Charging Constrained Electric Vehicle Routing Problem with Prioritized Customers

By

Advaith Krishnan

In collaboration with

Miralis Data

Thesis for the Degree of Master of Science by Research

30 August 2023

This project was supported by the Centre for Global Eco-Innovation and is part financed by the European Regional Development Fund.

Charging Constrained Electric Vehicle Routing Problem with Prioritized Customers

Advaith Krishnan, Masters by Research.
School of Management, Lancaster University

A thesis submitted for the degree of *Masters by Research.* August, 2023

Abstract

This thesis proposes a unique Vehicle Routing Problem (VRP) focusing on charging management optimisations for electric vehicles with prioritised customers. This problem is modelled as a hybrid of the Travelling Repairman Problem (TRP) and the Electric Vehicle Routing Problem (EVRP). After a brief literature review around the scope of the problem, a base mathematical model is formulated to explain the constraints and objective function of the problem. The problem is solved using a Nearest Neighbour Based Heuristic (NNBH) and Simulated Annealing with Variable Neighbourhood Search. The Nearest Neighbour Based Heuristic (NNBH) generates an initial solution. The initial solution is used by the metaheuristic for achieving a better final solution. The base mathematical model is used to benchmark the performance of the solution approach. The algorithmic framework developed is run for smaller and larger instances to demonstrate the accuracy and scalability of the model produced, respectively. The computational results of both instances show the success of the proposed model.

Contents

List of Tables

List of Figures

Acknowledgements

I want to thank the following people: Professor Guglielmo Lulli and Senior Lecturer Burak Boyaci from the Management School of Lancaster University; Will Maden, Chief Operating Officer (COO) of Miralis Data; the Centre of Global Eco-Innovation and the European Regional Development Fund; the CGE admin team for helping throughout this project period; and Ian Wright, Business Partnerships Officer of Lancaster University.

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. This thesis does not exceed the maximum permitted word length of 35,000 words including appendices and footnotes, but excluding the bibliography. The word count is: 13,250

Advaith Krishnan

Chapter 1

Introduction

Sustainable strategic interest in logistics has proliferated in recent years. This is mainly because logistics is considered one of the few remaining areas where significant savings are possible. In 2019, the Transport sector contributed £13.6 billion to the UK economy and covered 18.6 billion kilometres in distance (Dadds, 2021). However, Transport is the UK's largest carbon emitting sector, accounting for more than a quarter of emissions (Waite, 2021). To counteract these carbon emissions that are produced, the UK has implemented a decarbonizing strategy for the transport industry using the Net Zero programme (Skidmore, 2021). One of the many policies outlined in the ”Net Zero Strategy” is the prohibition on the sale of non-zero emission Heavy Goods Vehicles (HGVs) by 2035. Hence, the attention of the logistics industry has shifted from traditional petroleum and diesel-fuelled vehicles to Electric Vehicles (EVs).

But two issues are hindering the complete electric transformation of the logistic fleets. The first issue is that most EVs have a significantly shorter range than traditionally fuel-powered vehicles. Therefore, logistics companies are moving cautiously to electrify their fleets by swapping out the less-frequently used vehicles for electric vehicles. However, the second issue is more concerned with the charging systems of fleets. The economic incentive to cut costs drives companies and researchers. The charging infrastructure that the fleets use for recharging the EVs has a budget for how much electricity it can use for recharging the fleet (Keskin and Çatay, 2016). This means not all vehicles can be charged to 100%. Because of these issues, the prioritisation of which vehicles should be assigned to daily tasks becomes very complex. Along with this complexity, other variables such as distances needed to get

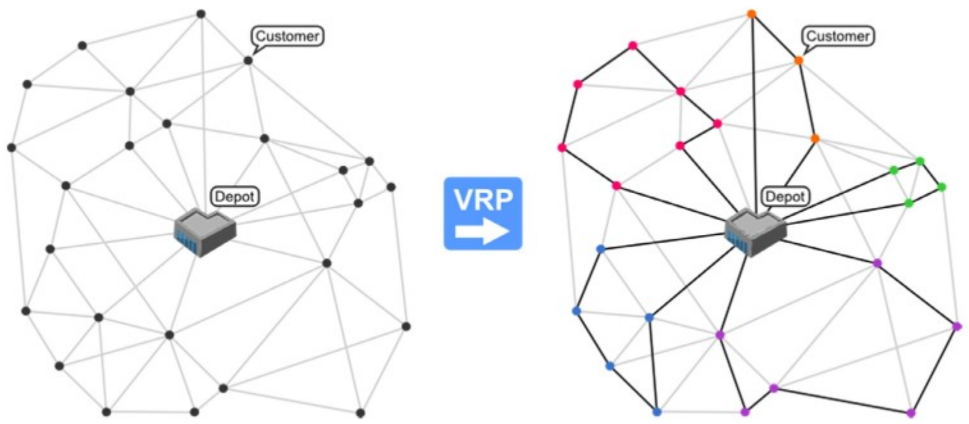

Figure 1.1: Example of VRP solution (M.A. Mohammed et al., 2017)

to the customer and back to the depot, prioritisation of each customer, and ranges of the vehicles, if the fleet is heterogeneous, are also to be considered daily.

These considerations offer a compelling reason to optimise the usage of the fleet and result in a need to address the Vehicle Routing Problem (VRP) for EVs. The optimisation problem associated with Vehicle Routing Problem (VRP) has been extensively studied and applied in distribution and collection services. In the traditional Vehicle Routing Problem, we have several customers located at different places, each with a known demand level. The vehicles leave the depot to deliver the required goods, and then they return. Only one vehicle may service each client at a time, ensuring that all of their needs are met and on the premise that only one vehicle visits each customer. Each vehicle can only travel a certain total distance with a certain amount of capacity, as seen in Fig 1.1. The term "static vehicle routing problem" (SVRP) refers to a situation in which all the details of the route are known in advance and remain unchanged after the route has started. If any constraint is placed, such as time, vehicle capacity, battery capacity, or other factors, the problem will increase in complexity. To obtain a close-to-optimal solution for this variant of VRP, heuristics, and metaheuristics are employed.

This paper gives a solution to this variant of VRP. Based on the above-stated variables, the budgeted vehicles would have to be charged based on the tasks that the vehicles are assigned. Vehicles would be assigned tasks based on battery capacity and range. The number of vehicles should be fewer than the number of routes that may be suggested, and the number of routes should be less than the number that can be offered to cover all the customers. As there is a prioritisation list of customers, there is the possibility for all the customers not to be delivered to, which is also a feasible solution.

The current project focuses on the heuristics and metaheuristics used to solve this problem. Heuristics are described as problem-specific algorithms that adopt a practical method which frequently delivers an acceptable level of accuracy for specific objectives. Heuristic techniques are often used when exact methods fail to provide an optimal solution due to problem complexity. As can be seen in Halim and Ismail (2017), the paper demonstrates that heuristic techniques can efficiently find near-optimal solutions for the TSP, which traditional methods cannot achieve in a reasonable time frame. In contrast, metaheuristics can be thought of as advanced problem-solving techniques that build upon the foundation of heuristic methods. While heuristic techniques are typically designed to solve specific classes of problems, metaheuristics are more general-purpose and can be applied to a wide range of optimisation problems. This thesis looks into heuristics and metaheuristics that are being implemented into the above-referred problem. Specifically, the heuristic utilised in this project is Nearest Neighbour Based Heuristic (NNBH) and the relevant metaheuristics utilised are Variable Neighbourhood Search (VNS) and Simulated Annealing (SA).

This project is a collaboration with the Centre of Global Eco-Innovation (CGE). It was founded in 2012 at Lancaster University and is aimed at building projects around the sustainability ethos. The centre partners with Small and Medium Enterprises (SMEs) around the region and works with the Research and Development departments of companies to tackle the issues in the environmental approach of their commercial products. For the project here, the University partnership is with Miralis Data (Clegg, 2021). Miralis Data is a software and innovation company that focuses on technological solutions to reduce carbon emissions in the transport, logistics and supply chain. Their most significant commercial product is Fuuse, a hardware-agnostic charge point management application for electric vehicles. It is aimed at both consumers, who are EV drivers, and charge point operators, who have installed commercial charge points for EV drivers to charge their vehicles. This project will be a vital piece of their new commercial product, called Fuuse Fleets. Fuuse Fleets is an end-to-end solution for logistics and supply chain enterprises to transition their fleets of vehicles to EVs. This project would help forecast the number of Electric Vehicles in their fleets based on daily customer requirements while transitioning gradually without impacting profitability

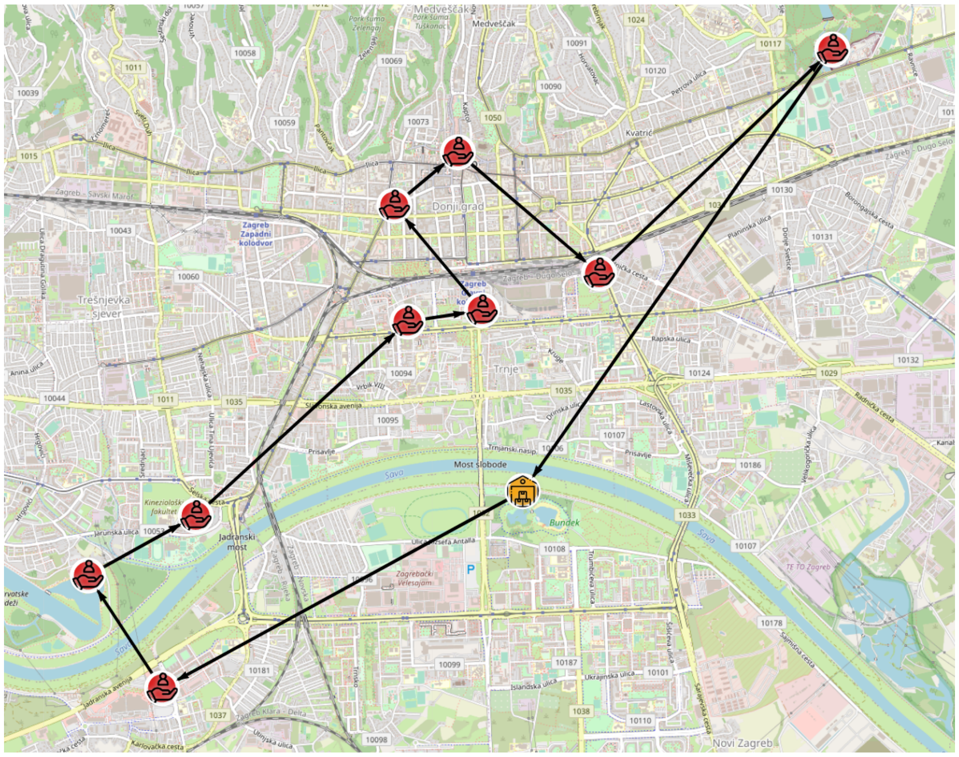

Figure 1.2: Scheduled Route for a vehicle on a specific day

from their daily revenues and deliveries. The project hopes to schedule the adequate number of transitioned vehicles required to service customers on any specific day based on customer needs as illustrated in Fig 1.2.

This report has seven chapters. The Introduction chapter familiarises the reader with the problem and briefly summarises the approach taken to solve it. Before examining the solution to the problem, the Literature Review chapter looks at the background work that researchers in the related Operational Research field have done in the past. It also describes how the chosen approach to this problem has been derived. The following Base Mathematical Model chapter explains the current problem and establishes the research problem and mathematical definitions of the variables for solution modelling. The subsequent chapter, named Heuristic Approach, is concerning the methodology used for finding a heuristical technique to determine how each vehicle could be routed based on the battery capacity of the vehicle and the priority of customers. It also includes a discussion on how the metaheuristics are implemented and integrated with the heuristics approach using the various operators.

These analytical procedures and the results obtained from them are described in the next chapter, named Computational Results. The following chapter, called Conclusion Chapter, draws upon the entire thesis. This chapter also extends a discussion of the implication of the findings to future research into this problem.

Chapter 2

Literature Review

This project aims to address the challenge of integrating charging management optimisations for electric vehicles with the Electric Vehicle Routing Problem (EVRP) by developing a tailored optimisation algorithm. The algorithm is specifically designed to incorporate unique features of the problem, requiring the utilisation of customised heuristics and metaheuristics. The objective is to devise the most effective routing algorithm that takes into account the intricacies of EVRP and the optimisation of charging management.

Since Dantzig and Ramser's investigation of the Vehicle Routing Problem (VRP) in 1959 (Dantzig and Ramser, 1959), the problem has drawn the attention of several researchers. For more than 60 years, numerous researchers have studied it given there are several practical uses for VRP. As mentioned earlier, VRP provides solutions to problems regarding a broad range of transportation and distribution topics. This includes the movement of people and goods, transportation services, and waste collection. All of these problems are significant economically, especially in industrialised nations. Companies and academics are driven by the economic incentive to reduce costs to identify the best solution and boost transportation efficiency (M. Mohammed et al., 2012).

As this research has developed over the years, the focus on various variants of the VRP has drastically increased. This comprises topics such as Vehicle Routing Problem with Time Windows (VRPTW) (Desrochers et al., 1992), Multi-Depot Vehicle Routing Problem (MDVRP) (Crevier et al., 2007), Capacitated Vehicle Routing Problem (CVRP) (Fukasawa et al., 2004), Vehicle Routing Problem with Backhauls (VRPB) (Toth and Vigo, 2002), Electric Vehicle Routing Problem (EVRP) (Lin et al., 2016), etc. This thesis also can be attributed as a niche variant of VRP, EVRP and the Travelling Repairman Problem (TRP) (Afrati et al., 1986). The paper by Desrochers et al. (1992) describes the original VRP model as a fleet of vehicles leaving a depot and visiting a set of customers to deliver goods and return to the depot. The problem that the model tries to tackle is the optimal set of routes for a fleet of vehicles to traverse to deliver goods to a given set of customers. This modified VRP has never been addressed before. However, they have been able to generate close-to-optimal answers, whose efficacy varies depending on the search space.

The paper from Lin et al. (2016) focuses on an optimal routing strategy for a generalised Vehicle Routing Problem aimed at electric vehicles. The paper focuses on how electric commercial vehicles with a restricted range may recharge at a charging station during their daily delivery (and pickup) activities. This is under the assumption that the charging station facilities are already in the service area or along the route to the location. The vehicles that depart from the depot must meet every customer exactly once to pick up or deliver commercial goods. If a vehicle's battery runs low, it will drive to the nearest charging station in the range to recharge. The paper promises definitive routes for the customer sets it has been provided and is built on a strong mathematical model that formulates all of the constraints. But this paper from Lin et al. (2016) differs from the current project in several ways. From a problem design perspective, the problem in the referenced paper has the inclusion of charging stations that the vehicles could drive to while on the route. This contrasts with this project, as this project focuses on assigning the route based on the vehicle's battery capacity and the distances between customers that the respective vehicle would have to traverse through. The paper by Lin et al. (2016) did not consider the use of heuristics for enhancing computational efficiency in their model, resulting in a substantial loss of processing and solution formulation time for larger customer instances. In contrast, this current study aims to address this limitation by exploring the application of both heuristics and metaheuristics to develop computationally efficient solutions for smaller and larger customer sets and vehicle instances.

Another aforementioned paper is the Travelling Repairman Problem (TRP) (Afrati et al., 1986). This paper creates a variation on the classical routing problem called the Travelling Salesman Problem (TSP) (Lenstra and Rinnooy Kan, 1975). TSP gives the solution to the problem: given a list of cities and distances between every city-to-city pair, derive the shortest path that would let a travelling salesman visit each city exactly once and return to the city of origin. The Travelling Repairman Problem (TRP) gives a solution to the same problem;

except that the differences are that the objective function and that not all nodes need to be visited. The objective function is aimed at maximising the profit function, and the other difference is that each city is visited less than once or equal to one. As seen in figure 2.1, the main premise of the TRP is that there are a set of speed cameras that needs to be maintained by a repairman. The set of cameras has weighted waiting times to complete the maintenance tasks by the repairman, called latency. The latency gives the set of cameras a priority of what cameras need to be maintained first and in which order they should be maintained. The current thesis takes inspiration from TRP because, in a real-world application, customers are put on a priority list. The commercial goods will need to be delivered to customers based on that priority list. In an ideal world, the priority list would be sorted in such a way that all the customers will be offered delivery in a single day's work. Unfortunately, the real world falls short in this aspect as there is a limited capacity of the fleet to serve certain prioritised customers daily. For that reason, it is safe to assume that many customers will not be visited in a single day. One of the constraints in the current thesis takes this into account.

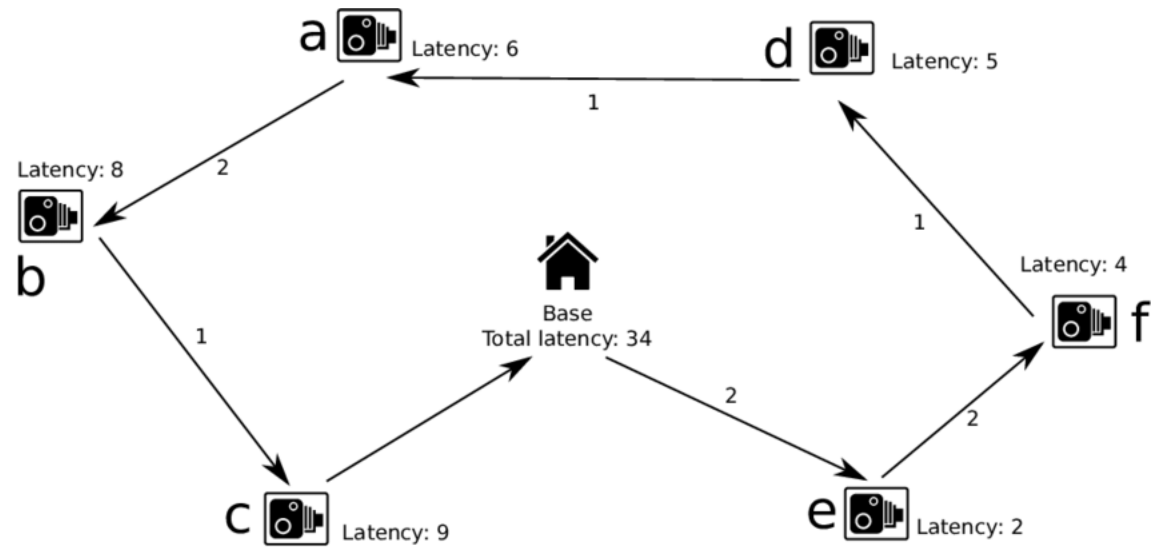

Figure 2.1: Example of Travelling Repairman Problem (Muritiba et al., 2021)

This thesis will also look at heuristics and metaheuristics to produce a feasible solution within the range of an accurate answer while improving computational efficiency. It looks at how a heuristic called the Nearest Neighbour Based Heuristic (NNBH) and metaheuristics called Variable Neighbourhood Search (VNS) and Simulated Annealing (SA) are implemented to solve this problem. Heuristics are a “class of procedures for finding acceptable solutions to a variety of difficult decision problems, that is, procedures for searching for the best solutions to optimisation problems” (Laguna and Martí, 2013).

Starting from the heuristics implemented, the Nearest Neighbour Based Heuristic is a straightforward algorithm used in this project to assign vehicles to unserved nodes in each route. The algorithm randomly selects an unserved node and assigns a vehicle to it. It then searches for nearby unserved nodes iteratively until certain limitations are encountered.

In the context of this project, three primary limitations are incorporated into the algorithm. Firstly, if all nodes have been visited, indicating that the entire route has been covered. The second limitation pertains to the energy required to visit a customer. The algorithm checks if the vehicle possesses sufficient charge to travel to the nearest unserved node and return to the depot. If the vehicle’s energy capacity is insufficient to complete this journey, the route is considered complete, and the vehicle returns to the depot. This limitation ensures that the vehicle does not venture beyond its energy constraints, preventing it from being stranded without adequate charge. The third constraint pertains to the cumulative energy consumption required for a specified number of vehicles. Specifically, these vehicles are designated to travel to the nearest customers and subsequently return to the depot. This condition involves assessing whether the upcoming vehicle can effectively embark on a journey to the nearest unvisited customer and subsequently complete the round trip back to the depot. If the aggregate distance, which encompasses the distance from the depot to the nearest customer, followed by the return trip to the depot, along with the cumulative distance travelled by the preceding vehicles, exceeds the predetermined energy budget assigned to the entire fleet, the route for that particular vehicle is concluded, and it returns to the depot. This constraint guarantees that the energy consumption remains within the specified budget, promoting sustainable fleet operation.

When any of these limitations are encountered, signifying that a route cannot be continued due to node visitation or energy constraints, the algorithm finalises the route, assigning it to the corresponding vehicle. By integrating these limitations, the Nearest Neighbour Based Heuristic algorithm effectively considers the practicalities of node visitation and energy availability, ensuring the optimisation of vehicle assignment while adhering to operational and energy limitations.

This approach of NNBH is similar to the academic paper proposed by Tavakkoli-

Moghaddam et al. (2006). However, their perspective of the problem has different constraints on the limitations of heuristics. Specifically, the vehicle capacity is not exceeded, as it is for the CVRP, and the service time for the vehicles is also not exceeded. In the current thesis, these variables are not deemed necessary as our problem is more focused on charge management juxtaposed with the Vehicle Routing Problem for EV fleets. Hence, the above variables are assumed to be not relevant to this version of the problem. The reason for choosing NNBH as the preferred heuristic was two-fold. Firstly, this heuristic is a local search method that is easy to implement, because of its approach to finding the next customer to visit. Secondly, the execution time of the heuristic is significantly quicker than many heuristics tackling the same problem, since the worst-case space complexity of the algorithm is $\Theta(\mathrm{N}^2)$.

The VNS metaheuristic is a well-known heuristic search approach that has been effectively used for a wide range of problems (Hansen et al., 2008). The metaheuristic uses the approach of iteratively moving between neighbourhoods to find a better value for the objective function. It uses a method called '*shake*' which randomly changes the neighbourhood to perform a local search on the result. The local search result is checked by criterion to confirm whether the result is acceptable for that iteration or not. VNS has been used to solve a variety of other VRPs, including Vehicle Routing Problem with Backhauls (VRPB) (Crispim and Brandão, 2005), Large scale Vehicle Routing Problem (Kytöjoki et al., 2007), Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) (Polacek et al., 2004), Vehicle Routing Problem with Time Windows (VRPTW) (Bräysy, 2003) and Open Vehicle Routing Problem (OVRP) (Fleszar et al., 2009). The algorithm's simplicity, precision, multiplicity, and efficiency are key characteristics that make the VNS metaheuristic an easy choice for an algorithm.

The Variable Neighbourhood Search (VNS) metaheuristic is employed in this context, utilising four operators to construct new neighbouring solutions. If a customer is not visited within the Nearest Neighbour Based Heuristic (NNBH) solution, two operators are employed to address this scenario. The first operator (InsertUnivisted()) attempts to include the unvisited customer in an existing solution. This involves evaluating the feasibility of adding the customer to a suitable location within a route. The second operator (SwapUnvisited()) focuses on improving the solution by swapping a low-priority customer with an unvisited customer. This exchange aims to enhance the overall quality of the solution by prioritising the unvisited customer. In cases where all customers have been visited within the NNBH solution, the solution undergoes either the 1-opt or 2-opt operator. The 1-opt operator selects a random customer from one route and transfers it to another randomly selected route. On the other hand, the 2-opt operator takes a subroute from one randomly chosen route, reverses the order of the nodes, and appends it to another randomly selected route. The solutions generated by the swap or insert operators are only accepted if they result in an increase in the total priority cost compared to the original solution. This ensures that the new solution offers improved prioritisation of customers. In the case of the 1-opt or 2-opt operator, the produced solution is accepted if it exhibits an increase in priority cost and if the vehicles have sufficient charge to accommodate the inserted route. By utilising these operators, the VNS metaheuristic explores different neighbourhood structures to enhance the quality of solutions. The operators are designed to maximise the prioritisation of customers, ensure feasibility, and optimise the utilisation of available energy resources.

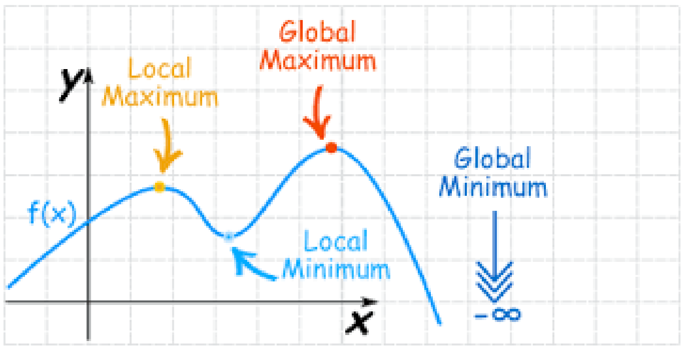

Figure 2.2: Global Maximum/Minimum and Local Maximum/Minimum (Xie, 2019)

The other metaheuristic used in this project is called Simulated Annealing (SA) and it is widely used in the VRP field. Simulated annealing operates by iteratively evaluating and modifying candidate solutions. At each iteration, the algorithm considers a neighbouring solution by applying a random modification to the current solution. If the new solution improves the objective function (i.e., decreases the cost or increases the quality), it is accepted as the new current solution. However, if the new solution is worse, it may still be accepted with a certain probability based on a temperature parameter and the magnitude of the degradation. This probabilistic acceptance allows the algorithm to escape local optima and explore different areas of the solution space. Fig 2.2 and 2.3 shows an example of the global and local minima and the trajectory SA would follow to solve the problem, respectively.

Simulated Annealing comes from a concept from metallurgy in physics that involves the heating and cooling of metal for it to alter its physical properties. Simulated annealing does not guarantee to find the global optimum but aims to find good solutions in a reasonable amount of time. The reason for choosing this approach is that the model mixes several local search methods dedicated to the problem. This produces high-quality solutions in a very short computational time compared to other methods associated with the exact VRP models (Afifi et al., 2013).

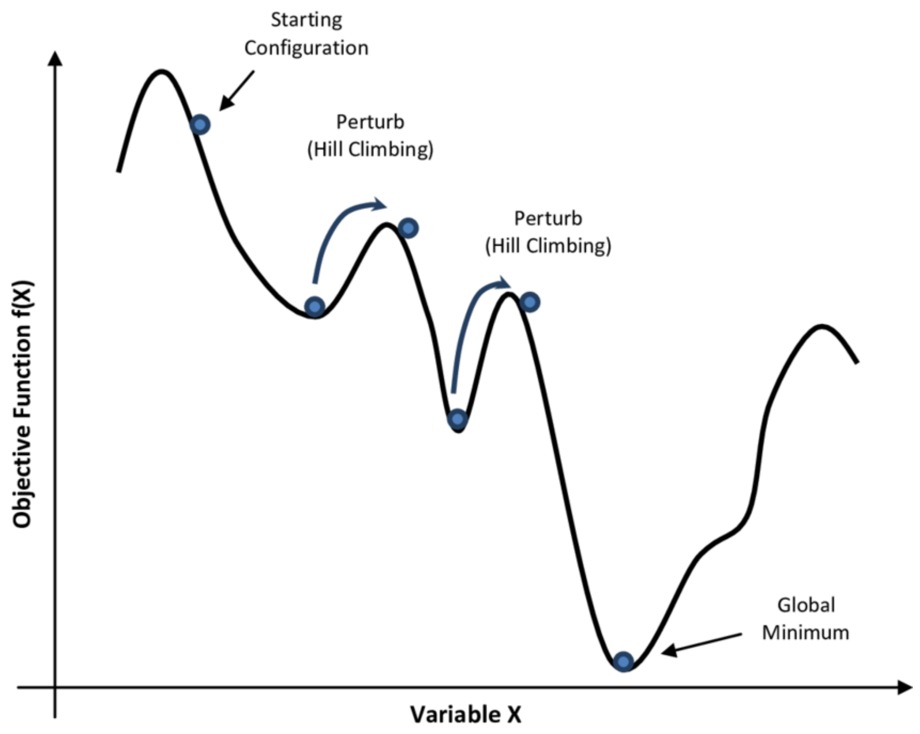

Figure 2.3: Simulated Annealing for a minimisation problem (Ghasemalizadeh et al., 2016)

To conclude this section, the literature described in this chapter was used to highlight three different parts of the project. Firstly, the current status of research in the field of Operational Research associated with VRP and EVs. Secondly, the different techniques used for finding a concrete solution to this version of VRP. Lastly, it shines a light on the uniqueness of this version of VRP and the approach to solving it. There has not been any research on VRP scenarios where the charging infrastructure has consumption budgets. Therefore, this would be a unique problem that has never been tackled before.

Chapter 3

Base Mathematical Model

This chapter defines the research problem in detail, with assumptions made to solve the problem. Using these assumptions, the graph notation is defined to propose the mathematical formulation. The mathematical formulation is described in detail after the objective function and constraints are established.

3.1 Research Problem Description

The proposed model has the following assumptions:

1. The customers must be met at most once by a single vehicle from a heterogeneous fleet of vehicles. It is important to note that not every customer needs to be visited.

2. Each route starts and ends at the depot with a sequence of customer nodes that the EV visits.

3. The fleet is heterogeneous with different ranges

4. The vehicles are all charged at the depot based on the journey they are meant to be assigned for the day. The depot has an energy budget for charging the fleet entirely and cannot be exceeded.

5. Variables such as vehicle driving speed, charging rates, delivery load and other second-order factors (like traffic jams, road conditions, elevation change in route, temperature, etc.) are not considered for this problem.

6. Energy consumption to cover a distance has a 1:1 conversion, i.e., 1 kWh of the vehicle covers 1 km of distance, for ease of analysis (Earl et al., 2018).

In the following section, the proposed model presents the notations used for this thesis. For this definition, V is the set of vehicles that are operational for the day's journey. Let N be a set of customers with a size of n that must be visited. R is a set of customers with the inclusion of the depot at the start and end of the set. Hence, let vertices 0 and $n+1$ be the depot (represented by ϕ). Sets with subscript 0 and $n+1$ indicate that they contain the depot, such as $R_0 = R_{n+1} = \phi$. The model can then be defined on a weighted and complete directed graph G = (R, A), where A = $\{(i, j)|i, j \in R, i \neq j\}$ denotes the graph's arcs.

The distance d$_{ij}$ represents the amount of distance or energy required to travel from customer i to customer j, which are the weights of the arcs in graph G. The profit gained by visiting the prioritised customer i is symbolised by p_i. The depot has a priority of 0, while the rest of the customers have a priority with a positive integer value. The vehicles in the fleet would be charged collectively by the energy budget, denoted as e. Each vehicle v can travel a range of r_v, hold a battery capacity of C_v and has an initial charge of q_v.

In this implementation, it associates a binary variable y_{ij}^v with decision variables that take a value of 1 if vehicle v is travelling from i to j and 0 otherwise, where $i, j \in R$ and $v \in V$. The variable x_i^v indicates whether the customer at i is served by vehicle v, where $i \in N$ and $v \in V$. The final decision variable is the amount of energy used to charge a vehicle v, indicated as z_v. The model aims to maximise the total priority cost.

Using these variables, the proposed mathematical formulation can be created.

3.2 Graph Notation

3.2.1 Problem Parameters

N : Customers $\{i, j, k\}$

R : Depot + Customers $\{\phi, i, j, k, \phi\}$

S : Subset of R

V: Vehicles $\{u, v\}$

q_v : Initial charging level of vehicle v

C_v : Battery capacity of vehicle v

p_i : Priority of customer i
r_v : Range of vehicle v
d_{ij} : Energy consumption to travel from i to j
e : Energy budget

3.2.2 Data

z_v: Amount of energy used to charge vehicle v
x_i^v: 1, if the customer at i is served by vehicle v ; 0, otherwise
y_{ij}^v: 1, if vehicle v is travelling from i to j ; 0, otherwise

3.3 Mathematical Formulation of Model

Objective function

$$\text{Max} \sum_{i=1}^{R} \sum_{v=1}^{V} x_i^v p_i \tag{3.1}$$

Subject to

$$\sum_{v=1}^{V} x_i^v \le 1 \qquad \forall i \in N \tag{3.2}$$

$$\sum_{i=1}^{R} y_{ij}^v = \sum_{i=1}^{R} y_{ji}^v = x_j^v \qquad \forall j \in R, v \in V, i \neq j \tag{3.3}$$

$$\sum_{j} y_{0j}^v = 1 \qquad \forall v \in V, j \in N \tag{3.4}$$

$$\sum_{i} y_{i0}^v = 1 \qquad \forall v \in V, i \in N \tag{3.5}$$

$$\sum_{(i,j)\in R} y_{ij}^v d_{ij} \le q_v + z_v \le C_v \qquad \forall v \in V, i \neq j \tag{3.6}$$

$$\sum_{v=1}^{V} z_v \leq e \qquad \forall v \in V \tag{3.7}$$

$$q_v + z_v \leq r_v \qquad \forall v \in V \tag{3.8}$$

$$\sum_{i,j \in S; i \neq j; i=1} y_{ij}^v \leq |S| - 1 \qquad \forall S \subsetneq R, |S| \geq 2, v \in V \tag{3.9}$$

$$x_i^v \in \{0,1\} \qquad \forall i, v \tag{3.10}$$

$$y_{ij}^v \in \{0,1\} \qquad \forall i, j, v \tag{3.11}$$

The objective function (3.1) maximises the sum of the priority of the served customers. Constraint (3.2) makes sure that any customer is not served by more than one vehicle. Constraint (3.3) asserts that every vehicle that visits a customer has to leave the same customer. The constraints (3.4) and (3.5) establish that every vehicle should leave and enter back into the depot respectfully. Next, equation (3.6) defines that the arc a vehicle takes to reach a customer exceeds neither the vehicle's charge nor the vehicle's maximum battery capacity. The inequalities (3.7) indicate that the energy used to charge the vehicle does not exceed the energy budget of the infrastructure that charges the vehicle, and (3.8) declares that the vehicle's journey does not exceed the range of the vehicle based on the charge it contains. Constraint (3.9) eliminates any sub-tours that might be made by any vehicle. For each subset of R, denoted as S, the maximum number of arcs between the elements in S is limited to the number of elements in the $S - 1$. Finally, constraints (3.10) and (3.11) specify that the decision variables are binary.

Chapter 4

Heuristic Approach

4.1 Nearest Neighbour Based Heuristic

As mentioned in the Introduction, the main heuristic that is utilised for the metaheuristics to have an initial solution is called Nearest Neighbour Based Heuristic (NNBH). NNBH is a simple algorithm where, for each route, a node is chosen at random and then a vehicle visits the node. The set of unserved neighbours is then searched for until one of the limitations is met. Once they are met, the iteration is restarted for a new vehicle. In this implementation, the limitations are:

- All of the nodes are visited
- The vehicle does not have enough charge to go to the node and then return to the depot
- The total cost of all the journeys is over the energy budget e

The steps of the proposed algorithm are as below:

Algorithm 1 Nearest Neighbour Based Heuristic

```
Require: [nearestNode ∈ R, currentNode ∈ R]
 1: $totalCost = 0$
 2: $distance = 0$
 3: for $v \in V$ do
 4:     $C_v = v.BatteryCapacity$
 5:     $currentNode = depot$
 6:     while $distance < C_v$ do
 7:         $nearestNode = smallestDistanceFromCurrentNode(currentNode)$
 8:         if ($nearestNode == null$ ||
 9: $calculateDistance(currentNode, nearestNode) + totalCost > e$ ||
10: $calculateDistance(currentNode, nearestNode) > C_v - distance$) then
11:             $ReturnToDepot()$
12:         else
13:             if $NearestNode.Visited == false$ then
14:                 $distance+ = calculateDistance(currentNode, nearestNode)$
15:                 $AddToRoute(nearestNode, v)$
16:                 $currentNode = nearestNode$
17:             else
18:                 continue
19:             end if
20:         end if
21:     end while
22:     $totalCost+ = distance$
23: end for
```

The algorithm above begins by assigning the variables *totalCost* of the complete solution and the *distance* of each vehicle to 0 in Step 1 and 2, respectively. Step 3 then commences the process of iterating through each vehicle. In Step 4, the battery capacity of the vehicle is reassigned to the battery capacity of the vehicle in iteration. Step 5 defines that the *currentNode* starts with the depot. The second iteration begins, from Step 6 onward, where the total distance travelled does not go beyond the battery capacity of the vehicle. While this condition is true, the nearest node from the current node is calculated and assigned to the variable *nearestNode*, as seen in Step 7. The algorithm checks through a few conditions. Firstly, whether there are no more nearest nodes to be visited, which is checked in Step 8. Secondly, if the addition of the distance, from the current node to the nearest node and back to the depot, and the total cost for the journey of the whole fleet, is greater than the energy budget assigned to the fleet. This is seen in Step 9. Lastly, whether the remaining battery left in the vehicle does not have enough charge to visit the current node and return to the depot, as shown in Step 10. If any of these conditions are true, then it enters Step 11 and the vehicle would be returned to the depot. If any of these conditions are false, however, then visiting the nearest node can be considered which is after Step 12. If the nearest node that is in consideration is not visited yet, then the distance travelled is aggregated. The current node is then the nearest node that the vehicle has travelled to and the iteration for the respective vehicle gets repeated. This is demonstrated in Steps 13 to 16. At the end of each iteration of the vehicle, the total cost is aggregated with the distance the vehicle has travelled, which is done in Step 21.

4.2 Simulated Annealing

Simulated Annealing (SA) is a stochastic optimisation algorithm technique that originated from statistical mechanics (Kirkpatrick et al., 1983) The process involves repeatedly heating a solid to a high temperature and then cooling it gradually to a lower temperature, allowing the atoms to move freely. If the cooling process is too rapid, the atoms do not have enough time to arrange themselves in an energetically favourable configuration. This analogy can be applied to combinatorial optimisation problems, where the solid represents the possible solution space, the energy corresponds to the objective function, and the minimum energy state corresponds to the optimal solution.

SA directs the search using a stochastic technique. It permits the search to continue to an adjacent state even if the move reduces the value of the objective function. The initial local search approach is guided in the following way by SA. If a move to a neighbour *X'* in the neighbourhood *N(X)* reduces or maintains the objective function value, the move is always allowed. More specifically, if $\Delta \geq 0$, the solution *X'* is accepted as the new current solution, where $\Delta = Cost(X^0) - Cost(X)$ and $Cost(X)$ is the value of the objective function. To move from one local optimum to another, there needs to be a chance for inferior solutions to be accepted. Therefore, $e^{(-\Delta/T)}$ calculates the probability that allows inferior solutions to be accepted and traverse away from a local optimum. T is a temperature parameter that ranges from a relatively high number to a smaller number close to zero. A cooling schedule sets the starting and incremental temperature levels at each algorithm stage, which controls these values. The algorithm is demonstrated in Algorithm 2.

4.2.1 Parameters Description

The SA features an inner loop and an outer loop. The inside loop governs the attainment of equilibrium at the current temperature, whereas the outside loop governs the pace of temperature decline. The following are the SA parameters:

EL	(Epoch Length) number of solutions accepted in each temperature for achieving equilibrium
MTT	maximum number of consecutive temperature trails
T_0	initial temperature
α	rate of the current temperature decrease
X	a feasible solution
$Cost(X)$	the value of objective function for X
n	counter for the number of accepted solutions in each temperature
r	counter for the number of consecutive temperature trails, where T_r is equal to temperature in iteration r

4.2.2 Initial Solution Generation

The initial solution for the metaheuristic is generated from the NNBH approach explained in Section 4.1 in Algorithm 1. The NNBH approach would be suitable for small to medium-sized problem sets, however, in larger-sized problems, where there are more than 20-30 customer nodes to be visited, the SA mechanism would help. This example would need to use several operators for improving the initially obtained solutions from the NNBH approach.

4.2.3 Simulated Annealing Algorithm

Algorithm 2 Simulated Annealing with VNS neighbourhood generation method

```
Require: [$r = 0, X^{best} = \emptyset$]
 1: Generate $X^0$ in NNBH algorithm
 2: $X^{best} = X^0$
 3: while $r < MTT$ and $T_r > 0$ do
 4:     $n = 0$
 5:     while $n < EL$ do
 6:         Select an operator by Variable Neighbour Search and run over $X^n$ as $X^n \mapsto X^{new}$
 7:         $\Delta Cost = Cost(X^{new}) - Cost(X^{best})$
 8:         Generate $y \to U(0,1)$ randomly
 9:         Set $Z = e^{(-\Delta Cost/T_r)}$
10:         if $\Delta Cost > 0$ or $y < Z$ then
11:             $X^{best} = X^{new}$
12:             $n = n + 1$
13:             $X^n = X^{new}$
14:         else if $y < Z$ then
15:             $n = n + 1$
16:             $X^n = X^{new}$
17:         end if
18:     end while
19:     $r = r + 1$
20:     $T_r = T_{r-1} - \alpha * T_{r-1}$
21: end while
22: Return $X^{best}$
```

The SA algorithm starts with generating an initial solution from the NNBH, named X^0, in Step 1 and the last best solution, named X^{best}, being X^0 in Step 2. The first do-while loop begins by checking if either the temperature trails counter (named r) is less than the *MTT* or temperature in the current iteration, which is showcased in Step 3. The number of accepted solutions counter (named n) is assigned to 0 in Step 4. Step 5 moves into the second loop where the accepted solution counter is less than the Epoch Length, named *EL*. The following step in Step 6 uses the Variable Neighbourhood Search to have a new neighbouring solution to compare with the acceptance criterion. The VNS algorithm outputs a new solution that gets saved as X^{new}. Step 7 checks the change in priority costs between the current best solution and the new solution saved as $\Delta Cost$. It also generates a random number between 0 and 1 in Step 8. In Step 9, the probability of accepting the new solution is calculated called Z which is used later in the algorithm. If the change in the priority costs is greater than the priority costs in the current best solution, as seen in Step 10, it passes the acceptance criteria and the solution is accepted. Steps 11-13 save the new solution as the current best solution (Step 11), increase the accepted solutions counter (Step 12), and save it to the list of accepted solutions for each iteration (Step 13). If it does not pass the acceptance criteria, then it continues the iteration through Step 13. If the randomly generated number is less than the probability, the solution is accepted as seen in Steps 14 - 17. Steps 19 and 20 would see the consecutive temperature trails counter being incremented and the temperature data-set being aggregated with the rate of the current temperature decreased as the offset. Finally, Step 22 prints out the best solution from the metaheuristic.

4.3 Variable Neighbourhood Search

In the context of the neighbourhood generation mechanism, VNS is used to move into feasible space and obtain the neighbourhood solution. In this implementation, the search variably checks through modified versions of 1-opt, 2-opt, Swap and Insert operators. The performance of operators in 1-opt and 2-opt methods between two routes can be seen in Figure 4.1. In the referenced figure, 1-opt moves the customer node C from the black route to the blue route. The 2-opt algorithm in the figure shows that a route from the black route is extracted (i.e., C-D), the route is reversed, and then inserted into the blue route.

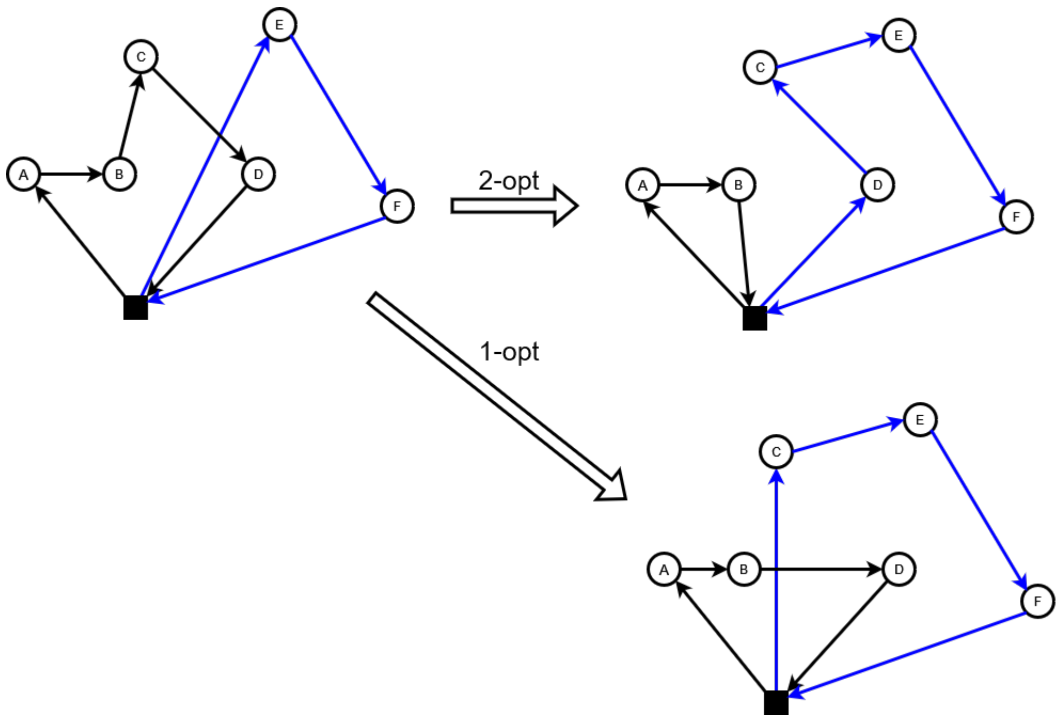

Figure 4.1: Example of 1-opt and 2-opt

The fundamental logic of 1-opt and 2-opt operators is two routes of two different vehicles in a feasible solution are randomly chosen. For 1-opt, one customer node is randomly chosen from the first route and inserted into a random location in the second route to find the best location. Contrastingly, 2-opt selects a random sub-route from the first route and inserts it at a random index in the second route. In this implementation, the acceptance criterion in 1-opt and 2-opt is unique from any other implementation. For a solution to be accepted, we first check if the aggregation of the customers' priority in the inserted route is better or worse. If it is better and if the vehicle has enough battery to do the proposed route, then the solution is accepted and returned into the Simulated Annealing process. The pseudocode of the algorithm for 1-opt can be seen below in Algorithm 5, 2-opt can be seen in Algorithm 6, the Swap operator in Algorithm 7, and the Insert operator in Algorithm 8.

4.3.1 Variable Neighbourhood Search Algorithm

Using these four operators, the Variable Neighbourhood Search would first check if there are any unvisited customers from the solution produced by the Nearest Neighbour Based Heuristic. If there are, the unvisited customers would randomly be either swapped with a visited customer or inserted into a random route. The solution from either of the two edge case operators would then be passed to the 1-opt or 2-opt operators randomly. Using the solution produced by either 1-opt or 2-opt, it passes through the acceptance criterion. If it passes through the acceptance criterion, it chooses that route and makes it more biased toward the operator that chose the route. If it does not pass the acceptance criterion, the search is iterated again until it finds a better solution. The biased approach makes finding the solution to the problem more variable. The algorithm for the Variable Neighbourhood Search is found in Algorithms 3 and 4.

Algorithm 3 Variable Neighbourhood Search

```
Require: [$kmax, route \in route = X^n$]
 1: $k = 1$
 2: while $k < kmax$ do
 3:     Randomly generate a number between 0 and 1000 $\rightarrow randomNumber$
 4:     if $unvisitedCustomers.size > 0$ then
 5:         Randomly generate a 1 or 0 ($unvisitedRandomNumber$)
 6:         if $unvisitedRandomNumber > 0$ then
 7:             $route = SwapUnvisitedCustomers()$
 8:         else
 9:             $route = InsertUnvisitedCustomer()$
10:         end if
11:     else if $randomNumber < NSV$ then
12:         $twoOptedRoute = 2opt(route)$
13:         $route = NeighbourhoodChange(twoOptedRoute, route)$
14:     else
15:         $oneOptedRoute = 1opt(route)$
16:         $route = NeighbourhoodChange(route, oneOptedRoute)$
17:     end if
```

```
18:     if $k = 1$ then
19:         break
20:     end if
21: end while
```

Algorithm 4 Neighbourhood Change Algorithm

```
Require: [$twoOptedRoute, oneOptedRoute$ passed from Algorithm 3]
 1: Calculate the priority cost ($onePriority$) of the 1-optedRoute
 2: Calculate the priority cost ($twoPriority$) of the 2-optedRoute
 3: if $onePriority != twoPriority$ then
 4:     $k = 1$
 5:     if $onePriority > twoPriority$ then
 6:         $NSV -= 2$
 7:         Return $oneOptedRoute$
 8:     else
 9:         $NSV += 2$
10:         Return $twoOptedRoute$
11:     end if
12: else
13:     $k++$
14: end if
```

Variable Neighbourhood Search is a 3-phased local search method. The first phase is called the *Shake* function. The algorithm uses this phase to escape from the local optima when the local search gets stuck. It helps the diversification in the search by perturbing the incumbent solution and resorting to a different solution in the neighbourhood under exploration. The second phase is the *Local search* method, which is a search for finding the local optima of the neighbourhood selected. The final stage is the *Neighbourhood Change* where the local optima are checked to verify if it is the maximised value or not. In this implementation, the *Shake* and *Local search* phases are located in all three operators. Hence starting with Algorithm 3, Step 1 creates an iterator variable called *k*. Step 3 creates a *randomNumber* ranging from 0-1000, which would be used as the *coin flip* to select the operator chosen. The range of 0-1000 allows the opportunity for an impartial operator to provide a better solution, as compared to a binary 0-1 variable. The *NSV* is the deciding factor on probabilities of getting the right operator based on previous successful solutions. If any unvisited customers are missed in the NNBH heuristic, either the *SwapUnvisited()* or *InsertUnvisited()* is selected randomly. If the *randomNumber* is smaller than *NSV*, the 2-opt method is selected, otherwise, 1-opt is pursued. The two operators would result in a route after the *Shake* and *Local search* functions and the result is passed into the *NeighbourhoodChange()* function. This function's algorithm is seen in Algorithm 4.

Algorithm 4 begins with the calculation of the priority costs of both routes passed on from Algorithm 3, named *onePriority* and *twoPriority*. If there is a difference between the priority costs of both routes, then the selection for an operator begins. The algorithm then compares the two priority costs, to see which is the better solution. If the route produced by the 1-opt is better than the 2-opt, the variable *NSV* will be decreased by 2. Otherwise, the variable will be increased by 2. This is to increase the chance of using the more successful operator. The increment/decrement of 2 allows the variable to be biased enough to pick one operator over the other. On the other hand, it is not incremented/decremented too much where the model is forcing the *randomNumber* to only pick one operator over the other for every iteration.

4.3.2 1-opt Algorithm

Algorithm 5 Priority-based 1-opt algorithm

```
Require: [$soln = X^n$]
1: Generate a randomRoute from soln
2: Select a random customer, removedCustomer, from the selected route
3: Get the index of the selected customer, removedIndex, in the selected route
4: Remove the selected customer from the selected route
5: Select a random route, insertedRoute, from the current solution, soln
6: Select a random index, insertedIndex, in the selected route
7: Insert the removed customer in the selected route at the selected index
8: $changeInDistanceInsertedRoute \leftarrow CalculateDistanceDelta(insertedIndex)$
9: $newPriorityCostForInsertedRoute$
10:     $\leftarrow insertedRoute.PriorityCost() + removedCustomer.priorityCost$
```

```
11: if $newPriorityCostInInsertedRoute > insertedRoute.PriorityCost$ then
12:     if $insertedRoute.cost + changeInDistanceInsertedRoute$
13:             $\leq insertedRouteVehicle.BatteryCapacity$ then
14:         Return modified route
15:     else
16:         break
17:     end if
18: else
19:     break
20: end if
```

The 1-opt algorithm above starts with the assumption that the solution used in this iteration is the last solution from the solution set produced by the Simulated Annealing process. Using this assumption, Steps 1 and 2 randomly pick a route from the solution and a customer from the route, respectively. The index of the removed customer from the route is also saved, in Step 3. Step 4 removes the selected customer from the selected route. Step 5 picks a random route where the removed customer can be inserted. Step 6 selects a random index from the route for the removed customer to be injected into. Step 7 inserts the customer at the selected index into the selected route. Step 8 calculates the change in distance costs by deducting the distance between customers before and after the inserted customer while aggregating the distance connecting the inserted customer to the adjacent customers on either side. The formula for the change in the length of inserting customer *v1* into a route is found in equation 4.1 below. In the formula, *dist(x,y)* calculates the distance between customer nodes x and y using the Euclidean distance formula.

$$\text{DistanceDelta} = -\text{dist(v1-1, v1+1)} + \text{dist(v1-1, v1)} + \text{dist(v1, v1+1)} \tag{4.1}$$

In Steps 9 and 10, the new priority costs of the inserted route is calculated. From Step 11, the acceptance criterion is checked. Firstly, it checks if there is an increase in prioritisation of the route. If there is an increase in prioritisation costs (Step 11), then the solution is accepted on the condition that the vehicle has enough battery to do the journey (Step 12). If it does not pass the acceptance criterion, then the solution is not passed into the Simulated Annealing process (Steps 16 & 19).

4.3.3 2-opt Algorithm

Algorithm 6 Priority-based 2-opt algorithm

```
Require: [$soln = X^n$]
 1: Randomly select a route removedRoute from the current solution soln
 2: Randomly select two distinct customers removedCustomer1 and removedCustomer2
    from removedRoute
 3: Find the index of removedCustomer1 (removedIndex1) and removedCustomer2
    (removedIndex2) in removedRoute
 4: Create a new route (extractedRoute) from removedCustomer1 to removedCustomer2
    and reverse the order of customers in extractedRoute
 5: Remove extractedRoute from removedRoute
 6: Randomly select another route (insertedRoute) from the current solution soln
 7: Randomly select an index (insertedIndex) in insertedRoute
 8: Insert extractedRoute into insertedRoute starting at index insertedIndex
 9: $changeInDistanceInsertedRoute \leftarrow CalculateDistanceDelta(extractedRoute)$
10: Compute the new priority cost for insertedRoute due to the insertion of extractedRoute
    (newPriorityCost)
11: if ($newPriorityCost > insertedRoute.PriorityCost()$) then
12:    if $insertedRoute.cost + changeInDistanceInsertedRoute$
13:          $\leq insertedRouteVehicle.BatteryCapacity$ then
14:       Return modified route
15:    else
16:       break
17:    end if
18: else
19:    break
20: end if
```

The 2-opt algorithm follows a similar approach to the 1-opt algorithm with the difference of a route being moved from one route to another rather than a customer. Steps 1-3 focus on the end customer nodes that need to be extracted with the indexes of both to be remembered. Step 4 focuses on creating the route that has all customer nodes in between *removedCustomer1* and *removedCustomer2* and reverses it. Step 5 proceeds with the removal of the extracted route from the selected route. Steps 6 & 7 select a random route from the solution and a random index from the route that the extracted route can be inserted into. The extracted route is inserted at the selected index, as seen in Step 8. Step 9 calculates the change in cost in *insertedRoute* when the extracted route is inserted. The formula for the *DistanceDelta* in this implementation, where the customers *v1* and *v2* are the end customer nodes, can be found in equation 4.2. Similar to 1-opt, the *dist(x,y)* formulation follows the Euclidean distance formula.

$$\text{DistanceDelta} = \text{dist(v1-1, v1)} + \text{dist(v2, v2+1)} + \text{dist(v1, v2)} - \text{dist(v1-1, v2+1)} \quad (4.2)$$

Step 10 shows the computation of the change in distance cost. From Steps 11 to 20, the acceptance criterion checks results are the same as 1-opt.

4.3.4 Swap operator

Apart from these two operators, there is also another operator that is used for edge cases called the Swap operator. The requirement of this operator is that the metaheuristic only works with the solution given if the solution from the NNBH does not visit some of the customers. As such the metaheuristic would not consider any unvisited customers. This is solved with a swap operator. This operator randomly picks a route from the solution, picks the least prioritised customer from the route and removes it. From the removed customer, the index of the customer is recorded and an unvisited customer from NNBH is inserted into the route. If it passes the same acceptance criterion, the swap is accepted and returned to the Simulated Annealing process. The algorithm follows very similarly to the *1-opt* algorithm, with the key difference that the removed customer is the lowest prioritised customer from the removed route. The algorithm can be found below in Algorithm 7.

Algorithm 7 Swapping Unvisited Customer

```
Require: [$soln = X^n, unvisitedNodes.size >= 1 \in unvisitedNodes \subsetneq N$]
 1: Choose a route randomly from the existing solution ($insertedRoute$)
 2: Find the lowest priority customer in the chosen route
 3: Insert an unvisited node at the position of the removed customer in the route
 4: Calculate the change in priority of the modified route
    ($changeInPriorityInInsertedRoute$)
 5: Calculate the change in distance of the modified route ($changeInDistanceInsertedRoute$)
 6: if $changeInPriorityInInsertedRoute > insertedRoute.PriorityCost()$ then
 7:    if $insertedRoute.cost + changeInDistanceInsertedRoute$
 8:          $\leq insertedRouteVehicle.BatteryCapacity$ then
 9:       Return modified route
10:    else
11:       break
12:    end if
13: else
14:    break
15: end if
```

The Swap algorithm above is very similar to the 1-opt algorithm with a couple of very key differences between the two algorithms. First, the *insertedRoute* and the *removedRoute* are the same as the route that the customer is being removed from is the same route where the unvisited customer is being inserted in. Second, if the solution is accepted, based on the acceptance criteria, the customer that is inserted is removed from the data set that contains all of the unvisited customers. The customer that is swapped with the inserted customer is inserted into the dataset as that customer is now unvisited. This helps increases the prioritised customers being visited on a route.

4.3.5 Insert operator

Another operator that is used to reduce the number of unvisited customers produced by the Nearest Neighbour Based Heuristic is the Insert operator. The reason for using this operator is to increase the number of visited customers in the final solution produced, which increases the OFV of the final solution. This operator randomly picks a route from the solution and picks a random index from the route. After selecting the index, a random unvisited customer is selected and inserted at the selected index from the selected route. If the vehicle assigned for the selected route can travel the extra distance taken to visit the selected customer, then the solution will be accepted. This accepted solution will then be returned to the Simulated Annealing process. The algorithm is found below in Algorithm 8.

Algorithm 8 Inserting Unvisited Customer

```
Require: [$soln = X^n, unvisitedNodes.size >= 1 \in unvisitedNodes \subsetneq N$]
 1: Choose a route randomly from the existing solution ($insertedRoute$)
 2: Choose an index randomly from the selected route ($insertIndex$)
 3: Insert an unvisited node at the index $insertIndex$ in $insertedRoute$)
 4: Calculate the change in distance of the modified route ($changeInDistanceInsertedRoute$)
 5: if $insertedRoute.cost + changeInDistanceInsertedRoute$
 6:         $\leq insertedRouteVehicle.BatteryCapacity$ then
 7:     Return modified route
 8: else
 9:     break
10: end if
```

The acceptance criteria for this operator are different from the other operators for a logical reason. When a new customer is added to a randomly selected route, the priority costs will only increase. This is because the priority costs of all of the customers are positive integer values. So passing the final solution through the condition of checking whether there is an increase in priority costs is redundant. Hence, the only checking criteria that need to be validated is if the increase in distance can be traversed by the assigned vehicle.

4.3.6 Heuristic Approach Flowchart

An overview of the entire heuristic approach can be seen in 4.2. This includes the Nearest Neighbour Based Heuristic (NNBH) and the metaheuristics Variable Neighbourhood Search (VNS) and Simulated Annealing (SA) working in tandem.

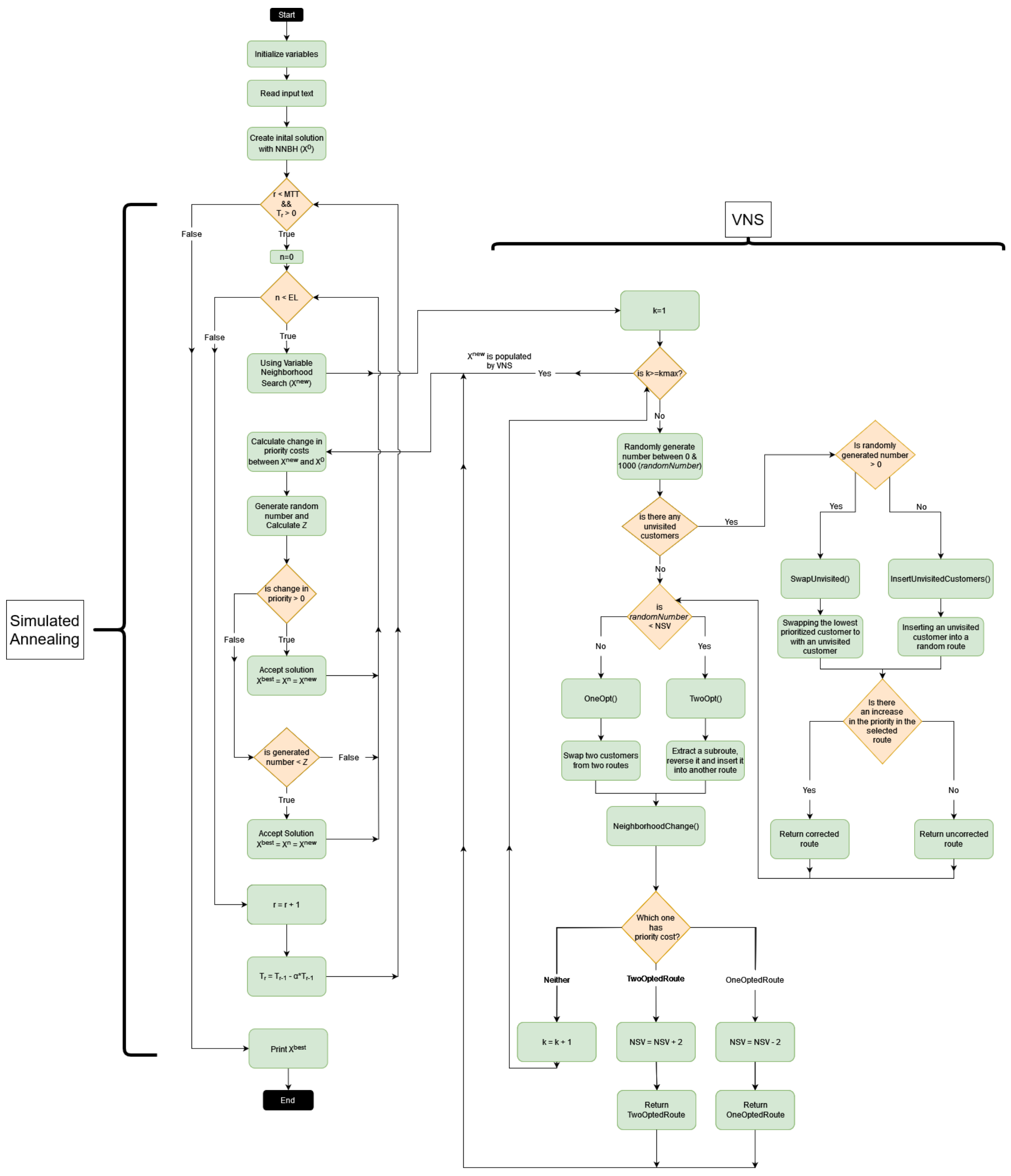
Start
Initialize variables
Read input text
Create inital solution with NNBH (X^0)
r < MTT && T_r > 0
False
True
n=0
n < EL
False
True
Using Variable Neighborhood Search (X^{new})
Calculate change in priority costs between X^{new} and X^0
Generate random number and Calculate Z
is change in priority > 0
False
True
Accept solution $X^{best} = X^n = X^{new}$
is generated number < Z
False
True
Accept Solution $X^{best} = X^n = X^{new}$
r = r + 1
$T_r = T_{r-1} - \alpha * T_{r-1}$
Print X^{best}
End
Simulated Annealing
VNS
k=1
X^{new} is populated by VNS
Yes
is k>=kmax?
No
Randomly generate number between 0 & 1000 (randomNumber)
is there any unvisited customers
Yes
No
Is randomly generated number > 0
Yes
No
SwapUnvisited()
InsertUnvisitedCustomers()
Swapping the lowest prioritized customer to with an unvisited customer
Inserting an unvisited customer into a random route
Is there an increase in the priority in the selected route
Yes
No
Return corrected route
Return uncorrected route
is randomNumber < NSV
No
Yes
OneOpt()
TwoOpt()
Swap two customers from two routes
Extract a subroute, reverse it and insert it into another route
NeighborhoodChange()
Which one has priority cost?
Neither
TwoOptedRoute
OneOptedRoute
k = k + 1
NSV = NSV + 2
NSV = NSV - 2
Return TwoOptedRoute
Return OneOptedRoute

Figure 4.2: Flowchart of the Heuristic Approach

Chapter 5

Computational Results

The computational results are presented in four sections. The first section is to define the environment in which the model is developed. The second section looks at an example of what the solution outputted looks like. The third and fourth section looks at the results of small and large instances of the problem, respectively.

5.1 Model Description

The model is developed in Microsoft Visual Studio C# 2019 and performed on an Intel(R) Core(TM) i7-10750H machine at 2.60GHz, with 16GB of RAM, running on a 64-bit platform on a Windows 11 Operating System. The Base Mathematical Model, as defined in Chapter 3, is also implemented in the same computer with the CPLEX library for the C# .Net framework.

The model is solved for small data on a heterogeneous fleet of vehicles as seen in Table 5.1. The data is derived from an electric commercial truck brand named Volta. Their website showcases the two main electric truck types; a standard range and a long-range (Anonymous, 2023). The test data uses a combination of the two types of fleets to incorporate some heterogeneity into the fleet. It solves on a small instance of 21 customers each with coordinates on an X-Y graph and priorities assigned to them. The higher the customer priority, the higher the priority for the fleet to deliver goods to them. The origin of the X-Y graph (i.e. coordinates of (0,0)) is the depot from which the vehicles would depart. The distances between each of the customers and the individual customer details are found in Table 5.3 and Table 5.2 respectively.

Fleet	**Van1**	**Van2**	**Van3**	**Van4**	**Van5**	**Van6**	**Van7**	**Van8**	**Van9**	**Van10**	**Van11**	**Van12**	**Van13**	**Van14**	**Van15**	**Van16**	**Van18**	**Van20**
Battery Capacity (KwH)	150	150	225	225	225	150	150	225	150	150	150	150	150	225	225	225	225	225
Range (miles)	150	150	225	225	225	150	150	225	150	150	150	150	150	225	225	225	225	225

Table 5.1: Fleet Information

Customers	**x**	**y**	**Priority**
D	0	0	0
1	10	20	1
2	30	10	2
3	-10	-20	2
4	-30	-40	3
5	-50	-10	3
6	45	-10	5
7	60	-20	3
8	90	25	3
9	60	50	3
10	21	49	3
11	15	39	3
12	38	42	4
13	28	19	2
14	-18	-49	5
15	30	27	2
16	-41	29	1
17	4	33	4
18	50	-19	5
19	-2	-41	4
20	-36	39	3
21	21	39	3

Table 5.2: Customer Details

Distance	**D**	**1**	**2**	**3**		**5**		**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
D	0	22.36068	31.62278	22.36068	50	50.9902	46.09772	63.24555	93.40771	78.1025	53.31041	41.78516	56.63921	33.83785	52.20153	40.36087	50.21952	33.24154	53.48832	41.04875	53.07542	44.29447
1	22.36068	0	22.36068	44.72136	72.11103	67.08204	46.09772	64.03124	80.1561	58.30952	31.01612	19.64688	35.60899	18.02776	74.46476	21.18962	51.78803	14.31782	55.86591	62.16912	49.76947	21.9545
2	31.62278	22.36068	0	50	78.1025	82.46211	25	42.42641	61.84658	50	40.02499	32.64966	32.98485	9.219544	76.05919	17	73.4983	34.71311	35.22783	60.20797	72.09022	30.36445
3	22.36068	44.72136	50	0	28.28427	41.23106	55.9017	70	109.6586	98.99495	75.6439	64.07808	78.40918	54.45181	30.08322	61.7171	57.98276	54.81788	60.00833	22.47221	64.4748	66.64833
4	50	72.11103	78.1025	28.28427	0	36.05551	80.77747	92.19544	136.4734	127.2792	102.5768	90.91755	106.527	82.73452	15	89.93887	69.87131	80.5295	82.71034	28.01785	79.22752	94.03191
5	50.9902	67.08204	82.46211	41.23106	36.05551	0	95	110.4536	144.3087	125.2996	92.31468	81.40025	102.2155	83.21658	50.44799	88.14193	40.02499	69.02898	100.4042	57.14018	50.96077	86.26703
6	46.09772	46.09772	25	55.9017	80.77747	95	0	18.02776	57.00877	61.84658	63.69458	57.45433	52.46904	33.61547	74.09453	39.92493	94.42987	59.4138	10.29563	56.30275	94.66784	54.56189
7	63.24555	64.03124	42.42641	70	92.19544	110.4536	18.02776	0	54.08327	70	79.25907	74.20243	65.78754	50.44799	83.21658	55.75841	112.2586	77.10383	10.04988	65.45991	112.681	70.72482
8	93.40771	80.1561	61.84658	109.6586	136.4734	144.3087	57.00877	54.08327	0	39.05125	73.05477	76.29548	54.70832	62.28965	130.9198	60.03332	131.0611	86.37129	59.46427	113.2254	126.7754	70.40597
9	78.1025	58.30952	50	98.99495	127.2792	125.2996	61.84658	70	39.05125	0	39.01282	46.32494	23.4094	44.55334	126.0357	37.80212	103.1601	58.5235	69.72087	110.1136	96.62815	40.5216
10	53.31041	31.01612	40.02499	75.6439	102.5768	92.31468	63.69458	79.25907	73.05477	39.01282	0	11.6619	18.38478	30.80584	105.4751	23.76973	65.14599	23.34524	73.92564	92.89241	57.87055	10
11	41.78516	19.64688	32.64966	64.07808	90.91755	81.40025	57.45433	74.20243	76.29548	46.32494	11.6619	0	23.19483	23.85372	93.98404	19.20937	56.88585	12.52996	67.74216	81.78631	51	6
12	56.63921	35.60899	32.98485	78.40918	106.527	102.2155	52.46904	65.78754	54.70832	23.4094	18.38478	23.19483	0	25.07987	106.8504	17	80.06248	35.17101	62.16912	92.13577	74.06079	17.26268
13	33.83785	18.02776	9.219544	54.45181	82.73452	83.21658	33.61547	50.44799	62.28965	44.55334	30.80584	23.85372	25.07987	0	82.0975	8.246211	69.72087	27.78489	43.909	67.08204	67.05222	21.18962
14	52.20153	74.46476	76.05919	30.08322	15	50.44799	74.09453	83.21658	130.9198	126.0357	105.4751	93.98404	106.8504	82.0975	0	89.88882	81.32035	84.89994	74.32362	17.88854	89.82205	96.25487
15	40.36087	21.18962	17	61.7171	89.93887	88.14193	39.92493	55.75841	60.03332	37.80212	23.76973	19.20937	17	8.246211	89.88882	0	71.02816	26.68333	50.15974	75.15318	67.08204	15
16	50.21952	51.78803	73.4983	57.98276	69.87131	40.02499	94.42987	112.2586	131.0611	103.1601	65.14599	56.88585	80.06248	69.72087	81.32035	71.02816	0	45.17743	102.8834	80.13114	11.18034	62.80127
17	33.24154	14.31782	34.71311	54.81788	80.5295	69.02898	59.4138	77.10383	86.37129	58.5235	23.34524	12.52996	35.17101	27.78489	84.89994	26.68333	45.17743	0	69.42622	74.24284	40.4475	18.02776
18	53.48832	55.86591	35.22783	60.00833	82.71034	100.4042	10.29563	10.04988	59.46427	69.72087	73.92564	67.74216	62.16912	43.909	74.32362	50.15974	102.8834	69.42622	0	56.46238	103.7304	64.84597
19	41.04875	62.16912	60.20797	22.47221	28.01785	57.14018	56.30275	65.45991	113.2254	110.1136	92.89241	81.78631	92.13577	67.08204	17.88854	75.15318	80.13114	74.24284	56.46238	0	86.92526	83.24062
20	53.07542	49.76947	72.09022	64.4748	79.22752	50.96077	94.66784	112.681	126.7754	96.62815	57.87055	51	74.06079	67.05222	89.82205	67.08204	11.18034	40.4475	103.7304	86.92526	0	57
21	44.29447	21.9545	30.36445	66.64833	94.03191	86.26703	54.56189	70.72482	70.40597	40.5216	10	6	17.26268	21.18962	96.25487	15	62.80127	18.02776	64.84597	83.24062	57	0

Table 5.3: Distance Matrix (in miles)

A few parameters were assigned for the metaheuristics, specifically on the SA and VNS methods. Table 5.4 shows the parameters assigned for this implementation in both Metaheuristics.

Parameter sections	**Variable Name**	**Value**
Energy budget for the charging facilities	e	1000
Simulated Annealing	EL MTT T_0 α	100 200 50 0.95
Variable Neighbourhood Search	$kmax$ NSV	100 500

Table 5.4: Parameter Description

5.2 Heuristic Solution

A graph solution for 5 vehicles with 15 customers can be seen in Figure 5.1. Each vehicle is associated with a colour, with a key of the colour-vehicle association on the top left of the solution. The depot is numbered 0 and every customer node is numbered between $1-15$. The distance between the selected customers is shown above the route. Hence from the referenced figure, it can be seen that Van 1 has taken route $0-10-11-0$, Van 2 takes route $0-14-4-0$, Van 3 takes route $0-15-12-8-6-0$, Van 4 covers route $0-5-0$, and Van 5 travels the route $0-15-12-8-6-0$. This fits the ranges of the vans as they cover ranges of $106.76, 117.20, 215.18, 101.98, 221.54$ for Van $1,2,3,4$, and 5 respectively. The distance cost of the whole route is 762.66 which is less than the energy budget. The priority value of the total solution is 38.

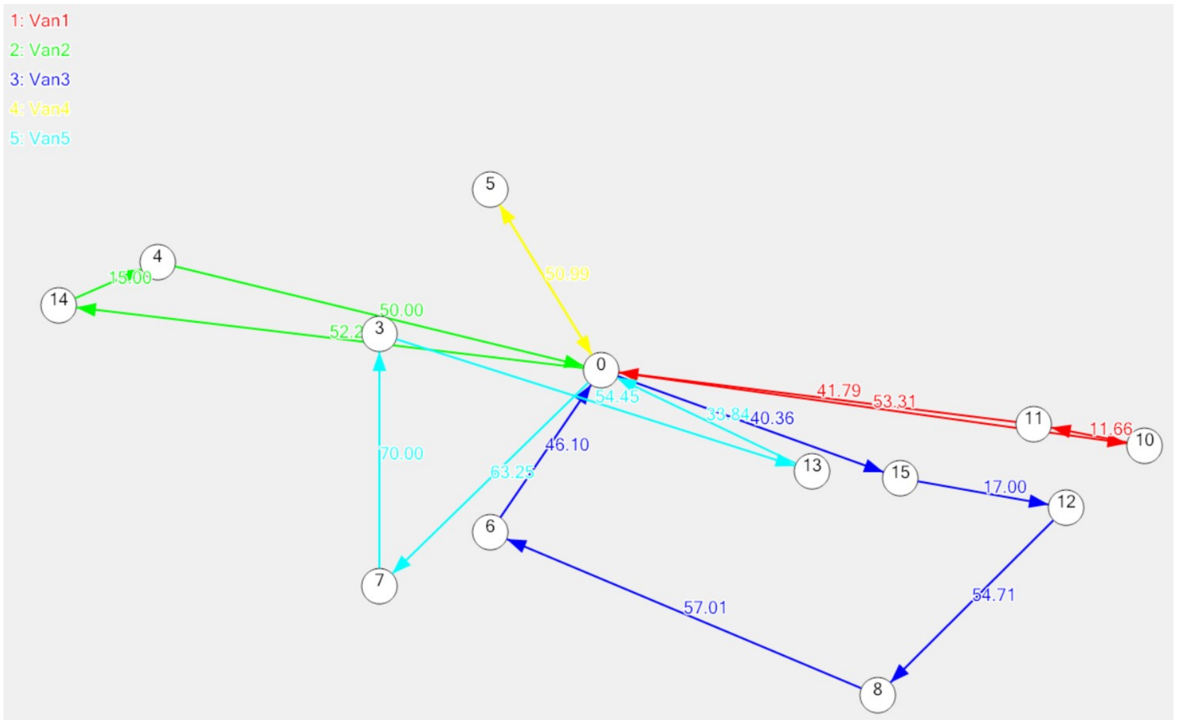

Figure 5.1: Graph solution Example

5.3 Small Instances Results

The results of the small instances can be seen below in Table 5.5. The first column (named *Instance*) shows the index of the test done on smaller instances of the problem to validate the heuristic model. The second column (named *Customers*) shows that the instances delivers to customers ranging from 2 to 16. In the following column (named *Vehicles*) the number of vehicles used for the delivery ranged from 2 to 5.

The fourth column is called *Base Mathematical Model*, which demonstrates the data collected by the Base Mathematical Model. The fourth column has 3 sub-columns, which will be referred to as 4.1, 4.2, and 4.3. Sub-column 4.1, named *OFV* has information about the objective function value resulting from the Mathematical Model. Sub-column 4.2, named *Customers Visited*, displays the customers visited in each solution produced by the Base Mathematical Model. The computational time for the Base Mathematical Model is also recorded in seconds, as seen in Sub-column 4.3 as *CPU Time (seconds)*. This Base

Mathematical Model has been used to benchmark the problem in similar instances and compare it with the heuristic method.

The fifth column is data collected by the final solution produced by the Heuristic Approach, named *Heuristic Approach*. This column has 4 sub-columns, which will be referred to as 5.1, 5.2, 5.3, and 5.4. Sub-column 5.1, named *OFV* is the objective function value produced by the Heuristic Approach. This is based on the customers visited by the solution, as seen in Sub-column 5.2 named *Customers Visited*. The computational time is recorded for the solutions to be produced, which can be seen in sub-column 5.3 named *CPU Time (seconds)*. The final 5.2 sub-column is named *Heuristic Applied*, showing if the solution was developed by the heuristic or a combination of the heuristic and the metaheuristic. This is because, for smaller and simpler instances, the use of Simulated Annealing is unnecessary and a feasible solution is produced just by the heuristic.

Column 6, named *Optimality Gap*, is the difference between the Objective Function Values between the solutions of the Base Mathematical Model and the Heuristic Approach. The formula for calculating the Optimality Gap is shown in equation 5.1.

$$(OFV_{BaseMathematicalModel} - OFV_{HeuristicApproach})/OFV_{BaseMathematicalModel} \tag{5.1}$$

The data represented in Table 5.5 in column 5 is an average across multiple runs of the model for each instance. This is because there are multiple random operators chosen in each Simulated Annealing iteration. Hence each run would produce a different answer. For the OFVs and customers visited (sub-columns 5.1 & 5.2), each of the instances was run 10 times, calculated the average of the results from the runs and rounded out. The data for the computation time (sub-column 5.3) for each instance was done the same but rounded out to the nearest hundredths decimal.

Instance	**Customers**	**Vehicles**	**Base Mathematical Model**			**Heuristics Approach**				**Optimality Gap**
No.	***N***	***V***	***OFV***	**Customers Visited**	***CPU Time (seconds)***	***OFV***	**Customers Visited**	***CPU Time (seconds)***	**Heuristc Applied**	
1	2	2	3	2	0.61	3	2	0.31	NNBH	0.00%
3	5	2	11	5	0.65	11	5	0.38	NNBH	0.00%
4	9	4	25	9	0.82	25	9	0.41	NNBH	0.00%
5	10	4	28	10	1.17	28	10	0.39	NNBH	0.00%
6	12	4	35	12	4.51	35	12	0.38	NNBH	0.00%
7	15	4	44	15	165.88	44	15	0.37	NNBH	0.00%
8	15	5	44	15	55.91	44	15	0.38	NNBH	0.00%
9	16	4	45	16	261.00	45	16	0.40	NNBH + SA	0.00%

Table 5.5: Small Instance testing data

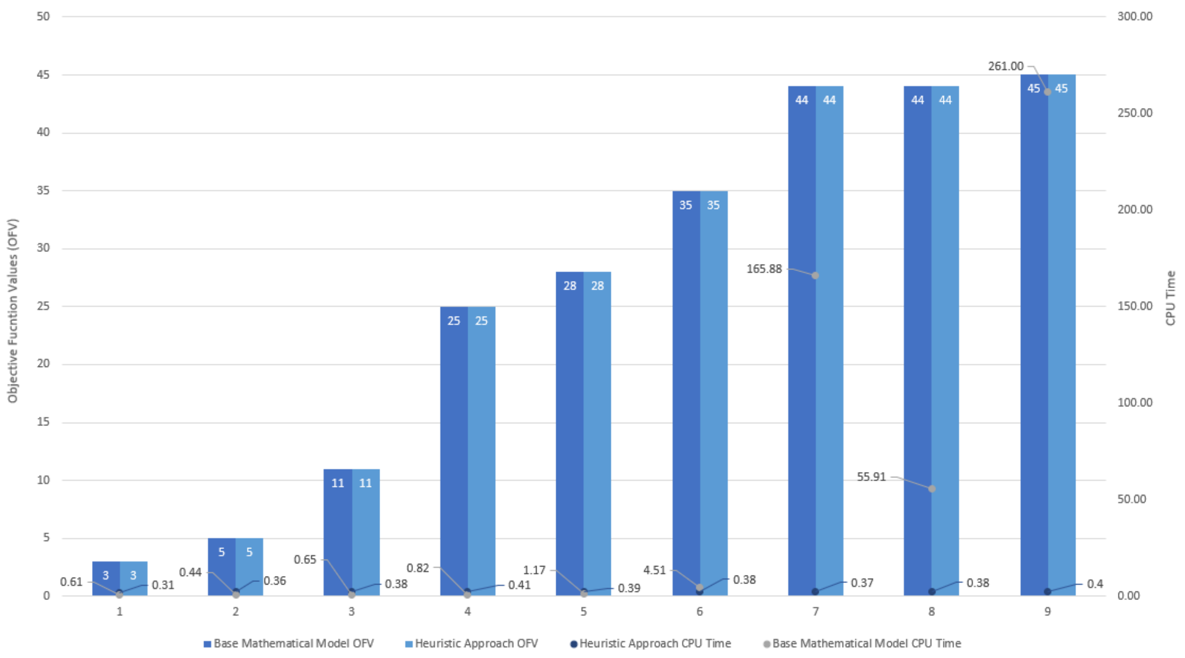

Figure 5.2: OFVs and CPU times of Base Mathematical Model and Heuristic Approach

As can be seen in Table 5.5 and Figure 5.2, there are many observations that can be made about the proposed model. Firstly, the proposed model can produce optimal solutions compared to the Base Mathematical Model. The optimality gap throughout the tests is all 0%. This indicates that the solutions generated by the heuristic approach exhibit a high degree of alignment with the desired outcomes, in relation to objective function values. Finally, the heuristic approach can produce solutions in less than half a second, with a mean computational time of 0.378 seconds. This shows how drastically quicker the heuristic approach can produce solutions as compared to the Base Mathematical Mode, which has a mean of 61.319 seconds. Combining the two observations, the heuristic approach produces reliable and high-quality solutions very efficiently.

Figures 5.3 and 5.4 show examples of how a Base Mathematical Model and Heuristic Approach solutions are produced. The solutions are for Testing Result 9 where 16 customers are delivered by 4 vehicles.

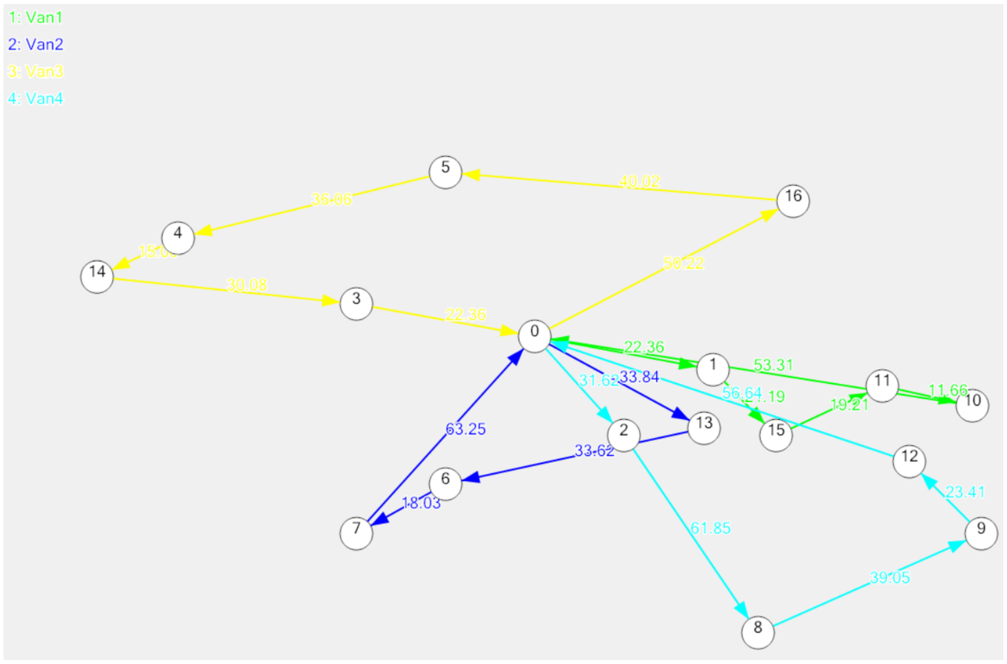

Figure 5.3: Small instance: Base Mathematical solution

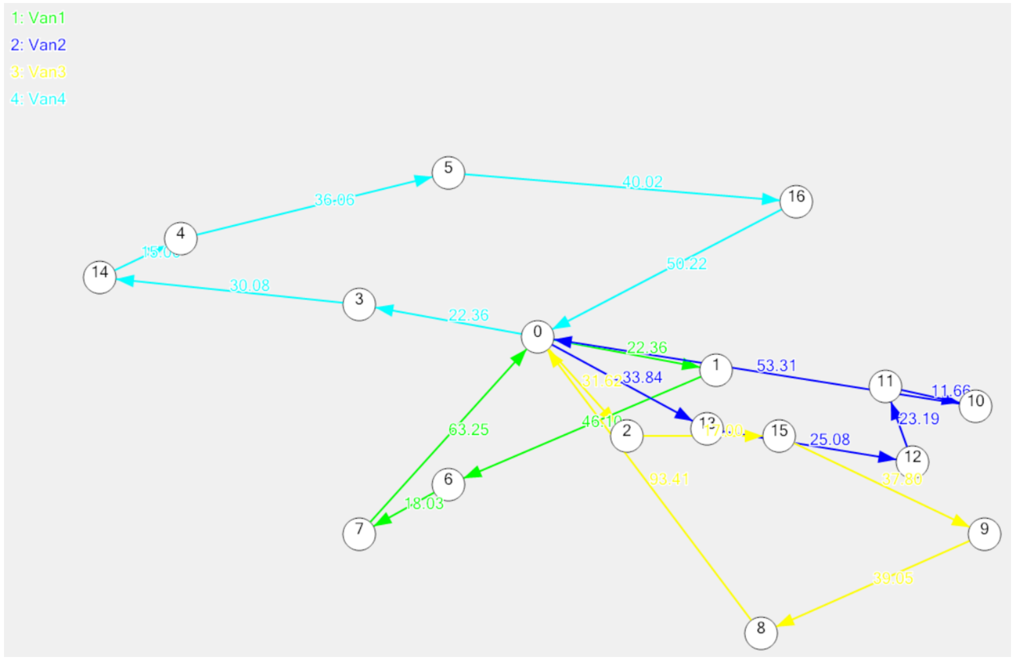

Figure 5.4: Small instance: Heuristic Approach solution

5.4 Large Instance Results

The benefits of the Simulated Annealing and its importance can be showcased by testing the model with larger instances of customers and vehicles. The testing data is seen in Table 5.6 below. Similar to Table 5.5, the first column is called the *Instance*, which shows the index of the testing instance done for the model. The second column is called *Customers*, which is the number of available customers that can be visited. The following column, named *Vehicles*, shows the number of vehicles in the fleet assigned with the energy budget. The value of the energy budget assigned for the fleet is seen in column 4, named *Energy Budget*.

The fifth column, named *NNBH Solution*, establishes the data collected from the Nearest Neighbour Based Heuristic. The fifth column has 4 sub-columns, which will be referred to as 5.1, 5.2, 5.3, and 5.4. Sub-column 5.1, named *OFV*, shows the Objective Function Value produced by the heuristic. Sub-column 5.2, called *Customers Visited*, shows how many customers are visited by the number of vehicles given for that instance. Sub-column 5.3, named *CPU Time (seconds)*, records the computational time taken for the heuristic to produce a final result. Column 5.4 is named *Distance Cost*, which displays the total distance covered for the iteration. This is to confirm that the distance of the solutions is within the energy budgets.

The sixth column, named *Metaheuristic Solution*, displays the data collected from the solutions produced by Variable Neighbourhood Search and Simulated Annealing metaheuristic. Similar to the previous column, this column also has 4 sub-columns, which will be referred to as 6.1, 6.2, 6.3, and 6.4. The representation of each sub-column is the same as the previous column but for the metaheuristic results. The seventh column, named *Combined CPU Time (seconds)*, shows the combined computational time of both the heuristic and metaheuristic working together. The final column, called *Final OFV*, shows the final OFV produced.

Similar to the data collected in Small Instances, the data represented in the table is an average across multiple runs of the model for each instance. Hence, the *OFVs* and *Customers Visited* (sub-column 5.1 & 6.1 and sub-column 5.2 & 6.2 respectively) are averaged out and rounded out. Meanwhile, the *CPU Time* and *Distance Cost* (sub-column 5.3 & 6.3 and sub-column 5.4 & 6.4 respectively) are averaged out and rounded to the nearest hundredths.

Many observations could be made in this table. The first observation from Table 5.6 is how the heuristic can quickly and consistently produce an initial solution, with a mean CPU time of 0.375 seconds. It also demonstrates how the metaheuristic produces a better result in

Instance	**Customers**	**Vehicles**	**Energy Budget**	**NNBH Solution**				**Metaheuristic Solution**				**Combined CPU Time (seconds)**	**Final OFV**
No.	***R***	***V***	***e***	**OFV**	***Customers Visited***	***CPU Time (seconds)***	**Distance Cost**	***OFV***	**Customers Visited**	**CPU Time (seconds)**	**Distance Cost**		
1	20	5	1000	61	20	0.43	806.55	61	20	0.09	877.73	0.52	61
2	30	5	1000	73	24	0.32	818.12	73	24	0.11	891.33	0.43	73
3	30	6	1000	73	24	0.41	818.12	74	25	0.08	968.76	0.49	74
4	30	6	2000	74	24	0.39	818.12	85	28	0.09	1005.14	0.48	85
5	40	7	2000	94	31	0.32	1132.51	96	32	0.55	1198.96	0.87	96
6	40	8	2000	100	33	0.34	1317.21	103	34	0.49	1301.5	0.83	103
7	50	9	2000	115	38	0.35	1209.23	118	39	0.6	1563.09	0.95	118
8	50	10	2000	115	38	0.43	1209.23	121	40	0.55	1586.52	0.98	121
9	60	11	2000	127	42	0.39	1321.18	136	46	0.52	1761.9	0.91	136
10	60	12	2000	127	42	0.4	1321.18	139	46	0.53	1908.62	0.93	139
11	60	15	3000	139	46	0.35	1694.29	149	50	0.57	2713.73	0.92	149

Table 5.6: Large instances testing data

rapid time, with an average CPU time of 0.38 seconds.

The second observation from the data set is that the metaheuristic can visit a higher prioritised and higher number of customers than the heuristic solution. This increases the objective function values of each instance, with an average increase of 5% in OFVs. This is with the help of the Swap, Insert, 1-opt and 2-opt operators in use with the VNS and SA metaheuristics.

Another key observation that can be made from the instances solved is the importance of energy budgeting in the problem. Focusing on Instance 2 & 3, the increase of vehicles from 5 to 6 does not affect the number of customers visited drastically. This is because the energy budget of 1000 restricts the possibility of visiting more customers, as the total distance cost for that instance is 968.76. However, in Testing Result 4, increasing the energy budget from 1000 to 2000 increases the OFV from 74 to 85.

Another observation that is made by the data set collected is the scalability of this model. This is shown in Testing Result 11, as the computational time for visiting 60 customers by 15 vehicles is less than a second. This establishes the applicability of this model in real-world scenarios.

Figures 5.5 and 5.6 show the visualisations of the data collected. Figure 5.5 compares the OFVs between the heuristic and metaheuristic solutions. Meanwhile, Figure 5.6 shows the influence of the energy budget on the final OFVs for each instance.

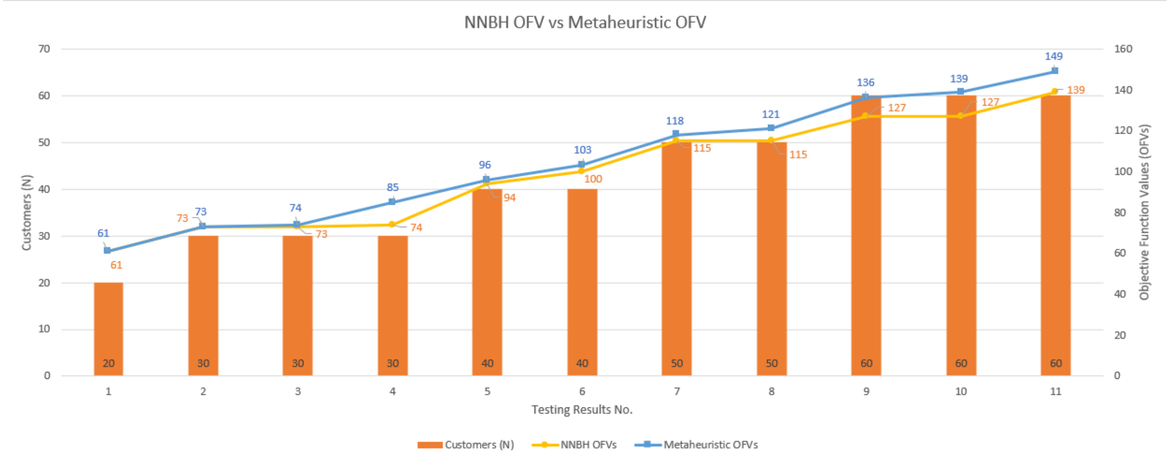

Figure 5.5: NNBH OFVs vs Metaheuristic OFVs

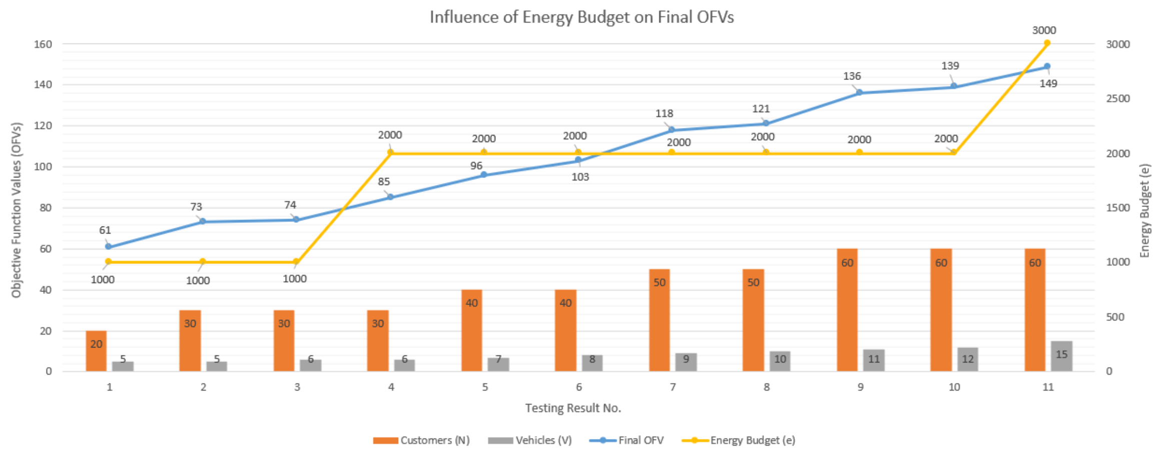

Figure 5.6: Influence of Energy Budget on Final OFVs

Figure 5.5 points out an interesting observation between the heuristic and metaheuristic OFVs. As the number of customers increases, the metaheuristic progressively improves in OFVs compared to the heuristic OFVs. This is because as the number of customers increases, the NNBH heuristic has more unvisited customers. This gives the metaheuristic more customers to swap with customers or insert into routes randomly, which has a better chance of yielding a higher objective function value. Figure 5.5 also shows the consistency of the solutions produced by the Nearest Neighbour Based Heuristic.

Figure 5.6 shows the influence that the energy budget has on the final OFV produced by the model. It is to demonstrate how even in the best-case scenarios, the energy budget constraints have an impact on the threshold of the OFV produced. As seen in Instances 2-4, the increase in the energy budget shows a gradual increase in OFV as mentioned earlier. This can also be seen between Instances 10 & 11. In Testing Result 11, the increase in the energy budget and the number of vehicles result in an increase in the OFV by 7.12%.

An example of a solution produced from the largest instance is seen in Figures 5.7 and 5.8. These examples are from a solution for 60 customers and 15 vehicles, which is the solution from Testing Result 11. Figure 5.7 shows the solution produced by the Nearest Neighbour Based Heuristic and Figure 5.8 shows the result of the Simulated Annealing metaheuristic. Figure 5.7 shows how each route is based on the nearest neighbour that the respective vehicle can travel to. Figure 5.8 shows how the operators move individual, sets of customers, or insert unvisited customers, from the routes defined previously by NNBH as they are shuffled around for a better solution.

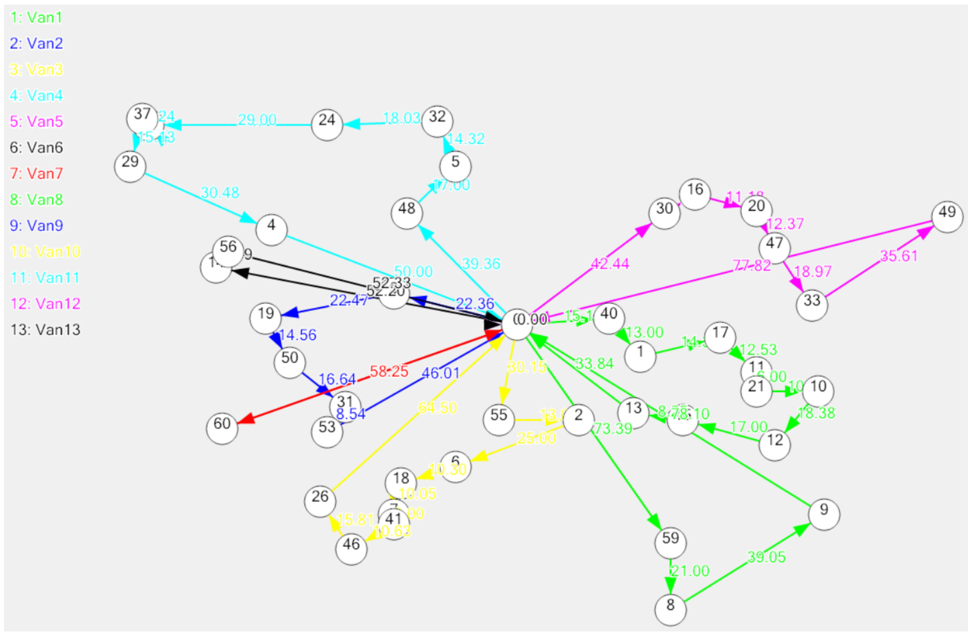

Figure 5.7: Large instance solution: NNBH

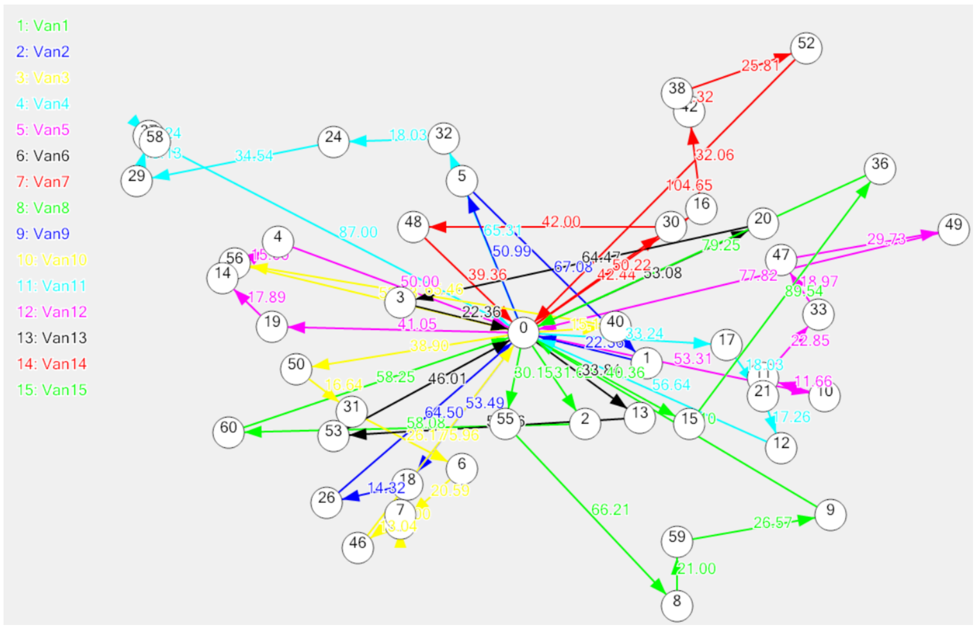

Figure 5.8: Large instance solution: SA & VNS

Chapter 6

Conclusions

6.1 Concluding Thoughts

This report demonstrates a unique variant of the traditional VRP model compiled as a hybrid of the Travelling Repairman Problem (TRP) and Electric Vehicle Routing Problem while incorporating an energy budget constraint on charging a fleet of vehicles. The objective function is to maximise the priority of the customers while being constrained by an energy budget for charging the fleet. The report proposes an algorithmic framework that uses a Nearest Neighbour Based Heuristic with a metaheuristic called Simulated Annealing and Variable Neighbourhood Search. A statistical test is conducted to compare the performances of the solution approach and the model verification. The testing phase initially uses the base mathematical model as a benchmark for smaller instances of solutions. For larger instances, computation times and objective function values are considered to demonstrate progress.

The computational results of both instances show the success of the proposed algorithmic framework. In smaller instances, the model was able to produce optimal solutions with an optimality gap of 0.00%. Furthermore, it can produce solutions within less than half a second, with a mean computational time of 0.378 seconds. For larger instances, the metaheuristic solutions produced have an increase of 5% in the OFVs compared to the initial solution produced by the heuristic. It also demonstrates the scalability prospect for large instances. Another speciality of this model is the metaheuristic progressively improves in OFVs compared to the heuristic as larger instances get larger. This is useful for real-world applications of this model.

The effectiveness of the model demonstrated by this paper will enable organisations that currently deploy, or intend to deploy, electric vehicle fleets in the commercial or logistics sector to incorporate this model for their successful use cases. The constraint considered by this, of the energy budget, is pressing a real-world case scenario as the demand for fleets of electric vans and trucks increases. By incorporating this model, fleet operators that face an increasing need for efficiency and priority maximisation of customers will be able to enhance their revenues by planning and scheduling efficient delivery of goods to their priority customers.

6.2 Future Work

The potential success of this model is promising based on the above data. However, there is always the need to improve the heuristic and metaheuristic approaches.

Many alternative heuristics can be implemented into this specific problem other than NNBH. The most famous heuristic for Vehicle Routing Problem is the Clarke and Wright Savings Algorithm (Rand, 2009). This heuristic is based on the idea of consolidating the routes of multiple vehicles to minimise the total cost. It starts by calculating the savings obtained by consolidating each pair of customers and then building routes by selecting pairs with the highest savings until all customers are assigned. In this implementation, the savings can be considered as the increase in the priority of customers in each route.

For the options of using metaheuristics, many can be added to or replaced with the current metaheuristic approach for this type of problem. One that can be added to this model is Tabu Search (Cordeau and Laporte, 2005). Tabu Search is a local search-based metaheuristic that uses a memory-based mechanism to avoid revisiting previously explored solutions. It iteratively explores the neighbourhood of the current solution and selects the best one that is not in the tabu list. Hence the solutions that are rejected by the Simulated Annealing approach can be added to the tabu list. This would result in finding an optimal solution at a quicker rate.

Another metaheuristic that can be added to the existing metaheuristic or be replaced is a Large Neighbourhood Search (Dumez et al., 2021). Large Neighbourhood Search is a heuristic approach that iteratively destroys and repairs a portion of the solution, called the neighbourhood, to find a better solution. Large Neighbourhood Search aims to focus on a small subset of the decision variables and apply a powerful optimisation algorithm, such as integer programming or local search, to find a better solution.

One more option that would also work as an additional or replacement metaheuristic is the Ant Colony Optimisation (Bell and McMullen, 2004). Ant Colony Optimisation is a metaheuristic optimisation technique inspired by the foraging behaviour of ants. Ant Colony Optimisation uses artificial ants to search for good solutions to optimisation problems, and the ants deposit pheromone trails as they move through the problem space. The pheromone levels on the edges of the solutions increase over time, guiding the search process towards these edges and increasing the probability of other ants choosing these edges in the future.

From a testing perspective, the use of an exact method approach would help benchmark the approach further. There are many implementations of exact method models for problems similar to this problem. Column Generation (Choi and Tcha, 2007) is a great example of an exact method approach. Column Generation is an optimisation technique used to solve large-scale optimisation problems. It is particularly useful for problems that involve a large number of variables or constraints, making them difficult to solve using traditional optimisation methods. This can be used to find an optimal solution for larger instances of this problem to understand the lower bound of the model.

A different approach to benchmark the model for larger instances is using Branch, Price and Cut algorithm (Archetti et al., 2015). The algorithm works by iteratively dividing the problem into smaller subproblems and solving them optimally using a pricing subproblem. The pricing subproblem generates new routes and assigns priority customers to them. The algorithm uses a cutting plane approach to eliminate sub-optimal solutions and strengthens the linear programming relaxation of the problem. The cuts are derived from violated constraints that are found during the solution process. This algorithm is particularly well-suited for problems with a large number of priority customers, as it can efficiently generate and evaluate a large number of routes. This approach can be used to find the lower bound of the problem.

In terms of how this project is utilised by Miralis Data, the project is aimed at being integrated with their product called Fuuse Fleets. The project helps the end-to-end fleet transformation solution have a succinct and comprehensive dashboard of what vehicles can be transformed in total. This project would help forecast the number of Electric Vehicles in their fleets based on daily customer requirements while transitioning gradually without impacting profitability from their daily revenues and deliveries. The project hopes to schedule the adequate number of transitioned vehicles required to service customers on any specific day based on customer needs.

Chapter 7

Appendix

EV	Electric Vehicle
HGV	Heavy Goods Vehicle
SVRP	Static Vehicle Routing Problem
VRP	Vehicle Routing Problem
NNBH	Nearest Neighbour Based Search
VNS	Variable Neighbourhood Search
SA	Simulated Annealing
CGE	Centre of Global Eco-Innovation

Table 1: Definitions of acronyms used in the Introduction

CVRP	Capacitated Vehicle Routing Problem
EVRP	Electric Vehicle Routing Problem
MDVRP	Multi-Depot Vehicle Routing Problem
NNBH	Nearest Neighbour Based Search
SA	Simulated Annealing
TRP	Travelling Repairman Problem
TSP	Travelling Salesman Problem
VRPTW	Vehicle Routing Problem with Time Windows
VRPB	Vehicle Routing Problem with Backhauls
VNS	Variable Neighbourhood Search

Table 2: Definitions of acronyms used in the Literature Review

References

Afifi, S., Dang, D., and Moukrim, A. (2013), “A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints”, *International Conference on Learning and Intelligent Optimization*, Springer, pp. 259–265, DOI: 10.1007/978-3-642-44973-4_27.

Afrati, F., Cosmadakis, S., Papadimitriou, C.H., Papageorgiou, G., and Papakostantinou, N. (1986), “The complexity of the travelling repairman problem”, *RAIRO - Theoretical Informatics and Applications* 20 (1), pp. 79–87, DOI: 10.1051/ita/1986200100791.

Anonymous (Apr. 2023), *Volta Trucks - Volta Zero Ambient Created specifically for urban distribution*, URL: https://voltatrucks.com/volta-zero-ambient.

Archetti, C., Bianchessi, N., and Speranza, M.G. (2015), “A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem”, *Computers & Operations Research* 64, pp. 1–10.

Bell, J.E. and McMullen, P.R. (2004), “Ant colony optimization techniques for the vehicle routing problem”, *Advanced engineering informatics* 18 (1), pp. 41–48.

Bräysy, O. (2003), “A reactive variable neighborhood search for the vehicle-routing problem with time windows”, *INFORMS Journal on Computing* 15 (4), pp. 347–368, DOI: 10.1287/ijoc.15.4.347.24896.

Choi, E. and Tcha, D.-W. (2007), “A column generation approach to the heterogeneous fleet vehicle routing problem”, *Computers & Operations Research* 34 (7), pp. 2080–2095.

Clegg, B. (2021), “Miralis Data”, *Impact* 2021 (1), pp. 35–39, DOI: 10.1080/2058802X.2021.1886768, URL: https://doi.org/10.1080/2058802X.2021.1886768.

Cordeau, J.-F. and Laporte, G. (2005), *Tabu search heuristics for the vehicle routing problem*, Springer.

Crevier, B., Cordeau, J.-F., and Laporte, G. (2007), "The multi-depot vehicle routing problem with inter-depot routes", *European Journal of Operational Research* 176 (2), pp. 756–773, ISSN: 0377-2217, DOI: `https://doi.org/10.1016/j.ejor.2005.08.015`, URL: `https://www.sciencedirect.com/science/article/pii/S0377221705006983`.

Crispim, J. and Brandão, J. (2005), "Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls", *Journal of the Operational Research Society* 56 (11), pp. 1296–1302, DOI: `10.1057/palgrave.jors.2601935`.

Dadds, K. (July 2021), "Domestic Road Freight Statistics, United Kingdom 2020", *Road Freight Statistics: 2020*, London, United Kingdom: Department for Transport of UK Government.

Dantzig, G.B. and Ramser, J.H. (1959), "The Truck Dispatching Problem", *Management Science* 6 (1), pp. 80–91, DOI: `10.1287/mnsc.6.1.80`, URL: `https://doi.org/10.1287/mnsc.6.1.80`.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992), "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows", *Operations Research* 40 (2), pp. 342–354, DOI: `10.1287/opre.40.2.342`, URL: `https://doi.org/10.1287/opre.40.2.342`.

Dumez, D., Lehuédé, F., and Péton, O. (2021), "A large neighborhood search approach to the vehicle routing problem with delivery options", *Transportation Research Part B: Methodological* 144, pp. 103–132.

Earl, T., Mathieu, L., Cornelis, S., Kenny, S., Ambel, C.C., and Nix, J. (Aug. 2018), "Analysis of long haul battery electric trucks in EU", URL: `https://www.transportenvironment.org/wp-content/uploads/2021/07/20180725_T&E_Battery_Electric_Trucks_EU_FINAL.pdf`.

Fleszar, K., Osman, I., and Hindi, K. (2009), "A variable neighbourhood search algorithm for the open vehicle routing problem", *European Journal of Operational Research* 195 (3), pp. 803–809, ISSN: 0377-2217, DOI: `10.1016/j.ejor.2007.06.064`, URL: `https://www.sciencedirect.com/science/article/pii/S0377221707011046`.

Fukasawa, R., Lysgaard, J., Poggi de Aragão, M., Reis, M., Uchoa, E., and Werneck, R.F. (2004), "Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem", *Integer Programming and Combinatorial Optimization*, ed. by D. Bienstock and G. Nemhauser, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–15, ISBN: 978-3-540-25960-2.

Ghasemalizadeh, O., Khaleghian, M., and Taheri, S. (Sept. 2016), "A Review of Optimization Techniques in Artificial Networks", *International Journal of Advanced Research* 4, pp. 1668–1686, DOI: `10.21474/IJAR01/1627`.

Halim, A.H. and Ismail, I. (2017), "Combinatorial Optimization: Comparison of Heuristic Algorithms in Travelling Salesman Problem", *Archives of Computational Methods in Engineering* 26 (2), pp. 367–380, DOI: `10.1007/s11831-017-9247-y`, URL: `https://doi.org/10.1007%5C%2Fs11831-017-9247-y`.

Hansen, P., Mladenović, N., and Moreno Pérez, J.A. (2008), "Variable Neighbourhood Search: Methods and Applications", *4OR* 6 (4), pp. 319–360, DOI: `10.1007/s10288-008-0089-1`.

Keskin, M. and Çatay, B. (2016), "Partial recharge strategies for the electric vehicle routing problem with time windows", *Transportation Research Part C: Emerging Technologies* 65, pp. 111–127, ISSN: 0968-090X, DOI: `https://doi.org/10.1016/j.trc.2016.01.013`, URL: `https://www.sciencedirect.com/science/article/pii/S0968090X16000322`.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. (1983), "Optimization by Simulated Annealing", *Science* 220 (4598), pp. 671–680, DOI: `10.1126/science.220.4598.671`, URL: `https://www.science.org/doi/abs/10.1126/science.220.4598.671`.

Kytöjoki, J., Nuortio, T., Bräysy, O., and Gendreau, M. (2007), "An efficient variable neighborhood search heuristic for very large scale vehicle routing problems", *Computers & Operations Research* 34 (9), pp. 2743–2757, ISSN: 0305-0548, DOI: `https://doi.`

org/10.1016/j.cor.2005.10.010, URL: https://www.sciencedirect.com/science/article/pii/S0305054805003394.

Laguna, M. and Martí, R. (2013), “Heuristics”, *Encyclopedia of Operations Research and Management Science*, ed. by S. Gass and M. Fu, Boston, MA: Springer US, pp. 695–703, ISBN: 978-1-4419-1153-7, DOI: 10.1007/978-1-4419-1153-7_1184.

Lenstra, J.K. and Rinnooy Kan, A.H.G. (1975), “Some Simple Applications of the Travelling Salesman Problem”, *Journal of the Operational Research Society* 26 (4), pp. 717–733, DOI: 10.1057/jors.1975.151, URL: https://doi.org/10.1057/jors.1975.151.

Lin, J., Zhou, W., and Wolfson, O. (2016), “Electric Vehicle Routing Problem”, *Transportation Research Procedia* 12, Tenth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain, pp. 508–521, ISSN: 2352-1465, DOI: https://doi.org/10.1016/j.trpro.2016.02.007, URL: https://www.sciencedirect.com/science/article/pii/S2352146516000089.

Mohammed, M., Ahmad, M., and Mostafa, S. (2012), “Using Genetic Algorithm in implementing Capacitated Vehicle Routing Problem”, *2012 International Conference on Computer & Information Science (ICCIS)*, vol. 1, pp. 257–262, DOI: 10.1109/ICCISci.2012.6297250.

Mohammed, M.A., Ghani, M.K.A., Hamed, R.I., Mostafa, S.A., Ibrahim, D.A., Jameel, H.K., and Alallah, A.H. (2017), “Solving vehicle routing problem by using improved K-nearest neighbor algorithm for best solution”, *Journal of Computational Science* 21, pp. 232–240, ISSN: 1877-7503, DOI: https://doi.org/10.1016/j.jocs.2017.04.012, URL: https://www.sciencedirect.com/science/article/pii/S187775031730426X.

Muritiba, A.E.F., Bonates, T.O., Da Silva, S.O., and Iori, M. (2021), “Branch-and-Cut and Iterated Local Search for the Weighted k-Traveling Repairman Problem: An Application to the Maintenance of Speed Cameras”, *Transportation Science* 55 (1), pp. 139–159, DOI: 10.1287/trsc.2020.1005, URL: https://doi.org/10.1287/trsc.2020.1005.

Polacek, M., Hartl, R., Doerner, K., and Reimann, M. (2004), “A variable neighborhood search for the multi depot vehicle routing problem with time windows”, *Journal of heuristics* 10 (6), pp. 613–627, DOI: 10.1007/s10732-005-5432-5.

Rand, G.K. (2009), "The Life and Times of the savings method for vehicle routing problems", *ORiON* 25 (2), pp. 125–145, DOI: `10.5784/25-2-78`.

Skidmore, C. (Oct. 2021), "Net Zero Strategy: Build Back Greener", *Net Zero Strategy: Build Back Greener*, London, United Kingdom: Department for Business, Energy & Industrial Strategy of UK Government, ISBN: 978-1-5286-2938-6.

Tavakkoli-Moghaddam, R., Safaei, N., and Gholipour, Y. (2006), "A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length", *Applied Mathematics and Computation* 176 (2), pp. 445–454, ISSN: 0096-3003, DOI: `https://doi.org/10.1016/j.amc.2005.09.040`, URL: `https://www.sciencedirect.com/science/article/pii/S0096300305008210`.

Toth, P. and Vigo, D. (2002), "VRP with Backhauls", *The Vehicle Routing Problem*, pp. 195–224, DOI: `10.1137/1.9780898718515.ch8`, URL: `https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.ch8`.

Waite, C. (Feb. 2021), "2019 UK Greenhouse Gas Emissions", *Final UK greenhouse gas emissions national statistics: 1990 to 2019*, London, United Kingdom: Department for Business, Energy & Industrial Strategy of UK Government, URL: `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/957887/2019_Final_greenhouse_gas_emissions_statistical_release.pdf`.

Xie, S. (Dec. 2019), "Extrema: Maxima & Minima - Calculus Basics - Medium", URL: `https://medium.com/self-study-calculus/extrema-maxima-minima-4e2e6956a44a`.