A Tabu Search Metaheuristic Algorithm for the Multiple Depot Vehicle Routing Problem with Time Windows

by

Jonathan Abrie de Freitas

29089655

Submitted in partial fulfilment of the requirements for the degree of

Bachelors of Industrial and Systems Engineering

in the

Faculty of Engineering, Built Environment and Information Technology

University of Pretoria

October 2012

Executive Summary

The problems encountered by courier companies in directing their fleets along road networks to visit customers, which are geographically distributed, are common problems which are encountered frequently. These problems are by no means isolated to courier companies. Any set of vehicles which is involved in delivery, collection or a combination of both delivery and collection encounter a variation of the vehicle routing problem (VRP). Several variations of the vehicle routing problem exist. The algorithmic solutions to the individual variants of the vehicle routing problem seek to optimise the routes assigned to a fleet of vehicles in visiting an array of nodes (which represent points of delivery or collection or from another perspective, a set of customers). The optimal solution of a VRP instance is the shortest, quickest or cheapest set of routes assigned to a fleet of vehicles which satisfies all customer demand without contravening any of the instance-specific constraints. The vehicle routing problem has been identified as an non-determinant polynomial-time (NP) hard problem. This classification gives an indication of the computational complexity of the problem. Problems of this class require an inordinate amount of time to be solved to optimality for large problem instances. To overcome such obstacles, heuristic and metaheuristic search algorithms are often utilised to arrive at near-optimal or satisfactory solutions in less time.

Contents

1 Introduction **6**
- 1.1 Problem Statement . . . 6
- 1.2 Potential Solution Strategies . . . 7
- 1.3 Solution Strategy . . . 7
- 1.4 Project Scope . . . 8
  - 1.4.1 Input . . . 9
  - 1.4.2 Output . . . 9
  - 1.4.3 Dataset Interfaces . . . 10
- 1.5 Case Study . . . 10
- 1.6 Considerations . . . 12
- 1.7 Document Structure . . . 12

2 Literature Review **13**
- 2.1 Introduction . . . 13
- 2.2 Problem Variants . . . 14
  - 2.2.1 Capacitated Vehicle Routing Problem . . . 15
  - 2.2.2 Vehicle Routing Problem with Time Windows . . . 15
  - 2.2.3 Vehicle Routing Problem with Backhauls . . . 15
  - 2.2.4 Vehicle Routing Problem with Pickup and Delivery . . . 16
  - 2.2.5 Multiple Depot Vehicle Routing Problem . . . 16
  - 2.2.6 Applications . . . 17
- 2.3 Metaheuristics . . . 18
  - 2.3.1 Simulated Annealing . . . 19
  - 2.3.2 Tabu Search . . . 20
  - 2.3.3 Ant Colony Optimisation . . . 20
  - 2.3.4 Application . . . 21
- 2.4 Conclusion . . . 22

3 Model Formulation **23**
- 3.1 The Problem . . . 23
  - 3.1.1 Problem Characteristics . . . 23
  - 3.1.2 Mathematical Model . . . 24
- 3.2 Solution Algorithm . . . 27
  - 3.2.1 Important Concepts . . . 27

3.2.2 Initial Solution Algorithm 28
3.2.3 Tabu Search Algorithm 29
3.3 Verification 31
3.3.1 Solomon's VRPTW Benchmarking Problems 31
3.3.2 Convergence and Repeatability of the Solution Algorithm 31
3.4 Conclusion 32

4 Implementation **39**
4.1 Introduction 39
4.2 Parameter Tuning 39
4.2.1 Design of Experiments 40
4.2.2 Screening Experiment 40
4.3 Case Study Problem 41
4.4 Conclusion 42

5 Reflection **50**

References **51**

Glossary **52**

Acronyms **53**

List of Figures

1.1 IDEF∅ diagram depicting dataset interfaces 10

3.1 Convergence of the solution algorithm 38

4.1 Screening experiment results and normal probability plot 44
4.2 Time slice one internode distance matrix 44
4.3 Time slice one internode travel-time matrix 45
4.4 Time slice two internode distance matrix 45
4.5 Time slice two internode travel-time matrix 46
4.6 Customer geographical locations 46
4.7 Tour one 48
4.8 Tour two 49

List of Tables

3.1 Solomon's VRPTW benchmarking problems 37
3.2 Algorithm repeatability 37

4.1 Case study problem customer data 43
4.2 Case study problem vehicle data 43
4.3 Case study problem results 47

List of Algorithms

1 Generate Initial Solution 32
2 Tabu Search 33
3 Identify Feasible Moves 34
4 Attempt to Exchange Nodes 35
5 Identify Best Move 35
6 Exchange Vehicle 36

Chapter 1

Introduction

1.1 Problem Statement

Inefficient vehicle fleets abound. In making several deliveries to customers which are displaced from one another, the most efficient path to be followed cannot be intuitively identified. This is especially true when the number of deliveries to be made is large. In identifying the best route to be followed by a single delivery vehicle, many questions will have to be answered. These questions include:

- In what order should the deliveries be made?
- Which routes should be taken from one delivery point to the next?

This problem has been identified as the travelling salesman problem (TSP). For small problem instances of the travelling salesman problem, it may be possible to intuitively identify the best route to be followed. As the size of a problem instance increases, one's intuition is likely to fail to identify the best route to be taken. When the number of deliveries to be made is large enough to require more than one vehicle to make deliveries, the complexity of the problem increases. The problem of segmenting the delivery load between the vehicles must also be considered. Partitioning the set of deliveries so that each vehicle can deliver to a subset is the additional step required. The questions which will have to be answered for this extended problem, the vehicle routing problem (VRP), include:

- How should the deliveries be divided between the vehicles of the fleet?
- In what order should the deliveries assigned to each vehicle be made?
- Which routes should be taken by each vehicle from one delivery point to the next?

The likelihood of identifying the most efficient delivery division, delivery schedule and routes to be taken using only one's intuition is small. This is part of the

reason why inefficient vehicle fleets abound. Another contributing factor is the exorbitant price of the software which is used to solve problems such as these. A cheaper software solution may give the managers of more vehicle fleets the opportunity to improve the efficiency of their fleets.

1.2 Potential Solution Strategies

In the establishment of the problem, in the *problem statement*, the problem has been identified as the VRP. The VRP is an example of a combinatorial optimisation problem. An array of potential solution strategies is available to solve problems within this class. The potential solution strategies at hand include exact methods, heuristic algorithms and metaheuristic algorithms. The VRP has been classified as an NP-hard problem. Winston and Venkataramanan [2003] note that NP-hard problems cannot be solved efficiently to optimality. Attempting to do so would require an inordinate amount of time. Therefore using approximating methods, such as heuristic methods and metaheuristic methods, for such problems is a required compromise. Metaheuristics generally produce better quality solutions than those produced by heuristic methods. Strangely enough, metaheuristics achieve this with their ability to accept degrading moves within the solution space of a problem instance. This ability allows metaheuristic algorithms to escape local optima within the search space of a problem instance. Metaheuristics have become the solution strategy of choice for many NP-hard combinatorial optimisation problems, including the VRP, because they produce high-quality solutions in an efficient manner.

1.3 Solution Strategy

In order to solve VRP instances which differ with respect to magnitude and details, metaheuristics are the most suitable solution strategy. Metaheuristic algorithms which have been designed to solve VRP instances require vehicle data, customer data and road network data as input. These algorithms make use of the data to identify a satisfactory solution. Most algorithms make use of a single set of internode travel-time and distance matrices. Such a strategy does not consider the variation in travel time caused by traffic. A preferred strategy, which considers the effect of traffic on travel time, utilises a set of time slices to model the difference in travel time brought about by traffic. Each time slice represents a period of similar travel time which is conveyed by its associated travel-time matrix. The strategy adopted for this project utilises a metaheuristic algorithm which receives the internode travel-time and distance matrices associated with each time slice within a set of time slices.

1.4 Project Scope

This project is a manifestation of the endeavour to curtail the inefficiency of vehicle fleets. The deliverable of this project is a programmed metaheuristic search algorithm to optimise a subset of vehicle routing problems. This algorithm will need to interface with AfriGIS data. AfriGIS is the geographical information and telecommunications solution company which has sponsored this project. The algorithm will receive distance and travel-time matrices for each time slice. These matrices are generated by a routing algorithm which interfaces with datasets containing customer and depot locations as well as the street network surrounding these nodes. Interfacing with this routing algorithm will allow for the distance and travel-time matrices to be communicated to the metaheuristic algorithm. The specific type of vehicle routing problem which the algorithm must be capable of solving includes the following factors:

- A single time window for each customer visit.
- Multiple vehicles.
- Multiple customer vertices (as many as 200).
- Delivery demand (in volume, weight or number of units) associated with each customer.
- One or multiple depots.
- Service time, which represents the off-loading time, associated with each customer.
- Capacity (in volume, weight or number of units) of each vehicle must be specified and heterogeneous vehicle capacity is assumed.
- The inclusion of time-dependent travelling times, along the arcs or edges of the road network, which will be dependent upon the time of day.
- Each customer must be served by one vehicle exactly once.

It is important to note that the travel times will be modelled as a function of the time and day. For instance, travelling a particular route during the week will most probably result in a travel time which differs considerably from the same route's weekend travelling time. The model will consider the varying travel times associated with the road network. Travelling times will be based upon historic traffic trends for a set of time slices, each of which consists of one or several hours with similar traffic conditions. Real-time traffic data will not be considered. All orders must be requested before the commencement of the day and thus a dynamic response model is not part of the scope of this project. It is assumed that each customer can be served by a single vehicle. This implies that the demand associated with each customer is at most equivalent to the capacity of an available vehicle.

1.4.1 Input

The algorithm will require the following as input:

- The time window associated with each delivery.
- The demand (in volume, weight or number of units) associated with each delivery.
- The service time associated with each delivery.
- The opening and closing time of the depot(s).
- The capacity (in volume, weight or number of units) of each vehicle.
- The fuel type of each vehicle (petrol or diesel).
- The fuel tank capacity of each vehicle (in litres of fuel).
- The starting and ending depot of each vehicle.
- Internode distance and travel-time matrices (one for each time slice).

1.4.2 Output

The output which the algorithm must generate includes:

- The customers (deliveries) assigned to each of the utilised vehicles in the scheduled order of delivery.
- The estimated time of arrival of each utilised vehicle at each of the customers assigned to it.

1.4.3 Dataset Interfaces

The IDEF∅ diagram below depicts the dataset interfaces and algorithm functions on a high level. The metaheuristic algorithm will utilise the output of an existing routing algorithm. The routing algorithm receives customer and depot locations and a set of time slices as input. The output of the routing algorithm is in the form of internode distance and travel-time matrices. A set of internode distance and travel-time matrices is generated for each time slice. It is the responsibility of the existing routing algorithm to interface with the datasets containing the customer and road network data. The internode distance and travel-time matrices are then utilised by the metaheuristic search algorithm in its search of the solution space of a problem instance.

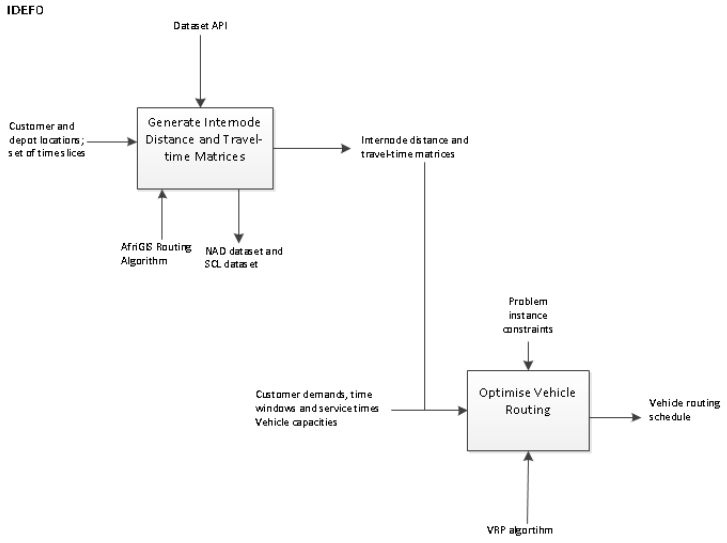


Figure 1.1: IDEF∅ diagram depicting dataset interfaces

1.5 Case Study

The following case study is an example of the sort of problem which the vehicle routing algorithm must be capable of solving. A case study similar to this one is presented and solved in Chapter 4.

Crema Press, a supplier of coffee and coffee-related products, makes use of its own fleet of vehicles to deliver to its clients. The outlets of the individual clients of Crema Press have different operating hours. Some clients also require deliveries to be made during time windows. The fleet of delivery vehicles is divided between three depots. The three depots are situated in Erasmusrand,

Bryanston and Mayfield. The fleet of delivery vehicles is made up of 20 vehicles. The 20 vehicles are divided between the three depots as follows: six vehicles are based at the Erasmusrand depot, seven vehicles are based at the Bryanston depot and the remaining seven vehicles are based at the Mayfield depot.The fleet is heterogeneous in terms of loading capacity. Vehicle loading capacity is measured in volume, weight or number of units. The Crema Press depots open at 5:00 on each morning during the week and close at 17:00. The depots are not operational over weekends. Several orders must be delivered each day. The number of orders to be delivered on a given day can easily be in excess of one hundred orders. This is especially true for days just prior to and just after weekends. Orders which will be delivered on a given day must be placed by 15:00 on the preceding day. This is to accommodate the administration and planning required for the delivery of each order.

Crema Press has identified travel cost as a cost component which must be improved. Optimising their vehicle routing and scheduling may well provide the increased travel efficiency which is desired. This vehicle routing and scheduling can be modelled as a combinatorial optimisation problem. A class of problems which definitely encompass the one described above are the vehicle routing problems. Some important characteristics of the problem described above are listed below:

- Heterogeneous vehicle fleet (in terms of loading capacity).
- Time windows for delivery.
- Multiple depots.
- Vehicles may have different departure and arrival depots.
- Vehicle loading capacity may not be exceeded at any point in time (no vehicle may be assigned to more customers than its loading capacity can accommodate).
- A particular client's order may not be divided between more than one delivery vehicle.

The specific vehicle routing problem variant which most resembles the problem described above is the multiple depot vehicle routing problem with time windows and a heterogeneous fleet. The objective of Crema Press is to incur the least possible travelling expense in delivering all orders. Travel expense is a weighted function of travel distance and travel time. Travel time or travel distance can be used to approximate travel expense. Therefore, in modelling this problem, minimising travel time, travel distance or a weighted combination of the two measures can meet the objective of Crema Press. In estimating travel time, traffic trends should be approximated. A simple method of estimating traffic trends involves estimating the time-dependent travel time along potential routes. This involves dividing the working day (that part of the day in which Crema

Press executes its business activities) into several time slices. This must be done so that the travel time along a route for a given time slice is approximately homogeneous. The implementation of time-dependent travel times takes traffic conditions into consideration. The use of as few as three to five time slices can divide the operational day into slices of different traffic activity and yield more accurate estimations of travel time.

1.6 Considerations

The vehicle routing algorithm to be developed must be capable of accommodating input which varies in its details and size. Therefore the algorithm must be in a generic form which will enable it to process different sets of input.

1.7 Document Structure

Chapter one includes the definition of the problem statement and the project deliverables. Chapter two provides a review of literature related to the vehicle routing problem and its solution strategies. Chapter three offers the formulation of the problem's mathematical model, the presentation of the solution algorithm and a selection of the verification activities undertaken. Chapter four considers the implementation of the metaheuristic algorithm, offers a discussion on parameter tuning and presents a case study problem and the solution identified by the metaheuristic algorithm.

Chapter 2

Literature Review

2.1 Introduction

The vehicle routing problem involves assigning a fleet of vehicles to a set of orders, which constitute geographically distributed points of collection or delivery, in the most cost-efficient way which conforms to the defined constraints. Mathematical models have been developed to represent the various forms of the vehicle routing problem. A model which represents the general vehicle routing problem follows [Joubert, 2003]. A set of K identical vehicles, each with capacity p, is available for the purposes of making deliveries to N customers. Customer $i \in I = \{1, \ldots, N\}$ has a known demand of quantity q_i. A vehicle's capacity may not be exceeded by the sum of the demand of the customers to which the vehicle must deliver. t_{ij} represents the travel time between nodes i and j where $i, j \in \{1, 2, \ldots, N\}$. c_{ij} represents the cost associated with travelling on the arc between nodes i and j where $i, j \in \{1, 2, \ldots, N\}$. d_{ij} represents the travel distance of the arc between nodes i and j where $i, j \in \{1, 2, \ldots, N\}$.

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k=\{1,\ldots,K\} \text{ travels from node } i \text{ to node } j \\ & \text{with } i,j=\{1,2,\ldots,N \mid i \neq j\} \\ 0 & \text{otherwise} \end{cases}$$

$$\min z = \sum_{i=0}^{N} \sum_{j=0; j \neq i}^{N} \sum_{k=1}^{K} c_{ij} x_{ijk} \tag{2.1}$$

subject to:

$$\sum_{j=1}^{N} x_{0jk} = \sum_{j=1}^{N} x_{j0k} = 1 \qquad \forall k = \{1, 2, \ldots, K\} \tag{2.2}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K} x_{0jk} \leq K \tag{2.3}$$

$$\sum_{i=1;i\neq j}^{N} \sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall j \in \{1, 2, \ldots, N\} \tag{2.4}$$

$$\sum_{j=1;j\neq i}^{N} \sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall i \in \{1, 2, \ldots, N\} \tag{2.5}$$

$$\sum_{i=1}^{N} q_i \sum_{j=0;j\neq i}^{N} x_{ijk} \leq p \qquad \forall k \in \{1, 2, \ldots, K\} \tag{2.6}$$

$$x_{ijk} \in \{0, 1\} \tag{2.7}$$

The objective function of the model strives to minimise the cost associated with satisfying all customer demand. c_{ij} in (2.1) may be replaced with t_{ij} to minimise the total travel time or with d_{ij} to minimise the total travel distance associated with satisfying all customer demand. (2.2) ensures that all routes start and end at the depot (node 0). (2.3) disallows exceeding the maximum number of vehicles/routes. (2.4) and (2.5) ensure that each customer node is visited only once. (2.6) ensures that the capacity of each vehicle is not exceeded by the demand of the customers which it services.

2.2 Problem Variants

Several forms of the vehicle routing problem exist. Toth and Vigo [2002] note that important variants of the vehicle routing problem include the capacitated vehicle routing problem (CVRP), the capacitated and distance-constrained vehicle routing problem (DCVRP), the vehicle routing problem with time windows (vehicle routing problem with time windows (VRPTW)), the vehicle routing problem with backhauls (VRPB) and the vehicle routing problem with pickup and delivery (VRPPD). Various combinations of these form yet more hybrid vehicle routing problem variants such as the vehicle routing problem with backhauls and time windows (VRPBTW) and the vehicle routing problem with pickup, delivery and time windows (VRPPDTW). The multiple depot vehicle routing problem (MDVRP) is yet another noteworthy problem variant. These forms differ in terms of their simplifying assumptions and model structures. Several variants of the vehicle routing problem are discussed briefly below.

2.2.1 Capacitated Vehicle Routing Problem

In the capacitated vehicle routing problem (CVRP), Toth and Vigo [2002] note that the demand and deliveries which correspond to customers are deterministic, known in advance and may not be divided/shared. The homogeneous fleet of vehicles is based at one central depot and a capacity constraint is imposed upon all of the vehicles. The objective is to minimise the total cost to serve all of the customers. The cost may be viewed as a weighted function of the number of routes and their length or travel time.

2.2.2 Vehicle Routing Problem with Time Windows

Toth and Vigo [2002] note that the vehicle routing problem with time windows (VRPTW) is an extension of the CVRP in which capacity constraints are imposed and a time interval $[a_i, b_i]$ of service opportunity for each customer $i \in \{1, \ldots, N\}$ is applied. This time interval of service opportunity is known as a *time window*. The time instant in which the vehicles depart from the depot, a service time s_i for each customer $i \in \{1, \ldots, N\}$ and the travel time t_{ij} for each arc (i, j) joining the set of customer nodes $i, j \in \{1, \ldots, N\}$ are important elements within the VRPTW. The service of each customer must commence within the applicable time window and the vehicle must remain at a customer node for s_i time instants. The time at which the vehicles depart from the depot is defined as time 0. The VRPTW involves finding a collection of exactly K circuits with the lowest cost such that the following constraints are conformed to.

- Each circuit visits the depot vertex.
- Each customer vertex is visited by exactly one circuit.
- The sum of the demands of the vertices visited by a circuit does not exceed the capacity of the vehicle.
- The service of each customer i commences within the time window $[a_i, b_i]$ and the vehicle remains at each customer vertex for s_i time instants.

2.2.3 Vehicle Routing Problem with Backhauls

Toth and Vigo [2002] note that the vehicle routing problem with backhauls (VRPB) involves two subsets of customers: linehaul customers and backhaul customers. Linehaul customers require a quantity of product to be delivered while backhaul customers require a quantity of product to be picked up. d_i is the nonnegative demand to be delivered or picked up at customer i and a fictitious demand d_0 is associated with the depot vertex. The VRPB involves finding the least-cost collection of exactly K circuits such that the following constraints are conformed to.

- Each circuit visits the depot vertex.

- Each customer vertex is visited by exactly one circuit.
- The total demands of the linehaul and backhaul customers on each circuit, separately, do not exceed the capacity of the vehicles.
- For all circuits involving both linehaul and backhaul customers, all linehaul customers precede backhaul customers.

2.2.4 Vehicle Routing Problem with Pickup and Delivery

The vehicle routing problem with pickup and delivery (VRPPD) associates two quantities with each customer i, d_i and p_i , which represent the demand of homogeneous commodities to be delivered and picked up respectively. With each customer i there are two associated vertices: an origin vertex of delivery demand, O_i, and a destination vertex of pickup demand, D_i. It is assumed that at each customer location, delivery is executed prior to pickup. The load of a particular vehicle, at a given point in time, is defined as its initial load, less the sum of all demands already delivered, plus the sum of all demands already picked up. The VRPPD involves finding the least-cost collection of exactly K circuits such that the following constraints are conformed to.

- Each circuit visits the depot vertex.
- Each customer vertex is visited by exactly one circuit.
- The load of the vehicle, at all times, must be positive and may not exceed its capacity.
- For each customer i, the customer O_i, when different from the depot, must be visited in the same circuit and before customer i.
- For each customer i, the customer D_i, when different from the depot, must be visited in the same circuit and after customer i.

2.2.5 Multiple Depot Vehicle Routing Problem

The multiple depot vehicle routing problem (MDVRP) is the extension of the vehicle routing problem in which the vehicles of the fleet may be initially located at different depots. The constraint which requires each vehicle to start and end at the same depot is often imposed. The less frequently encountered variant of the problem in which vehicles may initially depart from and finally arrive at different depots is applicable to some multiple depot vehicle routing problem instances. Set partitioning of customers is often utilised in multiple depot vehicle routing problems.

2.2.6 Applications

The constraints used to distinguish the variants of the vehicle routing problem can significantly impact the problem's solution. For instance, Rizzoli et al. [2007] have included an interesting figure in their journal article which illustrates the inversely proportional relationship between the number of tours required and the width of time windows in the VRPTW. For a particular VRPTW problem instance, when the time window width is 10 minutes over 85 tours are required. The same problem instance requires less than 55 tours when the time window is 120 minutes.

According to Rizzoli et al. [2007], the objective of supply chain systems is to distribute goods across the logistics network in the most cost efficient manner. As a result, algorithms which endeavour to solve vehicle routing problems tend to minimise measures which increase cost such as time spent or distance travelled. Due to the fact that the vehicle routing problem is a complex combinatorial problem, the difficulty of finding an exact solution increases with the size of the problem. The solution space of an NP-hard problem increases at a rate in excess of a polynomial rate as the number of customers or vertices increases. Even small instances of some of the models which make use of simplifying assumptions require metaheuristic methods to be solved in a reasonable amount of time. Many of the VRP models encountered in literature are deterministic in nature. Such a simplifying assumption yields models which are significantly less complex than their probabilistic counterparts. This may facilitate the solution of such models but such assumptions do impair the accuracy with which the models reflect the real problem at hand. Nonetheless, reasonably large complex dynamic vehicle routing problem models have been solved satisfactorily (not to optimality) within very acceptable amounts of time. One such model has been presented by Goel and Gruhn [2005]. This model incorporates a diversity of practical complexities such as time window restrictions, a heterogeneous vehicle fleet with different travel times, travel costs and capacity, multi-dimensional capacity constraints, order/vehicle compatibility constraints, orders with multiple pickup, delivery and service locations, different start and end locations for vehicles, route restrictions associated to orders and vehicles, and drivers' working hours. The algorithm incorporated Large neighbourhood Search, which is capable of handling the complexities of the model, and Goel and Gruhn note that their computational experiments performed well for instances with hundreds of vehicles and several hundreds of transportation requests with response times frequently less than a second. This illustrates that yielding satisfactory solutions to NP-hard problems is achievable.

Rizzoli et al. [2007] comment on the application of a time-dependent vehicle routing problem with time windows. In this model, the travel time along the arcs is dependent upon the time of day. It is assumed that during the day, there are some distinctive time slices in which the travel times are similar. It is assumed that the working day can be divided into l time slices, where the

element $t_{ij}(l)$ represents the travel time from node i to node j during time slice l. $l \in T_l$ where T_l is the set of time slices into which the working horizon is split. The objective is to minimise the total travel time. The particular problem instance discussed by Rizzoli et al. [2007], which involves 30 customers, has been solved using an ant colony optimisation metaheuristic algorithm. Another approach of modelling non-deterministic travel times involves using stochastic travelling times with known probability distributions. Such an approach has been demonstrated by Taş et al. [2012].

2.3 Metaheuristics

Several methods have been used to solve the various variants of the vehicle routing problem. These methods include heuristic methods and metaheuristic methods. Metaheuristic methods differ from heuristic methods in that metaheuristic methods include a mechanism which will allow non-improving solutions to be accepted. The mechanism used to accept non-improving solutions differs between most metaheuristic types but serves the same purpose. The purpose of this mechanism is to enable a more extensive search of the solution space. This mechanism has been aptly named the diversification mechanism. Heuristic methods do not incorporate diversification mechanisms. Heuristic methods have been frequently used on classical vehicle routing problem variants. According to Gendreau et al. [2007], heuristic methods were used frequently in the past to solve complex combinatorial optimisation problems. The classical variants of the vehicle routing problem often involve sets of simplifying assumptions. Heuristic methods have been successful with small and intermediate instances of the problem. According to Gendreau et al. [2007], since the development of metaheuristics by Glover in 1986, metaheuristic methods have largely replaced heuristic methods in the field of complex combinatorial optimisation. Larger instances of the vehicle routing problem often require metaheuristic methods to yield acceptable, yet suboptimal, solutions within a reasonable amount of time. For large instances of the problem involving practical complexities (fewer simplifying assumptions), metaheuristic methods must be recruited to yield acceptable solutions within a reasonable amount of time. Several types of metaheuristic search algorithms exist. Popular metaheuristic search algorithms include:

- Tabu search
- Simulated annealing
- Ant colony optimisation
- Genetic algorithm
- Greedy randomised adaptive search procedure (GRASP)
- Variable neighbourhood search (VNS)
- Neural networks

- Particle swarm optimisation

Metaheuristic algorithms have become the solution strategy of choice for VRPs. Tabu search and ant colony optimisation methods have shown some notable success with certain variants of the vehicle routing problem. Hybrid metaheuristic search algorithms have also been developed and successfully applied to VRPs.

A common process is followed by the different types of metaheuristic algorithms [Joubert, 2006]. Metaheuristic algorithms include three components: an initialisation mechanism, a diversification mechanism and an intensification mechanism. Initialisation is the process of finding an initial solution. Simple heuristics are often utilised to generate initial solutions. Diversification involves an extensive search of the solution space. Intensification involves searching promising areas of the solution space more thoroughly. Another component which is common to the different types of metaheuristic algorithms is some stopping criterion which terminates the algorithm.

Metaheuristic search methods often emulate processes found in nature, biological processes or physical processes. Neural networks mimics the behaviour of neurons in the brain to infer previous experiences in the decision-making process [Winston and Venkataramanan, 2003]. Genetic algorithms emulate evolution by incorporating processes which simulate reproduction, mutation and natural selection. Simulated annealing, Tabu search and ant colony optimisation are discussed briefly below.

2.3.1 Simulated Annealing

Simulated annealing (SA) is a Monte Carlo approach developed by Metropolis et al. (1953) (cited in Winston and Venkataramanan [2003]) that could be used to simulate the behaviour of atoms in achieving thermal equilibrium at a given temperature. In assuming the goal is to find a minimum energy-level configuration, a randomly generated perturbation of the structure of a current atomic configuration (energy state E_0) is applied. A perturbation applied to the structure results in a new energy state E_i. If a perturbation results in a lower energy at state i ($E_i < E_0$), the process is repeated using the new energy state. Where a higher energy state results ($E_i > E_0$), its acceptance is based upon a certain probability. It is this ability to accept a non-improving structure which enables the technique to leave local optima. Simulated annealing emulates the physical process of aggregating particles in a system as it is cooled. The energy exchange is simulated by changing from one neighbourhood solution to another. Boltzman's equation is used in deciding whether a change in state is accepted. For the Simulated Annealing algorithm, the melting temperature is called the initial temperature and the steps in decreasing the temperature are collectively

known as the cooling schedule. The initial state is any initial solution. Temperature is used as a control parameter in deciding whether suboptimal solutions are accepted during the solution space search. The probability of accepting a suboptimal solution is calculated using $e^{(\Delta C/T)}$, where T is the temperature and $\Delta C =$ (best objective function value $-$ current objective function value) is the change in the value of the objective function. It is important to note that in a minimisation problem, $\Delta C < 0$ and that as the value of T decreases, the probability of accepting a non-improving move decreases exponentially.

2.3.2 Tabu Search

Winston and Venkataramanan [2003] note that Tabu search makes use of both short- and long- term memory to forbid certain moves in the solution space of a problem. Short-term memory forbids cycling around a local neighbourhood in the solution space and it allows the search to move away from local optima. Long-term memory allows promising neighbourhoods to be searched thoroughly. Tabu search is a memory-based algorithm which has a lot in common with the rules applied by humans in daily decision making. Important components of a Tabu search algorithm are short-term memory Tabu rules and list size, long-term memory Tabu rules and list size, Tabu tenure, candidate list of moves, aspiration criteria, intensification, diversification and strategic oscillation. The short-term memory list size determines the number of forbidden moves in the Tabu list. Similarly, the long-term memory list size determines the number of forbidden moves in its Tabu list. The Tabu rules determine the members of the Tabu lists. Aspiration criteria measure the quality of a move in solution space and provide a mechanism to override Tabu lists. The candidate list offers moves which the algorithm may evaluate in its search. Intensification is associated with long-term memory and it promotes the searching of promising regions. An intensification strategy can include an elite list of candidates representing attractive regions. Diversification allows searching of less attractive regions and avoids cycling in attractive neighbourhoods by allowing unattractive moves in the short-term memory. Strategic oscillations involve executing intensification and diversification strategies in an alternating fashion around a target boundary.

2.3.3 Ant Colony Optimisation

Rizzoli et al. [2007] note that ant colony optimisation mimics the way in which foraging ants communicate, using pheromone trails, the shortest path to food sources. Ants lay pheromone trails, in varying quantities, as they move. When such a trail is encountered by an ant, it may choose to follow it (highly probable) and in turn strengthen it with its own pheromone. A positive feedback loop results from the collective behaviour of the ants. The more a particular trail has been reinforced by pheromones, the more attractive it becomes to ants. The main elements of these algorithms are artificial ants, which are simple computational agents that individually and iteratively construct solutions to the problem. In the case of the vehicle routing problem, the problem has been

modelled as a graph. Ants explore the graph by visiting nodes, which are connected by edges, in their pursuit of finding a solution. A solution is an ordered sequence of nodes. A dynamic memory structure which mimics the pheromone trail concept supplies information about the quality of previously obtained results and guides the search. Intermediate partial problem solutions are viewed as states. At every iteration k, each ant moves from state $x_k^{(i)}$ to $x_{k+1}^{(j)}$ and in turn extends the partial solution from node i to j. The algorithm can be organised into two main stages: the construction of a solution and the update of the pheromone trail. Each ant constructs a solution. In a given state, each ant determines a set of feasible expansions from it. The move selected by each ant is made by taking the following values into account. η_{ij}, the attractiveness of the move as calculated by some heuristic (*a priori* desirability of the move). τ_{ij}, the pheromone trail level, which indicates how useful this move has been in the past (*a posteriori* desirability of the move). An ant considers these two values in making a move from node i to j according to the following probability:

$$p_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in \Omega} [\tau_{ih}]^\alpha [\eta_{ih}]^\beta} & \text{if } j \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

where Ω is the set of nodes which can be visited from i. α and β weigh the influence of trails and attractiveness. Pheromone trails are updated upon reaching solutions. Initially, all pheromone trails are evaporated to forget bad solutions. The ants then deposit pheromone on the arcs which form part of their solutions.

2.3.4 Application

An example of a successful practical application of a metaheuristic algorithm to a vehicle routing problem which involves time windows (VRPTW) is discussed by Rizzoli et al. [2007]. This problem also incorporated a fleet of heterogeneous vehicles. The application involved the use of an ant colony optimisation metaheuristic to facilitate the planning of the distribution of palletised goods to over 600 stores distributed across Switzerland. Prior to the experimentation with the algorithm for planning purposes, human planners would devise distribution routing plans. The results of the experimentation indicated the superior planning capability of the metaheuristic for complex distribution problems. The time required for the algorithm to generate a feasible solution was just five minutes. The human planners required three hours to plan a feasible solution for the same problem. The metaheuristic found a solution to a particular problem instance which utilised, on average, 87.35% of vehicle loading capacity and required 1807 tours with a total distance travelled amounting to 143983 km. The human planners' solution utilised, on average, 76.91% of vehicle loading capacity, required 2056 tours with a total distance travelled of 147271 km. Both of these solutions for the particular problem instance are capable of delivering all of the orders associated with the problem instance. However, the solution proposed by the metaheuristic algorithm is considerably more efficient. This example demonstrates the value of metaheuristic algorithms in the

planning of complex distribution activities. The potential savings which can be realised when making use of metaheuristic algorithms for routing problems are significant.

2.4 Conclusion

This chapter provides a definition of the vehicle routing problem and introduces several of its variants. Metaheuristic methods are introduced and various types of metaheuristic algorithms are briefly described. This chapter includes descriptions of practical applications of metaheuristic algorithms to various vehicle routing problems.

Since a generic vehicle routing algorithm is required, the vehicle routing problem type which is most representative of the problems which will be solved using the algorithm must be identified. The algorithm is intended to receive, as input, problems with as many as twenty vehicles. The algorithm must accommodate time windows, multiple depots and a fleet of vehicles with heterogeneous capacity. The vehicle routing problem upon which the algorithm should be based is the multiple depot vehicle routing problem with time windows (multiple depot vehicle routing problem with time windows (MDVRPTW)). The consulted literature promotes the use of a Tabu search algorithm to solve vehicle routing problems with time windows.

Chapter 3

Model Formulation

In this chapter, the specific type of vehicle routing problem which the solution algorithm is designed to solve is defined. The formulation of the solution algorithm is also presented. Some analyses of the performance of the solution algorithm appear towards the end of this chapter. Finally, the solution algorithm is tested on several benchmark problems as part of the verification process.

3.1 The Problem

The problem is described and its characteristics are stated in the *Project Scope* section in Chapter 1. This section offers a review of the problem characteristics and the mathematical formulation of the problem's model.

3.1.1 Problem Characteristics

As elaborated upon in Chapter 1, the inefficiency of delivery vehicle fleets is a concern. In order to improve the efficiency of delivery vehicle fleets, an affordable metaheuristic vehicle routing problem algorithm has been identified as the preferable solution method. A generic metaheuristic algorithm which can accommodate a broad spectrum of vehicle routing problem instances is required. In order to accommodate a broad spectrum of vehicle routing problem instances, the characteristics which are common to many vehicle routing problem instances must be identified. These characteristics must heavily influence the structure of the problem's model. The characteristics which are representative of the sort of problem which will be solved by the metaheuristic algorithm follow:

- Time windows for each customer visit.
- Multiple vehicles.
- Multiple customers.

- Delivery demand (in volume, weight or units) associated with each customer.
- One or multiple depots (in the event of multiple depots, vehicles may end at a depot which differs from their starting depot).
- Service time, which represents the off-loading time, associated with each customer.
- Capacity (volume, weight or units) of each vehicle must be specified: heterogeneous vehicle capacity is assumed but the metaheuristic algorithm must be capable of solving problems with homogeneous vehicle capacity.
- The inclusion of time-dependent travelling times, along the arcs or edges of the road network, which are dependent upon the time of day.
- Each customer should be served by one vehicle exactly once.

From these problem characteristics, it is clear that the metaheuristic algorithm must possess a considerable measure of flexibility in terms of the problem factors which it must be capable of accommodating.The spectrum of problem instances which the metaheuristic algorithm is expected to solve can differ in several ways:

- Problem instance size: the number of customers and vehicles.
- Problem type: single depot or multiple depot; heterogeneous vehicle capacity or homogeneous vehicle capacity.
- Restrictiveness: accommodating or limiting time windows and constraints.

3.1.2 Mathematical Model

The mathematical model used to represent the described problem follows.

Set definitions:

$\bar{I}$ $\triangleq$ the set of all customers and depots

$\bar{Q}$ $\triangleq$ the set of all depots, where $\bar{Q} \subset \bar{I}$

$\bar{Q}^c$ $\triangleq$ the complement of $\bar{Q}$ which includes only customers and none of the depots, $\bar{Q}^c \subset \bar{I}$

$\bar{K}$ $\triangleq$ the set of all vehicles

$\bar{T}$ $\triangleq$ the set of all time slices

Let:

q_i be the known demand for location i, where$i \in \bar{Q}^c$

s_i be the service time for location i, where$i \in \bar{I}$

d_{ijt} be the distance between location i and j during time slice t, where $i, j \in \bar{I}, t \in \bar{T}$

c_{ijt} be the cost incurred on the arc between locations i and j during time slice t, where $\{i, j \in \bar{I} \mid i \neq j\}, t \in \bar{T}$

t_{ijt} be the travel time from location i to location j during time slice t, where$i, j \in \bar{I}, t \in \bar{T}$

p_k be the load capacity of vehicle k, where $k \in \bar{K}$

f_k be the fixed cost incurred when vehicle k is utilised, where $k \in \bar{K}$

a_i be the arrival time of the assigned vehicle at location i, where$i \in \bar{I}$

w_i be the waiting time of the assigned vehicle at location i, where $i \in \bar{I}$

e_i be the commencement time of the time window of location$i \in \bar{I}$

l_i be the termination time of the time window of location$i \in \bar{I}$

The decision variable, x_{ijk}^t:

$$x_{ijk}^t = \begin{cases} 1 & \text{if vehicle } k \in \bar{K} \text{ travels from location } i \in \bar{I} \text{ to location } j \in \bar{I}, \\ & i \neq j, \text{ during time slice } t \in \bar{T} \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \min z = & \sum_{i \in \bar{I}} \sum_{j \in \bar{I}, j \neq i} \sum_{k \in \bar{K}} \sum_{t \in \bar{T}} c_{ijt} x_{ijk}^t + \sum_{i \in \bar{Q}} \sum_{j \in \bar{Q}^c} \sum_{k \in \bar{K}} \sum_{t \in \bar{T}} f_k x_{ijk}^t \\ & + \mu \left(\sum_{i \in \bar{Q}^c} 1 - \sum_{k \in \bar{K}} \sum_{i \in \bar{I}} \sum_{j \in \bar{Q}^c, j \neq i} \sum_{t \in \bar{T}} x_{ijk}^t \right) \end{aligned} \tag{3.1}$$

subject to:

$$\sum_{i \in \bar{Q}} \sum_{j \in \bar{Q}^c} \sum_{k \in \bar{K}} \sum_{t \in \bar{T}} x_{ijk}^t = \sum_{i \in \bar{Q}^c} \sum_{j \in \bar{Q}} \sum_{k \in \bar{K}} \sum_{t \in \bar{T}} x_{ijk}^t \leq \sum_{k \in \bar{K}} 1 \tag{3.2}$$

$$\sum_{i\in\bar{Q}}\sum_{j\in\bar{I}}\sum_{t\in\bar{T}} x_{ijk}^{t} = \sum_{i\in\bar{I}}\sum_{j\in\bar{Q}}\sum_{t\in\bar{T}} x_{ijk}^{t} \leq 1 \qquad \forall k \in \bar{K} \tag{3.3}$$

$$\sum_{j\in\bar{I}, j\neq i}\sum_{k\in\bar{K}}\sum_{t\in\bar{T}} x_{ijk}^{t} \leq 1 \qquad \forall i \in \bar{Q}^{c} \tag{3.4}$$

$$\sum_{i\in\bar{I}, i\neq j}\sum_{k\in\bar{K}}\sum_{t\in\bar{T}} x_{ijk}^{t} \leq 1 \qquad \forall j \in \bar{Q}^{c} \tag{3.5}$$

$$\sum_{i\in\bar{Q}^{c}} q_i \sum_{j\in\bar{Q}^{c}, j\neq i}\sum_{t\in\bar{T}} x_{ijk}^{t} \leq p_k \qquad \forall k \in \bar{K} \tag{3.6}$$

$$\sum_{i\in\bar{Q}}\sum_{t\in\bar{T}}\sum_{k\in\bar{K}} x_{ijk}^{t} = 0 \qquad \forall j \in \bar{Q} \tag{3.7}$$

$$\sum_{j\in\bar{Q}}\sum_{t\in\bar{T}}\sum_{k\in\bar{K}} x_{ijk}^{t} = 0 \qquad \forall i \in \bar{Q} \tag{3.8}$$

$$x_{ijk}^{t} \in \{0,1\} \tag{3.9}$$

$$a_q = w_q = s_q = 0 \qquad \forall q \in \bar{Q} \tag{3.10}$$

$$e_i \leq (a_i + w_i) \leq l_i \qquad \forall i \in \bar{I} \tag{3.11}$$

$$\sum_{i\in\bar{I}, i\neq j}\sum_{k\in\bar{K}}\sum_{t\in\bar{T}} x_{ijk}^{t}(a_i + w_i + s_i + t_{ij}) \leq a_j \qquad \forall j \in \bar{I} \tag{3.12}$$

The objective function, (3.1), seeks to minimise the cost associated with serving a set of customers. The first set of summations calculates the travelling cost associated with visiting a set of customers. The second set of summations calculates the total fixed cost incurred by making use of a subset of the fleet of available vehicles or the entire fleet of available vehicles in visiting a set of customers. If a vehicle is used, its associated fixed cost will be included in the total fixed cost. The third collection of summations calculates the number of unvisited customers and applies a penalty cost, μ, to each unvisited customer. (3.2) ensures that the number of vehicles which depart from the depot(s) equates to the number of vehicles which return to the depot(s) and that this number is less than or equal to the number of available vehicles. (3.3) ensures that if a vehicle is used, its tour starts and ends at a depot. (3.4) and (3.5) disallow customers from being visited more than once. (3.6) avoids vehicle load capacity violations. (3.7) and (3.8) disallow tours which include direct trips from one depot to another depot. (3.9) defines the decision variable as a binary variable. (3.10) ensures that the waiting time and service time at every depot is zero. This constraint also ensures that the vehicles at each depot are present from the moment the depots open. (3.11) ensures that time window impositions are not violated. (3.12) ensures that the modelling of time is valid. It requires the arrival time at a destination node to be later than or equal to the sum of the arrival time at its departure node, the wait time at the departure node, the service time at the departure node and the travel time. This constraint ensures that vehicles are not modelled with the ability to be in more than one place at a given instant.

3.2 Solution Algorithm

The various components and concepts of the solution algorithm appear below.

3.2.1 Important Concepts

3.2.1.1 Time Window Compatibility

The concept of time window compatibility was introduced by Joubert [2003]. Time window compatibility is a measure of the compatibility of the time windows of a pair of customers. A pair of customers is said to be time window compatible if there exists a time opportunity to feasibly juxtapose these customers while considering travel time, service time and the time windows of the customers. A pair of customers can and probably will have varying degrees of time window compatibility depending on the visitation order.

3.2.1.2 Insertion Criteria

The insertion criteria is a measure of the additional distance and time incurred by inserting a particular customer into a specific position within a partially constructed tour. The calculation of the insertion criteria is shown by (3.13), (3.14) and (3.15). (3.14) calculates the additional distance incurred due to the insertion of customer u between customers i and j. (3.15) calculates the additional time incurred due to the insertion of customer u between customers i and j. (3.13), the insertion criteria, is a weighted average of the additional distance and time incurred due to the introduction of customer u between customers i and j. α_1 and α_2, in (3.13), are the weighting coefficients while a_j^{new}, in (3.15), is the post-insertion arrival time at customer j. (3.14) makes use of time slices to determine the distance between two nodes. This is done due to the fact that different routes may be taken during different time slices due to traffic conditions. For each term in (3.14), the time slice t must contain the departure time from the initial node (which is indicated by the first subscript). For instance: the distance, d_{ijt}, is the distance of the path chosen during time slice t from node i to node j. The time slice t in this instance must contain the departure time from node i. It is important to note that the insertion criteria is only calculated for inserts involving three time window compatible customers. This avoids unnecessary computational exertion.

$$\text{Insertion criteria:} \quad c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) \tag{3.13}$$

$$\text{where} \quad c_{11}(i,u,j) = d_{iut} + d_{ujt} - \mu d_{ijt}, \quad \mu \geq 0 \tag{3.14}$$

$$c_{12}(i,u,j) = a_j^{new} - a_j \tag{3.15}$$

3.2.1.3 Selection Criteria

The selection criteria is a measure of the benefit of inserting a particular customer into a specific position within a partially constructed tour as opposed to introducing a new direct route from one of the depots to this customer. The

selection criteria, for a vehicle routing problem involving a single depot, is expressed mathematically in (3.16).

$$c_2(i,u,j) = \lambda(d_{0ut} + t_{0ut}) + s_u - c_1(i,u,j), \quad \lambda \geq 0 \tag{3.16}$$

(3.16) is not relevant to vehicle routing problems which involve multiple depots. The following definitions are required in order to establish the selection criteria for multiple depot vehicle routing problems:

$\bar{P}$ the set of available (unassigned vehicles)
g_p the starting depot of vehicle $p \in \bar{P}$
h_p the ending depot of vehicle $p \in \bar{P}$

$$c_2(i,u,j) = \frac{\sum_{p\in\bar{P}}(d_{g_put_1} + t_{g_put_1} + d_{uh_pt_2} + t_{uh_pt_2})}{\sum_{p\in\bar{P}} 1} - c_1(i,u,j) \tag{3.17}$$

Time slice t_1 must contain the departure time from depot g_p, which equates to: $e_{g_p} + s_{g_p}$. Time slice t_2 must contain the departure time from customer u, which equates to: $e_{g_p} + s_{g_p} + t_{g_put_1} + s_u$. Much like (3.16), (3.17) is a measure of the benefit of inserting a particular customer, u, between customers i and j in a partially constructed tour as opposed to starting a new tour for customer u. (3.17) however considers all the potential depots which could be the starting point and ending point of a new dedicated tour to customer u. (3.17) avoids an inflated first term by taking the average of the distance and travel time of the possible new tours to customer u. (3.17) serves the same purpose as (3.16). (3.17) is just the multiple depot analogue of (3.16).

3.2.2 Initial Solution Algorithm

Algorithm 1 is a sequential insertion algorithm which has been adapted from the initial solution algorithm presented by Joubert [2003]. This algorithm generates an initial solution, which serves as the input to the Tabu search metaheuristic algorithm. This algorithm requires as input the customer, depot and vehicle data pertaining to a particular problem instance. The algorithm, after initialising a new tour, selects a vehicle randomly from the fleet of available vehicles. The starting and ending depot of the vehicle assigned to the tour are inserted as the start- and end- locations of the tour. The algorithm then identifies and inserts a seed customer into the tour.

The algorithm identifies as the seed customer the least time window compatible customer amongst the set of customers which have yet to be assigned to a tour. This is done by considering the time window compatibility of each unassigned customer with all other unassigned customers to determine the *total time window compatibility* for the unassigned customer in concern. The time window compatibility for each pair of unassigned customers will be considered twice due to the difference in the time window compatibility when the visitation sequence is changed for a given pair of customers. The least time window compatible customer is the one with the smallest *total time window compatibility.*

Once the seed customer has been inserted into the tour, the algorithm continues to attempt to insert customers into the tour until customers can no longer be feasibly inserted into the tour because of time window or vehicle load capacity constraints. The algorithm identifies for each unassigned customer the best feasible insert location. This is achieved by considering all potential insert locations for a given unassigned customer. The insertion criteria for each of these insert locations is calculated if the customer being considered for insertion is time window compatible with the customers between which it could be inserted. The insert location with the lowest insertion criteria is considered as the best insertion for a given unassigned customer. Insertion criteria is used in Algorithm 1 to compare the feasible inserts of a given unassigned customer (line 12). The best insertion for the unassigned customer in concern is then identified in line 13 of Algorithm 1.

The best feasible insert location for each customer is considered by the algorithm in its selection of the customer to be inserted into the tour. This is achieved by considering the best insertion of each unassigned customer and calculating its selection criteria. The insertion with the largest selection criteria is considered as the best insertion (line 15 of Algorithm 1). The insertion which is considered as the best by the selection criteria is inserted into the current tour. The algorithm continues to insert customers into the tour, which is under construction, in this manner. Once no additional customer can be feasibly inserted into the tour, the algorithm considers the tour as complete and initialises another tour. The algorithm will continue to add tours to the initial solution while at least one customer has yet to be assigned to a tour and at least one vehicle is still available. Vehicles become unavailable upon being assigned to a single tour.

3.2.3 Tabu Search Algorithm

The Tabu search metaheuristic algorithm, Algorithm 2, receives the initial solution, which is generated by Algorithm 1, as input. The incumbent solution is the best solution found thus far by the algorithm. When the Tabu search algorithm is first initiated, the initial solution is the best solution. The Tabu search algorithm starts with the initial solution as the current solution. The algorithm applies *changes* to the current solution in an attempt to yield a better solution. These *changes* take the form of moving a single customer from a certain position in one tour to another position in the same or an alternative tour as indicated in Algorithm 3. Algorithm 3 also adds feasible moves involving the insertion of unassigned customers into tours to the set of *potential moves*. This will only be attempted if unassigned customers exist within the current solution. Only moves which maintain the feasibility of the solution are considered as potential moves. The non-tabu move[1] in the set of *potential*

[1] Xu et al. [1998] note that a move is classified tabu when it is not permitted as a result of the tabu status of its attributes.

moves which has the most preferential effect on the cost of the current solution is selected as the *best move*. The algorithmic representation of the identification and selection of the best move is displayed in Algorithm 5. In the event that no feasible non-tabu move can be identified, the algorithm attempts to exchange two randomly selected nodes in the current solution. This is done in the hope that the exchange will introduce feasible moves. The node exchange algorithm is displayed in Algorithm 4. After executing a move or an exchange on the current solution, the objective function value (cost) of the new current solution is calculated. The algorithm also attempts to exchange vehicles between tours or exchange an available vehicle for a vehicle which is assigned to a tour. This is attempted just after the best move is executed by the algorithm. The algorithmic representation of the vehicle exchange routine is displayed in Algorithm 6. After executing the best move, the algorithm searches the solution for tours which involve only depots. If such a tour is found, the algorithm will make the vehicle assigned to it available and remove the tour from the solution if all of the customers associated with the problem instance have been assigned to a tour. The algorithm attempts to insert any unassigned customers into tours after the execution of each move in case the best move selection routine does not select a move involving an unassigned customer. Each time a new best solution is found, this solution is saved as the incumbent solution. The Tabu search algorithm terminates when a specified number of iterations have elapsed without finding a new best solution.

3.2.3.1 Tabu List

When a particular move is made, the inverse of that move is considered tabu. Such a move will be added to the tabu list, which is the short term memory tabu structure, for the duration of the tabu tenure. This allows the search to avoid returning to a past solution immediately. The frequency of executing each move is tracked by the solution algorithm. A limit, *Frequency limit*, is placed upon the number of times a particular move may be executed. Once this limit is reached by a particular move, that move is considered tabu. Such a move will be added to the frequency list which is the long term memory tabu structure. Unlike the tabu list, the frequency list does not make use of the tabu tenure. The moves added to the frequency list will remain in the list until the search terminates or the tabu memory structures' contents are erased.

3.2.3.2 Tabu Tenure

The tabu tenure is the number of iterations for which a tabu move will be prevented. The solution algorithm makes use of a dynamic tabu tenure. The dynamism of the tabu tenure is driven by the number of iterations which have elapsed since a new best solution was last found. The tabu tenure resembles a step function with an independent variable representing the number of iterations which have elapsed since a new best solution was last found. The value of the tabu tenure is smaller for a certain number of iterations when a new best

solution is found. This mechanism promotes the intensification of the search in a promising neighbourhood. The tabu tenure tends to increase with the number of iterations which have elapsed since a new best solution was last found. This increase in tabu tenure promotes the diversification of the search into uncharted neighbourhoods.

3.2.3.3 Aspiration Criteria

The solution algorithm makes use of an aspiration criterion which will allow tabu moves to be executed if their execution will yield a new best solution. This mechanism is represented in Algorithm 5, line 7.

3.2.3.4 Restarting the Search

The tabu memory structures are cleared when the search reaches a predetermined number of iterations without finding a new best solution. This mechanism is utilised later on during the search of a problem instance's solution space. This mechanism has the same effect as would restarting the Tabu search algorithm at the current solution (as opposed to the initial solution). This mechanism is utilised in an attempt to find a new best solution by allowing moves that would otherwise have been disallowed by the tabu memory structures.

3.3 Verification

3.3.1 Solomon's VRPTW Benchmarking Problems

The solution algorithm has been tested on several of Solomon's VRPTW benchmarking problem instances. The results of these tests appear in Table 3.1 below. For each problem instance, the problem identifier, number of vehicles utilised in the solution (N.V.) and the solution distance are detailed.

3.3.2 Convergence and Repeatability of the Solution Algorithm

The convergence and repeatability of a metaheuristic algorithm are important characteristics which give an indication of the efficacy and consistency of the algorithm. The convergence of an algorithm is best displayed in graphical form. Figure 3.4 displays graphically the convergence of the solution algorithm for one of Solomon's benchmark problem instances (RC102.100). The repeatability of an algorithm is an indicator of how consistently the algorithm performs. A measure of the variation in the performance of an algorithm can be used to give an indication of the consistency of the algorithm. In order to test the repeatability of the solution algorithm, the algorithm was executed twenty times on a particular problem instance (Solomon's RC102.100 benchmark problem). The results of these trials are tabulated in Table 3.2. Table 3.2 displays the result for each trial. The result of each trial is the objective function value (distance

for Solomon's benchmark problems) of the final solution found by the solution algorithm. The mean absolute deviation (MAD) is used as a measure of variation in analysing the repeatability of the algorithm. The MAD is significantly below five percent (of the mean of the performance results in Table 3.2) which indicates that the algorithm displays favourable repeatability.

3.4 Conclusion

This chapter offers a review of the nuances of the problem. The problem review is used as the point of departure in the establishment of the problem's mathematical model and in turn the solution algorithm. Finally, the algorithm is tested on several of Solomon's benchmarking problem instances and its performance is subjected to analyses.

Algorithm 1 Generate Initial Solution

Input: Customer, depot and vehicle data

```
1: Initialise solution
2: while unrouted customers exist and unused vehicles are available do
3:    Initialise tour
4:    Assign a vehicle to tour
5:    Insert assigned vehicle's start depot as tour start location
6:    Insert assigned vehicle's end depot as tour end location
7:    Identify and insert seed customer
8:    while  feasible inserts exist and unrouted customers exist do
9:       Initialise set of potential inserts
10:      for all unrouted customers  do
11:         Identify feasible inserts
12:         Compare feasible inserts and identify best insert
13:         Add best insert to the set of potential inserts
14:      end for
15:      Identify the best insert in the set of potential inserts
16:      Insert the best insert into tour
17:   end while
18: end while
```

Algorithm 2 Tabu Search

Input: *Initial solution*

1: *Incumbent solution* ← *Initial solution*
2: *Current solution* ← *Initial solution*
3: *Counter* ← 0
4: **while** *Counter*<*Stop condition* **do**
5: Identify a set of *Potential moves* for the *Current solution*
6: Identify the best move from the set of *Potential moves*
7: **if** *Potential moves* is ∅ **or** *Best move Tabu count*> 0 **or** *Best move Frequency count*==*Frequency limit* **then**
8: Attempt to execute feasible node exchange in the *Current solution*
9: **else**
10: Execute the best move on the *Current solution*
11: **end if**
12: **if** tours involving only depots exist and all customers have been assigned to a tour **then**
13: Add the vehicles which are assigned to depot-only tours to the set of *Available vehicles*
14: Remove tours involving only depots from the solution
15: **end if**
16: **if** Unassigned customers exist **then**
17: Insert unassigned customers if feasible insert positions exist
18: **end if**
19: Try to exchange vehicles
20: Add the executed move to the *Frequency list*
21: Add the inverse of the executed move to the *Tabu list*
22: Inverse of executed move *Tabu count* ← *Tabu tenure*
23: Executed move *Frequency count* ← executed move *Frequency count* +1
24: Update the *Tabu list*
25: Calculate cost of the *Current solution*
26: **if** *Current solution* cost<*Incumbent solution* cost **then**
27: *Incumbent solution* ← *Current solution*
28: *Counter* ← 0
29: clear *Tabu list*
30: clear *Frequency list*
31: **else**
32: *Counter* ← *Counter*+1
33: **end if**
34: Assign a suitable value to *tabu tenure* based on the value of *Counter*
35: **if** *Counter* has reached *Predetermined value* **then**
36: Clear tabu memory structures
37: **end if**
38: **end while**

Algorithm 3 Identify Feasible Moves

Input: *Current solution*

```
1: Initialise the set Potential moves
2: while Potential moves =∅ & StopCriterion is not satisfied do
3:    {StopCriterion avoids infinite looping}
4:    for arbitrary integer do
5:       a ← randomly selected tour from Current solution
6:       b ← randomly selected tour from Current solution
7:       {a may be the same tour as b}
8:       for each position j in a except the first and last do
9:          for each position k in b except the first and last do
10:            if a ≠ b or j ≠ k then
11:               Consider move p of customer in j to k
12:               if p is feasible then
13:                  Add p to Potential moves
14:               end if
15:            end if
16:         end for
17:      end for
18:   end for
19: end while
20: if unassigned customers exist then
21:    for each customer z in the set of Unassigned customers do
22:       for each tour y in Current solution do
23:          Consider the insert, x, of customer z into tour y
24:          if x is feasible then
25:             Add x to Potential moves
26:          end if
27:       end for
28:    end for
29: end if
30: return Potential moves
```

Algorithm 4 Attempt to Exchange Nodes

Input: *Current solution*

```
1: for five counts do
2:   if feasible exchange has yet to be identified then
3:     i ← random tour in Current solution
4:     j ← random tour in Current solution
5:     k ← random customer in i
6:     l ← random customer in j
7:     if i ≠ j or k ≠ l then
8:       if exchanging customers k and l will not violate any constraints then
9:         {moving k to l's current position and l to k's current position}
10:        exchange customers k and l
11:        feasible exchange found
12:      end if
13:    end if
14:  end if
15: end for
```

Algorithm 5 Identify Best Move

Input: *Current solution*, *Incumbent solution*, *Potential moves*

```
1: Best move ← randomly selected move in Potential moves
2: for all moves in Potential moves do
3:   if move differential cost < Best move differential cost then
4:     if move is not tabu then
5:       Best move ← move
6:     else
7:       if Current solution cost + move differential cost<Incumbent solution
         cost then
8:         Best move ← move
9:       end if
10:    end if
11:  end if
12: end for
13: return Best move
```

Algorithm 6 Exchange Vehicle

Input: *Current solution*

if All available vehicles have been assigned to a tour **then**
  Select random tour, i, in *Current solution*
  Select random tour, j, in *Current solution*
  $a \leftarrow$ vehicle assigned to tour i
  $b \leftarrow$ vehicle assigned to tour j
  $c \leftarrow$ the total demand of tour i
  $d \leftarrow$ the total demand of tour j
  $e \leftarrow$ the start depot of vehicle a
  $f \leftarrow$ the end depot of vehicle a
  $g \leftarrow$ the start depot of vehicle b
  $h \leftarrow$ the end depot of vehicle b
  $k \leftarrow$ the load capacity of vehicle a
  $l \leftarrow$ the load capacity of vehicle b
  if $i \neq j$ and $f == h$ and $e == g$ and $k \geq d$ and $l \geq c$ **then**
    Assign vehicle a to tour j
    Assign vehicle b to tour i
  end if
else
  Randomly select an available vehicle w
  $q \leftarrow$ the start depot of vehicle w
  $r \leftarrow$ the end depot of vehicle w
  $s \leftarrow$ the load capacity of vehicle w
  Randomly select a tour, t, in *Current solution*
  $u \leftarrow$ vehicle assigned to tour t
  $v \leftarrow$ the start depot of vehicle u
  $w \leftarrow$the end depot of vehicle u
  $x \leftarrow$ the total demand of tour t
  if $q == v$ and $r == w$ and $s \geq x$ **then**
    Add u to the set of *Available vehicles*
    Remove w from the set of *Available vehicles*
    Assign vehicle w to tour t
  end if
end if

Problem	N.V.	Distance
R101.100	15	2540.56
R102.100	11	1810.66
R201.100	10	1883.55
R202.100	10	1570.54
C101.100	13	1816.74
C102.100	13	1699.40
C201.100	11	1585.59
C202.100	10	1664.75
RC101.100	11	1974.43
RC102.100	12	1802.66
RC201.100	14	2365.40
RC202.100	11	1894.33

Table 3.1: Solomon's VRPTW benchmarking problems

Trial	Result
1	1786.47
2	1917.854
3	1835.46
4	1712.94
5	1859.47
6	1821
7	1793.54
8	1971.2
9	1852.06
10	1767.95
11	1818.72
12	1805.69
13	1816.98
14	1777.47
15	1817.39
16	1959.29
17	1970.98
18	1822.25
19	1821.41
20	1820.44
MAD	MAD (%)
50.63	2.76%

Table 3.2: Algorithm repeatability

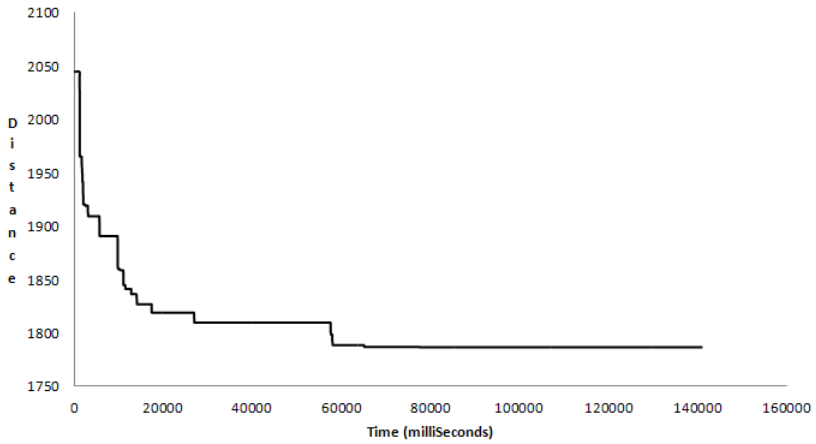


Figure 3.1: Convergence of the solution algorithm

Chapter 4

Implementation

4.1 Introduction

This chapter presents a case study problem which is an epitome of the type of problem instance to which the solution algorithm is intended to be applied. The solution yielded by the solution algorithm is also presented in this chapter. The solution algorithm has been programmed in the object-oriented programming language *Java*.
Metaheuristic optimisation algorithms which are intended to be applied to a number of problem instances must possess the flexibility required to accommodate the data associated with different problem instances. Another important capability of such algorithms is the ability to perform well over the spectrum of problem instances to which they will be applied. Algorithms which perform well over a variety of problem instances are said to be *robust*. The performance of a metaheuristic algorithm is often dependent upon the values of its parameters. As a result, in developing a robust optimisation algorithm, parameter tuning is a crucial exercise.

4.2 Parameter Tuning

Statistical methods are applied to tune the parameters of optimisation algorithms. Design of experiments (DOE) is one statistical method which can be utilised to find appropriate values for the parameters of optimisation algorithms. According to Ridge [2007] DOE is an efficient statistical method from which substantiated conclusions can be drawn. Alternative approaches which may be used to identify appropriate parameter values include the trial-and-error approach and one-factor-at-a-time (OFAT) analysis. DOE tends to demonstrate a greater appreciation for the interaction between factors which influence the performance of an optimisation algorithm. Thorough parameter tuning is a time-consuming exercise. Automated parameter-tuning software has been developed to facilitate this process and decrease the time required for parameter

tuning. An example of such software is CALIBRA, which is freely available: http://opalo.etsiig.uniovi.es/~adenso/file_d.html.

4.2.1 Design of Experiments

Montgomery and Runger [2007] note that the validity of the conclusions which are drawn from an experiment depends heavily upon the way in which the experiment is conducted. The design of an experiment therefore serves a critical purpose in the conducting of a valid experiment. Factorial experiments, which are utilised for experiments involving two or more factors, consist of trials involving all of the combinations of the levels of the factors being investigated. The factors which an experiment investigates are those parameters which the experimenter suspects have an impact upon some response variable of interest. The levels of a factor are the values which the factor can assume in the experiment. The more factors involved in an experiment, the larger the number of treatments undertaken in the experiment. To overcome this obstacle, several approaches make use of a screening step initially to identify those factors which have a considerable effect upon response variables of interest. Screening experiments which involve several factors often make use of fractional factorial experiments which only consider a fraction of the combinations of levels of the various factors. Fractional factorial experiments however lose some of the information which a full factorial experiment provides. The effects of some factors or some combinations of factors may be aliased in fractional factorial experiments. Aliased effects are those effects on the response variable of interest which cannot be attributed to a single factor or a single combination of factors. This may lead to the experimenter misinterpreting the experiment's results or not being able to isolate the effect of a particular factor due to the alias. This situation can however be remedied by conducting one or more additional small experiments which then provide more information which effectively removes the uncertainty pertaining to the aliased effects of interest. In this way series of compact experiments can effectively be used to determine the effects of parameters upon a response variable of interest. Those factors which are identified as considerable in the screening step are then subjected to thorough experimentation. Typically, screening experiments involve fewer levels (generally two or three) of each factor. The subsequent thorough experimentation will involve more levels of those factors which progress beyond the screening experiment.

4.2.2 Screening Experiment

Trial-and-error experimentation during the course of the development of the solution algorithm accommodated insightful observations of the effect of various factors upon the quality of the ultimate solution. The cost of the ultimate solution yielded by the solution algorithm is the response variable of interest. One factor which has been observed to have a notable effect upon the quality (or cost) of the ultimate solution is the quality of the initial solution presented to the metaheuristic algorithm. It was also noted that any increase in the value of

the termination criterion beyond a certain threshold value does not seem to have a considerable effect upon the response variable. The termination criterion is responsible for terminating the solution algorithm's search of the solution space. The following factors have been considered in the parameter tuning activities:

- Initial solution quality or cost.
- Search termination criterion.
- Tabu tenure (short-term tabu memory structure).
- Frequency limit (long-term tabu memory structure).

The results and normal probability plot of an unreplicated 2^k factorial experiment for a particular problem instance are displayed in Figure 4.1. The factor labels correspond to factors as follows: A represents initial solution cost (or quality), B represents tabu tenure, C represents frequency limit and D represents termination criterion. An unreplicated 2^k factorial experiment involves a single replication of all combinations of factor levels. For each factor, two levels are considered, a low level and a high level. The normal probability plot facilitates the identification of factors with considerable effects upon the response variable of interest. Such factor effects will be conspicuously displaced from the best-fit regression line. The response variable of interest in this experiment is the cost or quality of the ultimate solution produced by the solution algorithm. Those factors and factor combinations which have a considerable effect upon the response variable have been labelled on the normal probability plot. These factors are: initial solution cost (A), tabu tenure (B) and the termination criterion (D). The interaction between factors also affect the response variable considerably in some cases. The interaction between the factors in the following factor combinations can be identified upon the normal probability plot: CD, AB and BD. With some of the isolated factors identified in the screening experiment, more thorough experimentation involving more levels has been undertaken.

4.3 Case Study Problem

Solomon's VRPTW benchmarking problem instances are not accompanied by a road network. Therefore, all distances are based on Euclidean distance. The following case study problem requires inter-customer distances to be calculated using a shortest path algorithm which searches the relevant road network. Once the road network search has been executed, the routing algorithm issues as output the internode distance and travel-time matrices. The case study problem considered consists of distributing products to twenty customers. Table 4.1 displays the customer data pertaining to this case study problem. The delivery vehicle fleet consists of five vehicles. The vehicle data is tabulated in Table 4.2. For demonstrative purposes, this case study problem involves two time slices. Each time slice has associated internode distance and travel-time matrices. The necessity of including a unique set of internode distance and travel-time matrices

for each time slice stems from the effect of traffic upon the choice of path and travel time between two nodes. The first time slice spans half of the operational hours of the depot (node 1) and time slice two spans the remaining half of the depot's operational hours. The internode distance matrices for time slice one and two are displayed in Figure 4.2 and Figure 4.4 respectively. The internode travel-time matrices for time slice one and time slice two are displayed in Figure 4.3 and Figure 4.5 respectively. Time is measured in minutes from the opening time of the depot (node 1). This time is used as the datum and is set to zero. Therefore all times have been expressed as the number of minutes which have elapsed since the opening time of the depot. The results, depicted in Table 4.3, show that only two of the five available vehicles are utilised. The sequence of visitation is conveyed by this depiction. The total distance of the solution is 254.89 km. Solutions of a shorter distance have been encountered by the solution algorithm. It must however be noted that the objective function of the solution algorithm, presented in (3.1) in chapter 3, strives to minimise a combination of several cost drivers, including: the number of vehicles used, solution travel-time and solution distance. This case study problem involves a single depot. Each customer in the case study problem has a single time window of delivery opportunity and a heterogeneous vehicle fleet is utilised to service the set of customers. Geographical displays of the customer locations and the solution yielded by the solution algorithm appear in Figures 4.6, 4.7 and 4.8. Take note that two nodes, nodes 4 and 12, are hidden from view in these displays due to their close proximity with other nodes. This statement can be confirmed upon inspection of the coordinates of the nodes in Table 4.1. The case study problem's constraints have a significant impact upon the structure of the solution. In particular, the sequence of visitation is to a large extent at the mercy of the customers' time windows. In the event that the time windows of the customers in the case study problem are altered, it is highly probable that a radically different solution will be presented by the solution algorithm.

4.4 Conclusion

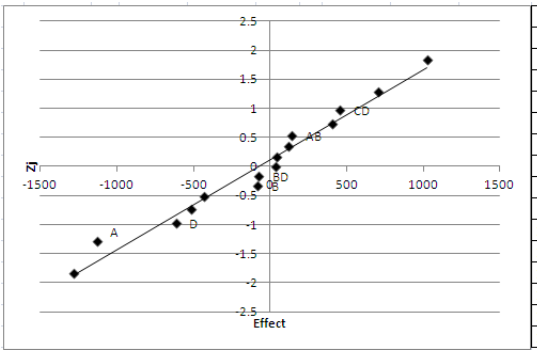
This chapter introduces a case study problem which is an epitome of the type of problem instance to which the solution algorithm will be applied. The algorithm's results for this case study problem are presented in this chapter. A discussion on parameter tuning, in which its importance in the development of robust optimisation algorithms is emphasized, is also presented in this chapter. An example of the parameter tuning activities which have been undertaken is presented and discussed in this chapter.

ID	Demand	Early	Late	ServiceTime	x-coordinate	y-coordinate
1	0	0	780	0	28.22397	-26.18496
2	15	117	716	21	28.103707	-26.00005
3	9	159	628	17	28.126228	-26.008651
4	10	113	663	58	28.117228	-25.977431
5	10	106	622	40	28.11697	-25.976711
6	17	193	675	15	28.176941	-26.003722
7	9	216	670	60	28.229415	-25.9757
8	9	76	634	46	28.161337	-25.8853
9	9	135	606	25	28.151463	-25.869916
10	18	13	614	21	28.167945	-25.855158
11	5	95	692	36	28.148665	-25.747338
12	15	135	679	48	28.188967	-25.862011
13	7	132	610	57	28.189432	-25.858073
14	18	135	630	32	28.204514	-25.88521
15	8	74	646	52	28.259813	-25.826475
16	5	187	685	54	28.234451	-25.770004
17	14	232	679	60	28.187231	-25.740372
18	9	4	651	55	28.242347	-25.700749
19	15	174	610	24	28.192748	-25.683407
20	17	14	686	58	28.24114	-25.677787
21	14	116	658	45	28.311646	-25.735168

Table 4.1: Case study problem customer data

Vehicle ID	Load Capacity	Fuel Type	Fuel Tank Capacity
1	80	P	50
2	100	D	40
3	50	P	30
4	40	D	60
5	150	P	100

Table 4.2: Case study problem vehicle data



j	(j-0.5)/n	zj	Effect	Factor
1	0.033333	-1.83391	-1282.57	C
2	0.1	-1.28155	-1128.15	A
3	0.166667	-0.96742	-613.459	D
4	0.233333	-0.72791	-515.713	ABC
5	0.3	-0.5244	-433.785	ABCD
6	0.366667	-0.34069	-80.663	B
7	0.433333	-0.16789	-73.931	BD
8	0.5	-1.4E-16	38.203	ACD
9	0.566667	0.167894	45.683	AD
10	0.633333	0.340695	119.361	BCD
11	0.7	0.524401	142.835	AB
12	0.766667	0.727913	406.901	BC
13	0.833333	0.967422	455.917	CD
14	0.9	1.281552	704.363	ABD
15	0.966667	1.833915	1023.275	AC

Figure 4.1: Screening experiment results and normal probability plot

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	27.28	26.07	30.31	30.57	24.29	31.94	37.59	40.29	41.39	59.92	45.54	46.01	41.43	46.96	54.67	57.29	62.85	63.7	67.51	58.79
2	27.32	0	3.61	3.04	3.3	10.45	18.74	14.94	17.64	18.75	34.63	19.52	20.65	19.95	30.88	32.71	32.92	40.5	39.98	46.4	39.72
3	26.07	3.66	0	4.69	4.95	7.12	17.37	14.44	17.14	18.24	34.84	19.98	20.14	19.44	31.08	32.92	33.13	40.71	40.18	46.6	39.93
4	30.8	3.04	4.69	0	0.26	9.56	15.7	11.91	14.61	15.71	31.59	16.48	17.61	16.91	27.84	29.68	29.88	37.47	36.94	43.36	36.68
5	31.06	3.3	4.95	0.26	0	9.82	15.94	12.16	14.86	15.97	31.85	16.73	17.86	17.17	28.09	29.93	30.14	37.72	37.19	43.62	36.94
6	24.28	10.48	7.15	9.59	9.85	0	10.3	17.67	20.37	21.48	37.46	23.22	23.38	22.68	29.18	35.54	35.75	43.33	42.8	49.23	39.64
7	32.6	18.67	17.3	15.63	15.89	10.26	0	16.55	19.25	20.36	35.36	18.91	19.38	14.8	20.44	30.11	32.73	38.29	39.13	42.94	34.23
8	37.58	15.34	14.44	12.3	12.56	17.62	16.54	0	2.7	3.8	18.41	5.55	5.7	6.59	17.9	18.99	18.04	26.78	25.1	31.52	26.75
9	40.29	18.05	17.15	15.01	15.27	20.33	19.25	2.71	0	2.89	17.5	4.64	4.79	7.81	16.99	18.08	17.13	25.87	24.19	30.61	25.84
10	41.39	19.14	18.24	16.1	16.36	21.43	20.34	3.8	2.9	0	15.81	3.49	3.65	6.67	15.85	14.75	15.45	22.54	22.5	28.92	24.69
11	60.34	35.33	35.21	32.3	32.55	37.98	36	18.35	17.45	15.75	0	18.14	17.92	21.98	17.03	10.08	4.64	13.91	11.33	17.75	17.5
12	46.13	19.59	19.47	16.55	16.81	23.12	18.87	5.5	4.59	3.46	17.73	0	0.58	3.57	12.75	16.03	16.02	25.14	23.08	30.31	21.59
13	46.6	20.06	19.94	17.02	17.28	23.58	19.34	5.95	5.05	3.92	17.15	0.59	0	4.03	13.22	15.45	15.43	25.61	22.49	30.78	22.06
14	42.09	20.27	19.38	17.24	17.49	22.56	14.83	6.61	7.75	6.62	21.61	3.57	4.03	0	12.88	19.74	19.9	26.77	26.95	31.95	23.23
15	46.46	30.48	30.35	27.44	27.69	29.19	20.46	18.28	17.37	16.24	16.77	13.19	13.66	12.87	0	9.44	14.14	16.95	20.23	22.12	13.4
16	54.8	32.79	32.67	29.76	30.01	35.44	30.46	19.15	18.24	14.91	10.28	15.6	15.38	19.63	9.61	0	7.65	8.26	12.75	16.86	10.82
17	57.04	32.98	32.86	29.94	30.2	35.63	32.7	17.98	17.07	15.38	4.61	15.79	15.56	19.63	13.73	6.78	0	9.28	7.1	13.52	13.31
18	62.77	40.61	40.49	37.57	37.83	43.26	38.42	27.1	26.2	24.51	13.74	25.93	23.19	26.72	17.02	8.2	9.13	0	7.25	11.59	10.1
19	63.91	39.88	39.76	36.84	37.1	42.53	39.57	24.88	23.97	22.28	11.25	22.68	22.46	26.53	20.6	12.63	6.9	7.02	0	6.84	17.98
20	66.64	45.61	45.49	42.58	42.83	48.26	42.3	30.61	29.71	28.01	16.98	30.34	28.2	31.13	21.43	16.65	12.63	11.26	5.74	0	13.21
21	58.45	39.42	39.3	36.39	36.64	39.35	34.1	27.23	26.32	25.19	17.56	22.14	22.61	22.93	13.23	11.12	14.16	10.19	18.05	13.93	0

Figure 4.2: Time slice one internode distance matrix

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	20.46	20.04	22.74	22.93	18.42	21.09	28.19	31.18	31.5	38.19	31.01	31.71	29.86	30.63	34.15	36.08	41.7	41.04	44.33	38.79
2	20.49	0	3.18	2.28	2.47	10.59	14.64	11.12	14.11	14.42	22.84	15.62	15.89	15.52	22.5	22.25	21.42	29.51	26.87	32.12	28.54
3	20.04	3.21	0	4.03	4.22	8.39	14.13	11.34	14.33	14.64	23.54	15.88	16.11	15.74	23.2	22.95	22.12	30.2	27.57	32.82	29.24
4	22.56	2.28	4.03	0	0.19	9.03	12.36	8.84	11.83	12.14	20.56	13.34	13.62	13.24	20.22	19.97	19.14	27.23	24.59	29.84	26.26
5	22.76	2.47	4.22	0.19	0	9.23	12.54	9.03	12.02	12.34	20.75	13.53	13.81	13.44	20.41	20.17	19.34	27.42	24.79	30.03	26.46
6	18.41	10.64	8.41	9.06	9.25	0	8.52	13.46	16.45	16.76	26.26	18	18.23	17.86	22.58	25.67	24.84	32.93	30.29	35.54	31.2
7	22.41	14.57	14.06	12.29	12.49	8.47	0	12.99	15.98	16.29	24.72	15.05	15.75	13.91	16.41	20.69	22.61	28.23	27.58	30.86	25.32
8	28.19	11.27	11.34	9	9.19	13.42	12.99	0	2.99	3.3	14.66	4.55	4.78	5.59	13.21	14.47	13.53	21.73	18.98	24.23	19.26
9	31.2	14.28	14.35	12	12.2	16.42	16	3.01	0	2.78	14.14	4.02	4.25	6.67	12.69	13.95	13.01	21.2	18.46	23.71	18.73
10	31.49	14.58	14.65	12.3	12.49	16.72	16.3	3.3	2.78	0	13.17	3.63	3.86	6.27	12.3	12.9	12.04	20.16	17.49	22.73	18.34
11	38.64	23.35	23.72	21.08	21.27	27.11	25.21	13.9	13.38	12.4	0	13.63	13.3	16.42	13.2	8.01	3.61	12.01	8.79	14.04	13.02
12	32.19	15.14	15.51	12.86	13.05	17.79	14.94	4.38	3.86	3.48	13.3	0	0.87	3.54	9.56	12.68	11.89	18.86	17.33	21.14	15.61
13	32.89	15.84	16.21	13.56	13.75	18.48	15.65	5.06	4.55	4.17	13.11	0.88	0	4.24	10.27	12.49	11.7	19.56	17.14	21.84	16.31
14	31.18	15.62	15.68	13.34	13.53	17.76	13.93	5.6	6.61	6.23	16.23	3.6	4.3	0	10.79	14.61	14.81	20.66	20.26	22.95	17.41
15	30.31	21.48	21.85	19.21	19.4	22.59	16.41	13.24	12.72	12.34	13	9.71	10.41	10.8	0	7.12	10.89	13.88	15.54	16.17	10.63
16	34.39	22.43	22.8	20.15	20.34	26.18	20.96	14.63	14.11	13.06	7.88	12.71	12.37	14.72	7.3	0	5.78	7.69	9.96	13.93	7.91
17	36.03	21.46	21.83	19.18	19.37	25.21	22.6	13.48	12.96	11.99	3.59	11.74	11.4	14.53	10.59	5.4	0	8.41	5.48	10.73	10.18
18	41.77	29.7	30.07	27.42	27.62	33.46	28.34	21.78	21.26	20.28	11.89	19.27	19.65	20.64	13.86	7.64	8.3	0	8.98	11.75	9.55
19	41.18	26.63	27	24.35	24.55	30.39	27.75	18.66	18.14	17.16	8.57	16.91	16.58	19.7	15.74	10.15	5.17	8.65	0	5.88	14.22
20	42.99	30.99	31.36	28.71	28.9	34.74	29.56	23.01	22.49	21.52	12.92	20.41	20.93	21.77	14.99	13.08	9.53	10.77	4.36	0	11.33
21	38.32	27.52	27.89	25.24	25.43	30.91	24.89	19.27	18.75	18.38	12.94	15.74	16.44	17.1	10.32	8.02	10.26	9.6	13.87	12.49	0

Figure 4.3: Time slice one internode travel-time matrix

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	32.05	33.65	24.37	51.77	38.03	48.34	42.9	51.48	65.65	87.74	40.98	53.02	73.82	75.56	48.78	54.07	68.38	59.18	83.31	85.04
2	41.67	0	5.786	3.002	4.374	11.4	19.68	19.06	21.86	33.02	44.54	29.33	27.32	30.73	31.46	35.64	37.93	35.38	57.94	55.58	66.57
3	27.49	5.814	0	4.471	6.632	12.43	24.1	23.06	26.41	19.85	46.72	24.96	27.51	20.95	32.42	51.06	29.62	37.36	56.14	67.15	57.99
4	48.16	2.881	4.67	0	0.315	15.07	16.36	17.23	17.08	20.88	45.71	14.42	17.49	29.33	44.12	48.04	50.97	44.15	34.06	67.25	48.93
5	37.37	2.919	5.624	0.288	0	16.84	27.08	18.13	25.02	14.79	48.46	22.42	22.19	26.35	31.63	52.31	27.16	57.77	50.15	67.21	34.16
6	21.36	10.59	10.32	10.96	9.329	0	9.441	14.85	18.17	26.01	36.97	32.34	41.26	22.79	40.29	56.31	30.92	68.83	34.99	47.61	44.56
7	54.77	23.42	28.69	27.28	28.19	10.82	0	15.43	29.22	16.72	56.13	23.45	17.14	17.06	23.87	50.71	40.13	62.34	57.11	36.38	56.83
8	67.4	15.42	22.92	13.26	18.12	16.11	24.48	0	2.646	4.742	24.51	5.932	6.173	8.361	16.01	27.7	29.52	22.92	35.2	35.58	44.59
9	34.06	28.4	18.08	19.07	23.38	24.11	17.63	4.606	0	2.869	22.95	3.956	7.424	9.951	23.01	16.41	21.35	38.64	42.3	53.18	23.96
10	61.53	29.44	24.09	14.84	29.05	19.4	31.03	3.22	4.994	0	13.47	5.623	5.709	10.02	16.09	23.09	22.63	31.25	24.42	26.3	43.1
11	95.74	52.82	43.26	52.28	30.36	46.3	38.41	23.78	16.31	14.93	0	26.12	30.53	23.14	29.47	10.06	5.877	20.87	10.57	26.97	27.39
12	68.35	28.49	16.62	13.58	14.99	22.75	16.11	6.847	6.21	3.726	27.49	0	0.796	3.629	22.41	15.42	23.08	44.8	20.77	52.4	24.66
13	81.26	27.15	23.23	26.08	22.13	28.14	20.17	8.383	5.198	6.906	22.67	0.984	0	6.987	13.74	15.96	19.25	30.69	38.6	41.09	28.39
14	54.9	27.31	24.93	22.34	30.84	20.83	13.24	10.51	10.77	5.989	35.44	6.308	5.675	0	14.7	23.85	22.4	39.05	24.44	45.03	21.09
15	66.12	50.44	51.46	28.68	37.17	38.15	26.32	21.73	20.61	13.93	18.28	12.76	20.41	21.96	0	15.48	21.55	18.09	16.29	37.73	21.7
16	49.5	42.34	45.91	49.21	47.78	34.1	43.76	16.22	28.86	19.23	14.46	14.36	18.24	32.72	8.617	0	7.953	10.34	18.27	20.49	11.06
17	84.44	50.36	38.65	39.35	42.1	50.28	49.08	15.66	21.32	19.47	4.916	16.79	21.16	27.47	21.5	5.856	0	10.98	7.334	18.51	17.59
18	106.3	53.86	71.62	30.48	63.88	42.74	30.94	23.55	34.23	29.5	18.22	34.01	23.77	47.2	27.03	7.347	9.692	0	11.39	12.16	10.65
19	88.45	49.07	37.47	37.78	65.22	60.05	40.27	39.64	40.04	25.05	17.46	23.33	34.73	26.85	27.32	15.44	6.081	10.57	0	8.956	22.43
20	113.7	77.27	39.95	73.42	44.38	50.05	54.19	25.92	35.39	29.88	21.86	48.35	27.77	46.55	35.47	17.19	20.96	9.72	7.919	0	21.78
21	62.73	62.59	48.2	40	59.96	33.48	37.83	45.06	31.6	23	21.1	32.73	18.83	31.78	19.19	9.813	21.26	12.21	17.69	22.44	0

Figure 4.4: Time slice two internode distance matrix

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	34.4	23.14	24.07	29.5	24.79	28.63	50.41	27.76	31.08	36.16	38.57	26.32	35.16	38.54	53.4	36.67	39.33	72.34	40.54	31.51
2	24.11	0	4.964	3.476	2.995	12.62	25.91	11.24	17.2	23.83	25.36	27.11	27.67	17.75	37.54	19.75	37.9	44.05	37.21	36.71	30.76
3	32.24	4.599	0	6.529	5.327	7.462	17.25	19.08	25.66	12.29	25.77	23.61	21.29	26.91	35.19	25.16	30.06	38.19	31.87	44.15	43.05
4	33.74	2.371	4.821	0	0.153	8.991	13.41	15.11	12.9	10.41	31.62	15.15	20.26	21.95	32.98	35.37	28.7	22.29	31.25	40.27	21.86
5	39.38	3.626	6.965	0.306	0	13.71	15.63	9.666	9.862	17.74	24.53	13.81	13.51	12.77	20.34	21.86	15.76	23.26	23.69	52.32	39.56
6	23.3	9.676	10.49	7.729	16.13	0	14.8	23.66	14.92	22.95	24.61	17.43	23.67	26.26	25.34	24.02	38.88	36.77	25.2	29.62	28.27
7	38.78	25.15	15.91	14.67	21.99	14.84	0	18.65	25.58	24.23	36.34	13.9	23.02	14.42	14.91	35.8	31.32	35.55	41.92	53.97	27.43
8	50.31	19.71	14.52	7.753	14.71	23.81	18.05	0	2.423	3.776	16.22	7.745	7.84	9.133	23.02	23.46	12.42	30.95	25.39	27.33	28.06
9	43.69	24.38	14.71	19.59	18.35	19.61	26.95	3.467	0	4.745	21.92	4.498	6.707	5.618	10.5	11.49	19.28	32.23	28.35	42.49	31.81
10	49.15	24.46	20.13	14.81	20.34	27.31	20.47	3.954	3.635	0	11.54	3.34	4.433	9.932	16.85	19	14.58	17.44	25.67	28.3	25.44
11	39.94	27.65	35.06	24.12	35.7	30.83	32.41	11.49	22.18	20.29	0	23.94	21.47	28.68	17.75	10.9	6.413	21.1	12.66	19.75	14.84
12	27.83	20.57	13.34	22.36	22.64	26.4	16.44	3.851	6.797	4.26	17.15	0	1.04	3.578	15	19.77	14.39	26.72	24.74	20.15	13.09
13	29.67	16.68	24.14	20.38	23.62	27.38	20.82	5.725	3.739	6.696	17.52	1.487	0	4.468	16.15	11.69	17.59	24.3	19.11	24.44	19.66
14	44.77	12.75	21.9	20.81	16.53	31.62	11.97	7.35	11.68	10.42	15.35	3.946	7.246	0	12.72	25.48	20.1	21.61	29.12	40.5	27.93
15	35.57	24.77	27.94	17.22	17.03	18.55	16.34	10.72	15	15.45	11.09	15.74	17.7	12.49	0	12.6	10.43	19.16	27.55	14.81	10.2
16	51.33	25.35	39.24	35.7	21	21.67	33.13	13.93	23.96	15.88	10.84	22.03	18.29	19.54	13.05	0	5.227	12.93	11.29	12.6	11.82
17	47.76	20.01	33.1	30.68	19.5	30.7	18.96	13.88	14.43	10.91	3.377	9.711	11.05	15.99	16.78	7.569	0	14.43	9.332	14.67	12.03
18	70.69	32.5	45.15	23.4	46.35	56.15	40.79	19.94	29.51	19.12	15.64	29.96	31.7	29.18	13.26	10.29	13.5	0	7.726	17.68	13.67
19	47.77	29.58	41.16	38.16	25.43	42.41	44.19	33.32	21.53	27.29	8.949	26.23	14.76	20.91	14.12	15.74	9.225	14.1	0	8.686	25.38
20	76.11	27.97	51.15	38.17	47.3	38.17	30.05	39.13	23.92	25.13	11.14	22.65	19.63	30.31	15.63	17.93	8.095	9.601	5.516	0	17.85
21	54.32	47.61	45.54	38.15	44.05	39.82	22.48	26.02	30.59	24.56	16.99	16.22	27.77	28.99	15.48	10.34	12.01	15.94	23.62	19.42	0

Figure 4.5: Time slice two internode travel-time matrix

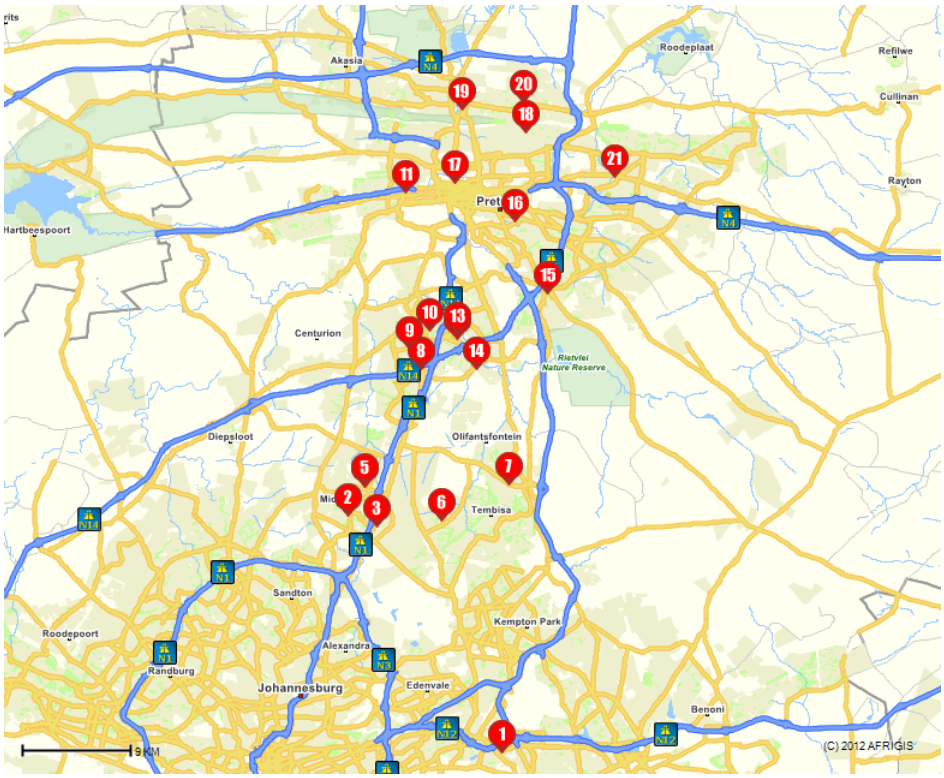


Figure 4.6: Customer geographical locations

Tour ID		Tour ID	
1		2	
Vehicle ID		Vehicle ID	
2		5	
Tour Distance		Tour Distance	
94.76 km		160.12 km	
Node	ETA	Node	ETA
1	0	1	0
15	74	21	116
13	136.41	20	173.49
8	198.47	19	235.85
7	257.46	18	268.5
5	329.95	16	331.14
4	370.14	11	393.02
2	430.51	17	435.43
3	456.48	10	506.34
6	480.94	9	530.98
1	519.24	14	561.6
		12	597.54
		1	673.38

Table 4.3: Case study problem results

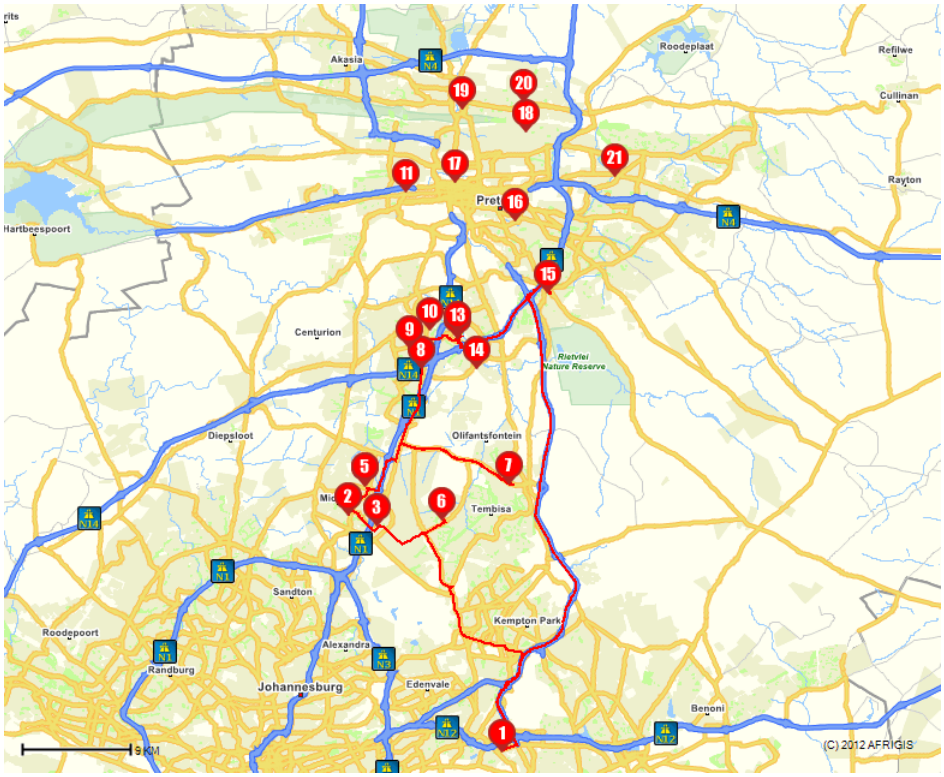


Figure 4.7: Tour one

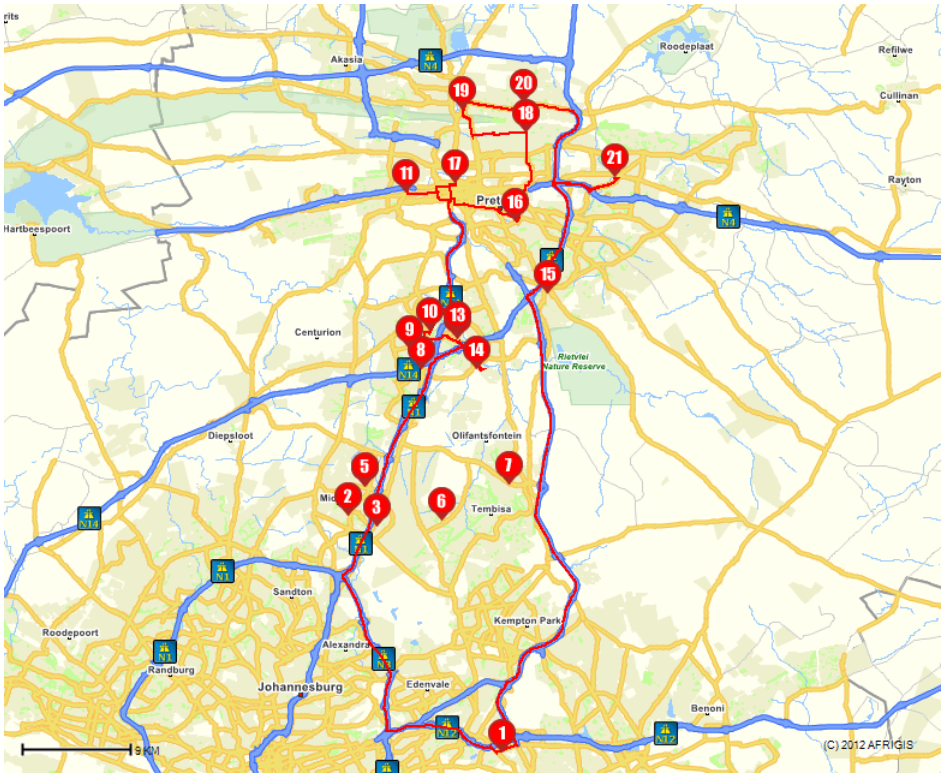


Figure 4.8: Tour two

Chapter 5

Reflection

This project has presented several obstacles during its span. Some notable obstacles include: a one-and-a-half month delay in its initiation due to delayed authorisation by project stakeholders; learning the programming language, Java, without assistance and with very limited programming experience. As a result, the climax of this project is its conclusion. It must however be acknowledged that this project, along with all its obstacles, has provided an incredibly fruitful learning experience. If it were not for the difficulties encountered during the span of this project, the learning experience offered by the project would likely have been significantly less fruitful. The developed solution algorithm is capable of achieving the targets which were initially set for it. It is capable of solving a considerable array of problem instances which may involve one or several depots, a heterogeneous vehicle fleet or a homogeneous vehicle fleet and instances with differing magnitudes, details and levels of restrictiveness. With optimisation algorithms, metaheuristic algorithms in particular, there is always an opportunity for improvement. A considerable amount of experimentation and parameter tuning has already been undertaken, however, as with a high-performance vehicle, the desire for yet further improved performance may set the wheels in motion once more.

References

Toth, P. and Vigo, D. eds., 2002. *The vehicle routing problem.* Philadelphia: Society for Industrial and Applied Mathematics.

Goel, A. and Gruhn, V., 2005. Solving a dynamic real-life vehicle routing problem. In: Haasis, H., Kopfer, H. and Schønberger, J. eds., 2005. *Operations Research Proceedings 2005*, Bremen: Springer, pp.367-372.

Gendreau, M., Potvin, J., Bräumlaysy, O., Hasle, G. and Løkketangen, A., 2007. Metaheuristics for the vehicle routing problem and its extensions: a categorised bibliography. In: Golden, B., Raghavan, S. and Wasil, E. eds., 2008. *The vehicle routing problem: latest advances and new challenges*, Operations Research/Computer Science Interfaces, Vol. 43, pp.143-169.

Rizzoli, A., Montemanni, R., Lucibello, E. and Gambardella, L., 2007. Ant colony optimisation for real-world vehicle routing problems, *Computer Science Collection*, pp.135-151, online Springerlink.

Joubert, J.W., 2003. *An initial solution heuristic for the vehicle routing and scheduling problem.* MEng thesis. University of Pretoria, Pretoria.

Joubert, J.W., 2006. *An integrated and intelligent metaheuristic for constrained vehicle routing.* PhD thesis. University of Pretoria, Pretoria.

Winston, W.L. and Venkataramanan, M., 2003. *Introduction to mathematical programming*, Operations Research: Volume One. Fourth Edition. Belmont: Brooks/Cole.

Taş, D., Daellert, N., van Woensel, T. and de Kok, A.G., 2012. Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, 40 (1), pp.214-224.

Xu, J., Chiu, S.Y. and Glover, F., 1998. Fine-tuning a tabu search algorithm with statistical tests. *Operations Research*, 5 (3), pp.233-244, online Elsevier.

Ridge, E., 2007. *Design of experiments for the tuning of optimisation algorithms.* PhD thesis. University of York, York.

Montgomery, D.C. and Runger, G.C., 2007. *Applied statistics and probability for engineers.* 4th ed. John Wiley and Sons Inc.

Glossary

Alias An aliased effect is an effect on the response variable of interest which cannot be attributed to a unique factor or a unique factor combination. 40

Effect The mean change in the response variable of interest brought about by the level of a factor in concern. 40

Factor The factors which an experiment investigates are those parameters which the experimenter suspects have an impact upon some response variable of interest. 40

Level The levels of a factor are the values which the factor can assume in the experiment. 40

Tour is a set of customers served by a single vehicle. Each tour starts and ends at a depot. 17

Treatment A treatment is a specific level of a factor of interest or a combination of factors, each of which assume a specific level. 40

Acronyms

CVRP capacitated vehicle routing problem. 14, 15

DCVRP capacitated and distance-constrained vehicle routing problem. 14

DOE design of experiments. 39

GRASP greedy randomised adaptive search procedure. 18

MAD mean absolute deviation. 32

MDVRP multiple depot vehicle routing problem. 14, 16

MDVRPTW multiple depot vehicle routing problem with time windows. 22

NP non-determinant polynomial-time. 1, 7, 17

OFAT one-factor-at-a-time. 39

SA simulated annealing. 19

TSP travelling salesman problem. 6

VNS variable neighbourhood search. 18

VRP vehicle routing problem. 1, 6, 7, 17, 19

VRPB vehicle routing problem with backhauls. 14, 15

VRPBTW vehicle routing problem with backhauls and time windows. 14

VRPPD vehicle routing problem with pickup and delivery. 14, 16

VRPPDTW vehicle routing problem with pickup, delivery and time windows. 14

VRPTW vehicle routing problem with time windows. 14, 15, 17, 21, 31, 41