	2
39	870	930	7	8	3

Table C.3: S40 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
1	360	1080	9	7	1
2	360	1080	7	8	3
3	770	920	7	4	2
4	445	540	8	7	2
5	765	930	7	9	4
6	360	1080	8	13	1
7	770	920	7	4	2
8	660	930	11	3	2
9	360	1080	11	7	2
10	360	1080	10	7	1
11	360	930	7	4	1
12	360	1080	7	8	1
13	660	1080	4	7	1
14	660	930	3	16	2
15	360	1080	8	13	1
16	360	1080	13	8	1
17	660	930	17	8	4
18	540	660	4	3	3
19	360	1080	13	14	1
20	660	930	4	3	2
21	360	1080	8	13	1
22	770	925	7	12	1
23	660	930	19	7	6
24	360	1080	7	8	1
25	750	930	7	17	2
26	360	1080	9	7	1
27	360	1080	8	13	1
28	770	930	7	14	2
29	745	840	16	7	2
30	360	1080	13	7	11
31	360	1080	7	4	1
32	660	930	4	3	2
33	660	900	1	2	1

Table C.3: S40 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
34	360	1080	7	3	1
35	360	1080	7	8	1
36	660	910	2	3	1
37	360	1080	6	7	1
38	660	930	10	4	2
39	660	930	4	8	2
40	660	900	1	2	1

Table C.4: S99 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
1	390	1065	7	9	1
2	390	1065	19	7	1
3	390	1065	19	7	2
4	390	930	7	19	1
5	390	1065	7	19	2
6	390	1065	7	10	1
7	390	1065	7	10	1
8	390	1065	7	10	1
9	390	1065	7	10	1
10	390	1065	7	2	1
11	390	1065	2	7	1
12	390	1065	2	7	1
13	390	1065	7	2	1
14	660	1065	26	7	1
15	390	1065	7	17	13
16	390	1065	7	9	4
17	390	1065	7	9	6
18	390	1065	7	9	1
19	390	1065	9	7	1
20	390	1065	7	9	1
21	390	1065	9	7	1
22	390	1065	9	7	1

Table C.4: S99 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
23	390	1065	17	7	1
24	420	630	6	3	9
25	420	630	6	3	2
26	420	630	6	3	2
27	420	630	6	3	1
28	420	630	6	3	1
29	420	630	3	6	9
30	420	630	3	6	2
31	420	630	3	6	2
32	420	630	3	6	2
33	420	630	3	6	1
34	390	1065	7	27	1
35	390	1065	7	27	1
36	390	1065	7	27	1
37	390	1065	7	27	1
38	390	1065	7	27	1
39	390	1065	7	27	1
40	390	1065	8	7	1
41	390	1065	7	2	1
42	390	1065	7	27	1
43	390	1065	8	7	1
44	390	1065	8	7	1
45	390	1065	7	27	1
46	390	1065	7	8	1
47	390	1065	11	7	1
48	390	1065	7	27	1
49	390	1065	7	2	2
50	390	930	11	7	1
51	390	1065	7	11	1
52	390	1065	7	27	1
53	390	1065	7	27	1
54	660	930	9	7	2
55	870	1005	7	19	3

Table C.4: S99 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
56	610	930	7	19	2
57	660	930	4	7	2
58	390	1065	7	4	1
59	390	1065	7	4	1
60	390	1065	7	4	1
61	390	1065	27	7	1
62	390	1065	7	7	1
63	390	1065	3	7	1
64	660	930	10	3	2
65	360	930	7	26	2
66	660	930	27	7	2
67	660	930	4	9	4
68	660	750	11	7	2
69	840	930	7	8	9
70	840	930	7	8	9
71	645	750	8	7	2
72	660	930	17	11	2
73	660	960	10	3	2
74	360	930	7	2	2
75	660	930	27	7	2
76	660	745	4	7	2
77	660	930	2	8	4
78	870	930	7	8	1
79	660	930	4	3	9
80	660	930	4	3	2
81	590	720	4	7	2
82	660	900	2	11	2
83	660	735	9	7	2
84	660	750	11	7	3
85	645	750	8	7	4
86	660	930	27	7	2
87	390	1065	7	4	3
88	660	930	4	11	2

Table C.4: S99 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
89	660	750	4	7	4
90	660	900	4	11	8
91	390	1065	7	19	2
92	660	900	19	11	4
93	660	710	11	7	4
94	600	755	19	3	2
95	585	720	27	7	2
96	660	750	4	3	2
97	850	1000	7	2	4
98	820	950	7	2	2
99	840	930	7	2	1

Table C.5: S102 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
1	945	1100	7	12	2
2	360	1080	7	2	4
3	360	1080	7	2	1
4	360	1080	2	14	1
5	360	1080	2	14	1
6	360	1080	13	7	11
7	360	1080	9	7	1
8	660	930	7	2	3
9	360	1080	8	7	1
10	360	1080	7	10	1
11	360	1080	7	10	2
12	360	1080	7	8	1
13	360	1080	11	7	2
14	360	1080	13	14	1
15	660	900	7	1	1
16	360	1080	7	3	1
17	360	1080	7	8	3
18	360	1080	7	11	1

Table C.5: S102 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
19	360	1080	7	8	1
20	360	1080	7	8	1
21	360	1080	7	8	1
22	360	1080	8	7	1
23	360	1080	8	7	1
24	360	1080	8	7	1
25	360	1080	8	7	1
26	360	1080	8	7	1
27	360	1080	8	7	1
28	360	1080	8	7	1
29	360	1080	8	7	1
30	360	1080	8	7	1
31	360	1080	8	7	1
32	360	1080	7	4	1
33	660	930	7	8	2
34	660	910	2	3	1
35	360	1080	4	7	1
36	360	1080	7	10	1
37	360	1080	10	4	1
38	360	1080	7	10	1
39	360	1080	10	4	1
40	360	1080	7	13	2
41	360	1080	3	3	1
42	360	1080	7	4	1
43	360	1080	7	3	1
44	770	930	7	23	4
45	600	930	3	18	2
46	660	930	15	9	2
47	660	930	15	9	2
48	660	930	15	9	1
49	660	930	9	11	2
50	660	930	4	11	2
51	660	930	2	8	2

Table C.5: S102 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
52	660	900	6	4	4
53	660	900	6	4	2
54	660	930	10	4	2
55	660	930	10	4	2
56	660	930	10	4	2
57	445	540	8	7	2
58	755	840	9	7	2
59	660	900	9	4	2
60	660	930	9	11	2
61	795	930	7	9	2
62	660	840	8	3	2
63	880	930	3	18	2
64	660	840	19	3	2
65	880	930	3	20	2
66	660	930	10	9	2
67	660	930	4	9	2
68	750	840	14	7	2
69	660	930	22	24	2
70	660	930	16	10	2
71	660	930	10	23	2
72	785	880	16	7	2
73	660	910	16	3	2
74	660	900	4	3	2
75	880	930	3	18	2
76	660	930	16	9	2
77	600	930	3	9	2
78	770	840	4	7	2
79	770	920	7	4	2
80	770	920	7	4	2
81	770	920	7	4	1
82	770	920	7	4	2
83	770	900	7	2	2
84	770	890	7	10	3

Table C.5: S102 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
85	630	930	7	9	2
86	745	840	16	7	2
87	660	900	4	3	2
88	750	930	3	20	2
89	660	930	11	2	2
90	795	930	7	9	3
91	760	840	25	7	2
92	660	930	1	4	3
93	660	930	1	4	2
94	660	930	1	4	1
95	660	930	2	1	4
96	660	930	2	1	1
97	540	660	4	3	3
98	840	895	3	18	3
99	540	660	4	3	3
100	840	895	3	18	3
101	540	660	4	3	1
102	840	895	3	18	1

Table C.6: S139 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
1	390	1065	3	7	1
2	390	1065	3	7	2
3	390	1065	19	7	1
4	390	1065	19	7	1
5	390	1065	19	7	1
6	390	1065	19	7	1
7	390	1065	19	7	1
8	390	1065	19	7	1
9	390	1065	4	7	1
10	390	1065	19	7	1
11	390	1065	7	19	5

Table C.6: S139 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
12	390	1065	7	19	8
13	390	1065	10	7	1
14	390	1065	7	10	1
15	390	1065	7	2	1
16	390	1065	2	7	1
17	390	1065	2	7	1
18	390	1065	7	2	1
19	390	1065	7	2	1
20	390	1065	8	7	1
21	390	1065	8	7	1
22	390	1065	8	7	1
23	390	1065	8	7	1
24	390	1065	9	7	1
25	390	1065	9	7	1
26	390	1065	7	9	1
27	390	1065	7	17	13
28	390	1065	9	7	2
29	390	1065	7	17	3
30	390	930	7	9	4
31	390	1065	7	9	5
32	390	1065	7	9	8
33	390	1065	7	9	1
34	390	1065	7	9	1
35	390	1065	7	9	1
36	390	1065	7	9	1
37	390	1065	17	7	1
38	390	1095	17	7	1
39	390	1065	9	7	1
40	390	1095	17	7	1
41	390	1065	7	9	1
42	390	1095	17	7	1
43	390	1065	9	7	1
44	390	1095	17	7	1

Table C.6: S139 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
45	390	1065	9	7	1
46	390	1065	7	9	1
47	390	1065	7	9	1
48	390	1065	7	9	1
49	390	1065	9	7	1
50	390	1095	17	7	1
51	390	1065	9	7	1
52	390	1065	7	9	1
53	390	1065	7	9	1
54	390	1065	9	7	1
55	390	1065	9	7	1
56	390	1065	7	9	1
57	390	1065	7	8	1
58	390	1065	7	8	2
59	390	1065	7	11	2
60	390	1065	7	11	7
61	390	1065	7	8	1
62	390	1065	7	11	1
63	390	1065	7	2	1
64	390	1065	7	8	1
65	390	1110	11	7	1
66	390	1065	2	7	1
67	390	1065	7	8	1
68	390	1065	7	11	1
69	390	1065	7	11	1
70	390	1065	2	7	1
71	390	1065	8	3	1
72	390	1065	19	7	1
73	390	1065	7	8	1
74	660	750	8	7	1
75	660	930	7	10	1
76	620	930	7	11	1
77	870	990	7	27	2

Table C.6: S139 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
78	660	780	9	8	1
79	390	1065	7	10	4
80	660	900	8	4	2
81	440	660	11	3	2
82	390	1065	4	7	1
83	390	1065	7	4	1
84	390	1065	7	4	13
85	390	1065	7	4	6
86	390	1065	7	4	7
87	390	1065	4	7	1
88	390	1065	7	4	1
89	390	1065	4	3	1
90	390	1065	4	3	1
91	390	1065	4	7	2
92	390	1065	4	7	1
93	390	1065	27	7	1
94	390	1065	7	7	1
95	390	1135	9	7	1
96	390	1065	4	10	2
97	660	780	4	3	2
98	360	840	3	18	2
99	660	930	8	3	2
100	870	930	7	8	4
101	360	840	3	18	2
102	660	930	8	3	2
103	480	810	8	28	8
104	360	985	7	27	2
105	660	930	17	7	2
106	660	750	9	7	2
107	660	750	4	7	2
108	360	930	7	4	4
109	660	930	27	7	4
110	605	930	7	8	2

Table C.6: S139 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
111	660	780	4	3	5
112	660	750	4	8	2
113	695	780	11	7	2
114	660	750	8	7	2
115	360	930	3	4	2
116	750	930	18	3	2
117	600	930	30	8	2
118	660	930	4	3	2
119	845	870	3	20	2
120	865	930	7	4	2
121	610	750	3	8	2
122	540	600	18	3	2
123	360	985	7	27	2
124	700	930	8	7	2
125	660	930	10	4	4
126	660	780	4	3	4
127	660	840	3	18	4
128	870	985	7	27	4
129	610	930	3	4	4
130	360	690	3	4	2
131	545	930	20	3	2
132	660	810	9	8	2
133	665	750	9	7	2
134	660	780	11	7	2
135	870	985	7	27	2
136	665	750	9	7	8
137	865	930	7	4	2
138	660	750	2	7	2
139	795	900	7	2	7

Table C.7: S200 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
1	650	750	19	7	1
2	390	1065	7	19	1
3	390	1065	7	19	1
4	390	1065	7	19	2
5	390	1065	10	7	1
6	390	1065	7	10	1
7	390	1065	7	10	1
8	390	1065	10	7	1
9	390	1065	7	10	2
10	390	1065	7	10	2
11	390	1065	2	7	1
12	390	1065	2	7	1
13	390	1065	2	7	1
14	390	1065	26	7	1
15	635	930	3	7	1
16	390	1065	7	17	13
17	390	1065	7	9	5
18	390	1065	7	17	1
19	390	1065	7	9	1
20	390	1065	7	9	1
21	390	1065	7	9	1
22	390	1065	17	7	1
23	390	1065	7	9	1
24	420	630	6	3	1
25	420	630	6	3	10
26	420	630	6	3	4
27	420	630	6	3	1
28	420	630	3	6	1
29	420	630	3	6	10
30	420	630	3	6	4
31	360	1065	7	8	1
32	390	1065	11	7	1
33	390	1075	7	8	1

Table C.7: S200 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
34	390	1065	7	8	1
35	390	1065	7	8	2
36	390	1075	7	8	2
37	390	1075	7	8	2
38	390	1065	4	7	1
39	390	1065	7	8	1
40	390	1065	7	8	1
41	390	1065	8	13	1
42	390	1065	13	8	1
43	390	1065	8	7	1
44	390	1065	8	13	1
45	390	1065	7	8	1
46	390	1065	7	8	1
47	390	1065	13	8	1
48	390	1065	8	7	1
49	390	1065	7	8	1
50	390	1065	8	7	1
51	390	1065	8	13	1
52	390	1065	13	8	1
53	390	1065	13	8	1
54	390	1065	8	13	1
55	390	1065	13	8	1
56	390	1065	7	8	1
57	390	1065	13	8	1
58	390	1065	8	7	1
59	390	1065	8	13	1
60	390	1065	13	8	1
61	390	1065	8	13	1
62	390	1065	8	13	1
63	390	1065	13	8	1
64	390	1065	13	8	1
65	390	1065	8	13	1
66	390	1065	7	8	1

Table C.7: S200 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
67	390	1065	7	8	1
68	390	1065	13	8	1
69	390	1065	8	13	1
70	390	1065	8	13	1
71	390	1065	7	8	1
72	390	1065	13	8	1
73	390	1065	8	7	1
74	390	1065	13	8	1
75	390	1065	8	13	1
76	390	1065	8	7	1
77	390	1065	11	7	1
78	390	1065	13	8	1
79	390	1065	8	13	1
80	390	1065	7	13	1
81	390	1065	8	13	1
82	390	1065	13	8	1
83	390	1065	7	8	1
84	390	1065	8	7	1
85	390	1065	13	8	1
86	390	1065	13	8	1
87	390	1065	8	7	1
88	390	1065	8	13	1
89	390	1065	7	8	1
90	390	1065	13	8	1
91	390	1065	13	8	1
92	390	1065	8	13	1
93	390	1065	13	8	1
94	390	1065	13	8	1
95	390	1065	8	13	1
96	390	1065	8	7	1
97	390	1065	8	13	1
98	390	1065	13	8	1
99	390	1065	7	8	1

Table C.7: S200 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
100	390	1065	8	13	1
101	390	1065	7	8	1
102	390	1065	8	7	1
103	390	1065	13	8	1
104	390	1065	13	8	1
105	390	1065	13	8	1
106	390	1065	8	13	1
107	390	1065	7	8	1
108	390	1065	8	13	1
109	390	1065	8	13	1
110	390	1065	13	8	1
111	390	1065	13	8	1
112	390	1065	8	13	1
113	390	1065	8	13	1
114	390	1065	8	7	1
115	390	1065	7	8	1
116	390	1065	8	7	1
117	390	1065	8	7	1
118	390	1065	8	13	1
119	390	1065	8	13	1
120	390	1065	7	6	1
121	390	1065	7	6	1
122	860	930	7	17	6
123	660	930	9	2	2
124	825	930	3	10	2
125	660	930	9	8	2
126	660	930	4	7	4
127	690	930	27	7	2
128	690	930	27	7	4
129	390	1065	7	4	1
130	390	1065	7	4	2
131	390	1065	7	4	2
132	390	1065	7	4	5

Table C.7: S200 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
133	390	1065	7	4	1
134	390	1065	7	4	1
135	390	1065	7	4	1
136	390	1065	7	4	1
137	390	1065	4	7	1
138	390	1065	4	7	1
139	390	1065	7	4	1
140	390	1065	4	7	1
141	390	1065	4	7	1
142	390	1065	4	7	1
143	390	1065	4	7	1
144	390	1065	7	4	1
145	390	1065	7	7	1
146	390	1065	7	6	1
147	390	1065	7	6	1
148	635	750	4	7	2
149	660	930	10	17	2
150	660	930	8	3	2
151	660	930	10	3	4
152	820	930	3	20	4
153	660	930	4	3	2
154	820	930	3	18	2
155	660	900	17	4	4
156	660	930	11	8	2
157	660	740	17	7	2
158	660	900	2	17	2
159	840	930	3	10	7
160	750	930	18	3	7
161	820	900	3	4	4
162	755	820	20	3	4
163	860	930	7	4	2
164	660	930	10	8	2
165	660	900	9	4	2

Table C.7: S200 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
166	660	900	14	11	4
167	660	930	9	8	2
168	660	900	8	4	2
169	645	750	8	7	2
170	660	930	4	3	6
171	820	930	3	18	6
172	660	780	4	3	2
173	610	930	3	8	2
174	360	930	8	17	2
175	610	930	3	8	3
176	540	930	18	3	3
177	720	780	8	7	3
178	660	900	17	11	2
179	700	930	11	19	2
180	650	755	11	7	2
181	660	900	19	4	1
182	660	930	8	11	4
183	660	930	4	9	2
184	610	750	3	8	4
185	660	910	4	26	2
186	660	900	19	9	2
187	800	900	3	4	2
188	750	930	18	3	2
189	660	780	8	3	4
190	840	930	7	9	3
191	700	930	11	19	2
192	665	750	9	7	2
193	660	930	9	8	1
194	810	930	7	8	2
195	870	985	7	27	4
196	650	755	11	7	2
197	660	755	8	7	2
198	660	900	2	4	2

Table C.7: S200 - booking list.

Group No.	EDT	LAT	Origin	Destination	No. of people in group
199	840	1000	7	2	4
200	840	930	7	2	1

C.1 Fleet of aircraft

Table C.8: Fleet for booking list S10, S40 and S120.

Plane No.	Type	Starting Location
1	1	7
2	1	7
3	1	7
4	1	4
5	1	18
6	2	3
7	2	7
8	2	7
9	2	7
10	2	7
11	2	17
12	2	3
13	2	8
14	2	19

Table C.9: Fleet for booking list 39.

Plane No.	Type	Starting Location
1	2	7
2	2	11
3	2	3
4	1	7
5	1	7
6	2	8
7	2	4
8	1	7
9	1	7
10	1	7
11	2	4
12	2	8
13	2	3
14	2	7

Table C.10: Fleet for booking list S99.

Plane No.	Type	Starting Location
1	2	7
2	2	7
3	2	3
4	1	7
5	1	8
6	2	7
7	1	3
8	2	3
9	2	7
10	2	26
11	1	7
12	1	7
13	2	7
14	1	3

Table C.11: Fleet for booking list S139.

Plane No.	**Type**	**Starting Location**
1	2	17
2	2	7
3	2	7
4	1	7
5	2	7
6	1	7
7	2	3
8	1	7
9	1	7
10	1	7
11	2	7
12	2	7
13	2	3
14	2	17
15	1	7
16	1	7
17	1	7
18	1	7

Table C.12: Fleet for booking list S200.

Plane No.	**Type**	**Starting Location**
1	2	8
2	2	7
3	2	2
4	1	18
5	1	7
6	1	7
7	2	7
8	2	9
9	1	7
10	1	18
11	2	7
12	2	9
13	2	2
14	2	8
15	2	8
16	2	7
17	1	7
18	1	18
19	1	7
20	1	18

Appendix D ABSM generated schedules

Table D.1: Schedule S10.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
1	4	360	21	397	0	
	21	815	20	867	11	10
2	7	670	8	717	4	7 6 8
	8	717	4	755	4	9 6 8
	4	755	5	775	5	2 9 6 8
	5	775	6	814	5	4
	6	814	5	854	1	5
	5	854	8	893	3	3 5
	8	893	4	931	0	
	4	931	7	987	1	1

Table D.2: Schedule S39.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
2	11	390	7	431	4	20 21
	7	431	4	487	5	4 21
	4	487	3	547	5	3 4
	3	610	7	705	4	2 1
	7	705	11	746	5	11 2
	11	746	7	787	0	
	7	865	4	921	5	5 7
	4	921	7	977	0	
	7	977	8	1024	5	8 10
	8	1024	10	1057	4	10
1	7	870	17	909	6	12

Table D.2: Schedule S39.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
2	8	775	4	813	1	17
2	4	390	10	430	0	
	10	430	7	457	5	18
1	7	870	8	910	7	9
2	7	390	26	423	0	
	26	423	3	481	8	23
2	4	390	8	428	0	
	8	660	17	684	1	15
2	8	660	8	670	2	14
	8	670	11	688	4	16 14
	11	688	10	717	2	16
	10	717	7	743	2	19
2	7	390	26	429	4	13 6
	26	429	17	477	3	6
	17	660	4	712	5	22 6

Table D.3: Schedule S40.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
1	7	360	19	396	0	
	19	660	7	696	6	28
1	7	770	7	780	6	15
	7	780	4	827	10	12 11 15
	4	827	8	860	6	7 12 11
	8	860	14	873	4	12 11
	14	873	17	891	2	11
	17	891	13	913	0	
	13	913	7	961	11	25
1	7	855	7	865	10	9 14
	7	865	8	905	10	9 14
	8	905	9	920	4	9
	9	920	8	936	2	18
	8	936	13	956	6	17 18
	13	956	8	976	4	24 23 18

Table D.3: Schedule S40.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	8	976	7	1016	3	24 18
	7	1016	14	1056	1	24
1	4	540	3	591	3	5
2	3	660	16	724	2	3
	16	745	4	759	2	26
	4	759	10	799	3	4 26
	10	799	7	825	4	19 4 26
2	7	360	4	416	1	13
	4	660	3	721	5	6 13
	3	721	1	801	0	
	1	801	2	832	2	1
2	7	360	10	386	0	
	10	660	4	700	2	20
2	7	770	12	834	1	10
2	17	660	19	675	4	27
	19	735	8	762	4	27
2	3	360	8	449	0	
	8	449	11	467	2	16
	11	467	6	496	4	22 16
	6	496	7	531	5	8 22 16
2	8	360	11	378	0	
	11	660	2	676	2	21
	2	676	3	774	3	2 21

Table D.4: Schedule S99.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
2	7	390	7	400	3	30
	7	400	26	439	5	34 30
	26	439	19	486	3	30
	7	390	2	428	5	16
	2	660	7	698	4	2
	7	698	8	745	5	21 2
	8	745	7	792	3	36
2						

Table D.4: Schedule S99.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	7	850	2	888	5	18 19
	2	888	7	926	2	1
	7	926	10	952	5	28 27
	10	952	7	978	1	28
	7	978	7	988	4	32 28
	7	988	19	1030	4	32 28
	19	1030	11	1049	1	28
	7	390	9	433	11	25
	9	433	19	455	7	38 39
	19	455	7	490	10	45 38 39
	7	840	7	850	9	23
1	7	870	8	910	10	24 23
	8	910	4	943	0	
	4	943	11	979	8	9
	11	979	17	997	8	9
	17	997	7	1036	9	43 9
	8	360	4	393	0	
	4	660	3	711	11	7
1	3	711	4	762	6	5
	4	762	6	789	10	10 5
	6	789	9	821	4	10
	3	360	4	411	0	
	4	411	11	448	10	11
1	11	448	7	482	11	41 42
	7	482	10	506	6	20
	10	660	4	695	10	40 20
	4	695	3	746	4	40
	3	390	7	485	1	6
	7	485	7	495	4	31 26
	7	495	19	537	4	31 26
2	19	600	9	625	4	47 26
	9	625	4	669	2	47
	4	669	3	729	4	12 47
2	7	360	6	395	0	

Table D.4: Schedule S99.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	6	420	3	497	4	14
2	26	660	7	699	1	48
	7	820	2	858	4	17 15
	2	858	11	874	2	3
1	7	390	17	429	11	29
	17	660	7	699	2	44
	7	840	11	875	11	22 44
	11	875	8	891	9	22
	8	891	4	924	6	37
	4	924	7	971	6	37
1	7	390	27	441	11	35
	27	660	7	711	9	49 51 50
	7	711	19	747	0	
	19	747	11	765	4	46
2	7	610	4	666	2	33
	4	666	4	676	4	8 33
	4	676	19	729	4	8 33
	19	729	7	770	2	8
1	3	360	3	370	10	4
	3	370	6	434	10	4
	6	434	3	499	11	13

Table D.5: Schedule S102.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
1	7	780	4	827	11	38 28
	4	827	9	864	8	16 15 38
	9	864	11	883	10	43 15 38
	11	883	23	905	6	51 38
	23	905	2	930	2	51
	2	930	7	962	2	3
	7	962	8	1002	11	27 29 3
	8	1002	4	1035	4	27 3
	4	1035	14	1071	2	3
1	7	360	13	408	0	
	13	408	7	456	11	52
	7	456	6	486	0	
	6	660	4	688	6	21
	4	770	16	783	2	18
	16	783	25	817	4	60 18
	25	817	7	848	4	60 18
1	7	770	2	802	11	35 30 25 34 24
	2	802	11	817	6	35 30 34
	11	817	10	843	5	30 34
	10	843	8	872	8	47 30
	8	872	4	904	6	47
1	4	540	3	591	7	20
	3	890	18	916	6	11
	18	916	8	1004	0	
	8	1004	7	1044	11	39
1	18	360	3	386	0	
	3	840	3	850	7	13
	3	850	3	860	8	7 13
	3	860	18	886	7	13
2	3	600	16	664	2	9
	16	785	9	825	4	57 9
	9	825	7	876	2	57
	7	876	22	903	0	

Table D.5: Schedule S102.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	22	903	24	921	2	62
2	7	660	1	710	5	22 37 31
	1	710	9	734	5	2 37 31
	9	734	16	774	3	2 37
	16	774	13	818	5	59 2 37
	13	818	4	866	4	53 59 2
	4	866	9	910	3	53 59
	9	910	14	923	1	53
2	7	360	10	386	5	33
	10	386	16	425	2	46
	16	660	16	670	4	56 46
	16	690	4	704	4	56 46
	4	704	25	747	2	56
	25	760	10	785	4	63 56
	10	785	7	811	2	63
2	7	360	10	386	0	
	10	660	10	670	2	48
	10	670	15	693	4	49 48
	15	693	23	707	4	49 48
	23	707	16	735	2	48
	16	735	9	775	4	58 48
	9	775	2	799	2	58
	2	799	3	897	3	4 58
2	17	360	7	406	0	
	7	795	9	846	5	32
	9	846	4	890	2	45
	4	890	7	946	1	14
	7	946	3	1041	2	26
2	3	360	15	442	0	
	15	660	9	688	5	55
	9	755	14	769	2	44
	14	769	25	790	4	54 44
	25	790	7	826	4	54 44
2	8	445	9	462	2	40

Table D.5: Schedule S102.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	9	462	11	482	3	42 40
	11	482	7	522	5	50 42 40
	7	945	12	1009	2	36
	19	660	2	676	2	61
	2	676	8	700	4	5 61
2	8	700	16	734	4	41 61
	16	734	3	798	4	41 61
	3	880	20	912	4	10 8

Table D.6: Schedule S139.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	17	360	7	406	0	
2	7	870	8	917	4	22
	8	917	9	934	4	42
	9	934	7	984	5	47 42
2	7	360	2	398	0	
	2	660	7	698	2	2
	7	360	11	401	0	
2	11	440	18	551	2	53
	18	551	3	580	4	59 53
1	7	390	8	430	7	21
	8	480	28	663	8	45
	7	660	10	686	5	28 16
2	10	686	2	715	4	16
	2	715	7	753	4	1
	7	753	9	804	4	24
	7	620	11	655	11	32 31
1	11	660	9	678	5	54 55
	9	678	7	721	7	48 54 55
2	3	360	4	421	4	5
	4	660	8	698	2	14
1	7	360	4	407	10	20
	7	390	4	437	11	19
1						

Table D.6: Schedule S139.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	4	660	7	707	8	13 12
	7	707	17	746	11	33
	17	746	7	785	6	56
	7	785	9	828	8	26
	9	828	7	871	11	46
	7	871	9	914	6	25
	7	390	10	416	5	29
2	10	670	7	696	5	51 52
	7	696	4	752	4	52
	7	390	19	432	5	35
	19	432	8	458	0	
2	8	670	3	759	5	39
	3	759	20	791	0	
	20	791	3	823	2	61
	3	610	8	699	2	6
	8	700	4	738	4	44 41
2	4	738	7	794	4	15 44
	7	794	8	841	4	23 15
	8	841	10	874	2	15
	17	660	8	684	2	57
2	8	684	7	732	5	43 57
	7	752	27	813	4	37
	27	813	7	874	5	62
	7	390	9	433	11	27
1	9	685	7	728	10	49
	7	795	2	827	9	30 17
	2	827	11	842	2	30
	7	390	4	437	11	18
	4	660	3	711	11	11
	3	711	4	762	8	7 4
	4	762	3	813	6	10 7
1	3	813	18	839	8	8 7
	18	839	3	865	2	58
	3	865	7	944	5	9 3

Table D.6: Schedule S139.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	7	944	20	1029	2	9
1	7	880	27	931	8	38
	7	400	19	436	11	34 36
	19	436	17	450	11	60 34
	17	450	9	468	8	60
1	9	660	8	675	11	50 60
	8	675	7	715	8	60
	7	715	30	751	0	
	30	751	8	817	2	63

Table D.7: Schedule S200.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	8	360	7	407	2	49
	7	810	17	856	5	35 29 49
2	17	856	8	880	2	29
	8	880	13	903	0	
	13	903	8	925	4	64
	7	390	6	425	4	37
	6	425	10	443	4	37
2	10	453	19	488	4	37
	19	488	17	503	0	
	17	660	7	706	2	68
2	2	660	4	708	2	3
	4	708	26	737	4	18 17
	26	737	9	782	3	76 17
	9	782	11	802	1	76
	11	802	8	820	3	60 76
	8	820	7	867	1	76
2	18	360	6	437	0	
	6	437	3	502	5	20
	3	502	20	530	0	
	20	755	3	783	4	75
1	7	390	8	430	11	27

Table D.7: Schedule S200.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	8	430	7	470	11	42
	7	470	17	509	11	34
	17	660	8	682	7	66 69 67
	8	682	11	698	11	45 66 69 67
	11	700	4	737	11	58 61 66 67
	4	737	7	784	11	14 58 61 66
	7	784	7	794	10	32 61
	7	794	10	817	11	33 32 61
	10	817	19	847	7	57 33 61
	19	847	8	870	3	57 33
	8	870	13	890	1	33
1	7	390	6	420	9	30
	6	430	9	461	9	30
	9	665	14	678	2	52
	14	678	11	693	6	65 52
	11	693	7	728	6	59 52
2	7	360	3	455	0	
	3	455	6	533	4	6
2	9	660	8	677	5	53
	8	720	7	767	5	41 44
	7	860	4	916	2	24
1	7	390	8	430	11	26
	8	430	13	450	11	46
	13	450	8	470	11	62
	8	660	3	734	6	39
	3	820	18	846	8	10
1	18	750	3	776	9	71
	3	825	7	904	10	7 9
	7	904	10	927	9	9
2	7	850	9	901	5	28 31
	9	901	8	917	2	28
2	9	670	14	684	4	51 50
	14	724	2	745	4	51 50

Table D.7: Schedule S200.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	2	745	4	794	2	51
2	2	660	19	676	2	2
	19	676	17	691	5	73 74 2
	17	691	9	711	3	73 74
	9	711	10	750	1	73
	10	750	4	790	5	56 73
	4	790	3	851	4	56
2	8	660	13	682	5	40 48
	13	682	14	703	2	40
	14	733	4	775	2	40
2	8	645	19	672	4	43
	19	672	7	713	5	72 43
	7	870	27	931	4	38
	27	931	10	1008	0	
	10	1008	7	1034	2	54
2	7	840	2	878	5	21
	2	878	7	916	3	1
	7	916	4	972	5	22
	4	972	7	1027	0	
	7	1027	6	1062	4	25
1	7	860	10	883	6	36
	10	883	17	915	8	55 36
	17	915	8	937	0	
	8	937	13	957	11	47
	13	957	8	977	11	63
	8	977	4	1010	0	
	4	1010	7	1057	7	12
1	18	540	3	566	3	70
	3	610	4	661	9	8
	4	661	8	694	11	13 8
	8	694	14	707	2	13
	14	707	7	747	2	13
	7	747	27	798	0	
	27	798	7	850	6	77

Table D.7: Schedule S200.

Type	Origin	Departure	Dest.	Arrival	Cap. at Origin	Groups
	7	850	14	890	0	
1	7	390	4	437	11	23
	4	670	3	721	10	16
	3	800	4	851	6	4
1	18	360	3	386	0	
	3	420	6	484	11	5
	6	484	3	549	11	19
	3	820	20	848	4	11

Appendix E List of cities

Table E.1: List of Locations.

No.	X - Coord	Y - Coord	Code	Location Name
1	23,041	-18,771	XOR	lechwe islan camp
2	22,737	-19,386	XIG	xigere
3	25,163	-17,831	BBK	chobe
4	23,644	-18,541	CBE	linyanti
5	23,493	-18,272	KWD	lianshulu
6	23,435	-19,197	SWI	moremi tented camp
7	23,428	-19,973	MUB	maun
8	22,815	-18,955	VUM	vumburu south
9	22,597	-18,955	JAO	jacana
10	23,377	-19,467	CTB	chitabe trails
11	22,795	-19,209	MOM	little mombo
12	25,151	-20,456	TSO	jacks camp
13	22,424	-18,824	SRA	seronga
14	22,703	-19,018	OMD	duba plains
15	23,162	-19,111	SHN	shinde camp
16	23,518	-18,57	SLI	selinda
17	22,476	-19,276	HND	tubu rr
18	25,819	-17,823	LVI	river club
19	22,551	-19,415	ABU	abu
20	25,845	-18,101	VFA	vic falls
21	24,179	-17,635	MLO	-
22	23,05	-19,549	XXB	Eagle Island
23	23,283	-19,105	KWZ	Kwara
24	22,789	-19,491	NXB	Nxabega

Table E.1: List of Locations.

No.	X - Coord	Y - Coord	Code	Location Name
25	22,909	-19,311	PJO	Chiefs camp
26	23,7501	-19,1245	BKA	Banoka
27	24,02129	-21,4876	KHP	Kalahari Plains
28	29,10615	-22,1872	TUI	Tuli
29	27,46234	-21,1629	FRW	Francistown
30	23,65011	-20,9751	DVL	Deception Valley Lodge