A decision support tool for vehicle routing in the retail sector aimed at improving driver-route familiarity

by

Marlize Helene de Villiers

Final year project presented in partial fulfilment of the requirements for the degree of
Bachelor of Engineering (Industrial Engineering)
in the Faculty of Engineering at Stellenbosch University

Supervisor: Mr JCP King
Co-supervisor: Prof JH van Vuuren

December 2022

Declaration

By submitting this project electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 1, 2022

Abstract

A typical activity in the supply chain of retail organisations that is not visible to the consumers, but essential for ensuring the availability of a variety of products at a competitive price, is called vehicle routing. This activity consists of the transportation of commodities from a distribution center, often also referred to as a depot, to stores using a fleet of delivery vehicles. The industry partner attached to this project often encounters challenges when it comes to the practical implementation of planned delivery routes, which cannot be addressed efficiently by standard and commercially available routing software. Unfavourable traffic conditions, unanticipated road works, and drivers who travel on roads that are not suited for the operated delivery vehicles are some of the problems reported by the industry partner. These challenges often lead to an increase in travel times and a subequent degradation in the operational efficiency with which deliveries are performed. A possible solution aimed at improving the practical implementation of planned delivery routes is to increase driver-route familiarity.

The goal of this project is to design and implement computerised decision support to provide high-quality routing solutions for retail organisations, which attempts to improve the practical implementation of planned delivery routes by increasing driver-route familiarity (allowing drivers to travel on routes with which they are familiar). In pursuit of this goal, a multi-attribute vehicle routing problem for increased driver-route familiarity is derived in the form of a mixed-integer programming problem. The model is concerned with computing a set of high-quality routing solutions with the objective of minimising transportation cost.

The proposed model accounts for common operational constraints encountered by retail organisations, such as the limited capacity of delivery vehicles, time-windows associated with customers, split deliveries, and a heterogenous fleet of delivery vehicles that are available to service customers. The model is implemented in IBM ILOG's CPLEX Optimisation Studio 20.1.0, a cutting-edge software suite that solves mixed-integer programming problems exactly by employing the branch-and-cut method. The implementation of the model is verified and validated according to a well-researched strategy to ensure the reliability and credibility of the model. The model and its implementation is embedded within a user-friendly decision support tool and applied to a case study involving real-world data to demonstrate its practical applicability.

Graduate Attributes Reference

Attribute	Reference	
	Section	**Page**
1. Problem solving: Demonstrate competence to identify, assess, formulate and solve convergent and divergent engineering problems creatively and innovatively.	*All*	*All*
5. Engineering methods, skills and tools, including information technology: Demonstrate competence to use appropriate engineering methods, skills and tools, including those based on information technology.	*2,3,4 & 5*	*3–57*
6. Professional and technical communication: Demonstrate competence to communicate effectively, both orally and in writing, with engineering audiences and the community at large.	*All*	*All*
9. Independent learning ability: Demonstrate competence to engage in independent learning through well developed learning skills.	*2,3,4 & 5*	*3–57*
10. Engineering professionalism: Demonstrate critical awareness of the need to act professionally and ethically and to exercise judgement and take responsibility within own limits of competence.	*All*	*All*

Acknowledgements

The author wishes to acknowledge the following people and institutions for their various contributions towards the completion of this work:

- First and foremost, I give honor and glory to our Lord and Saviour, Jesus Christ, for the abilities and abundance He has bestowed upon me. I thank Him for blessing me with perserverence and for directing my steps throughout my undergraduate degree.
- My parents, Hendrik and Lorette de Villiers, for their unwavering love and emotional support whenever I felt overwhelmed, as well as my sister, Estelle de Villiers, for her encouragement and assistance with daily responsibilities when my workload was overwhelming.
- My fiancé and best friend, Gideon Visser, for his continuous encouragement and prayers, and for teaching me how to rest well. Thank you for patiently listening to my monologues about my project and providing me with coffee whenever I needed it.
- My supervisor, Jacobus King, for your time, patience and willing devotion toward ensuring the successful completion of this project. I thank you for the passion you shared with me and for believing in my abilities whenever I doubted myself.
- My brilliant co-supervisor, Prof JH van Vuuren, and the exceptional research group he leads, the *Stellenbosch Unit for Operations Research in Engineering* (SUnORE), for their expertise, techinical support, academic disccusions, and for stirring my interest in the Operations Research field.
- My friends, Anri Brink, Lisa Bruwer, and Gerdo Breytenbach, for your support throughout our undergraduate degree and the COVID-19 pandemic, for all the experiences inside and outside of the faculty, the lunch-time conversations, and for all the laughter we shared together.

Table of Contents

List of Acronyms

CPLEX: IBM ILOG CPLEX Optimization Studio 20.1.0

CVRP: Capacitated vehicle routing problem

DST: Decision support tool

FSM-VRP: Fleet size and mix vehicle routing problem

GUI: Graphical user interface

HFVRP: Heterogeneous fixed fleet vehicle routing problem

IP: Integer programming

LIFO: Last-in-first-out

LP: Linear programming

MIP: Mixed-integer programming

MAVRP: Multi-attribute vehicle routing problem

PVRP: Periodic vehicle routing problem

SDVRP: Split delivery vehicle routing problem

TSP: Travelling salesman problem

V&V: Verification and validation

VRP: Vehicle routing problem

VRPTW: Vehicle routing problem with time-windows

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Contents

1.1 Background

A supply chain may be defined as a system of organisations, people, activities, information, and resources involved in fulfilling a customer request [11]. A supply chain encapsulates all activities involved to deliver goods or services to the consumer, including, but not limited to, new product development, marketing operations, distribution, finance, and customer service. The supply chain of retail organisations usually consists of multiple of these activities, including sales, distribution, and finance [5]. The effectiveness and success of a retail organisation depends, to a large extent, on the organisation's ability to effectively manage and reduce cost across all activities forming part of its supply chain [56]. It is therefore important to maintain efficiency within each activity constituting the supply chain, in order to remain competitive in the emerging retail market conditions. Retail organisations are required to satisfy the demand of consumers who value high-quality products, prefer a wide assortment of these products, as well as demand timeous delivery thereof — all at a competitive price.

A key element in the supply chain of any retail organisation is its transportation system, which acts as the link between, and often the backbone of many other supply chain activities. It is therefore crucial for retail organisations to focus their attention on developing and improving transportation systems in order to lower transportation cost. In the current emerging competitive business landscape, it is required for organisations to address changes in the market and customer requirements promptly and properly. The competitive environment of the retail sector in South Africa demands that organisations exhibit thorough planning and delivery routing to minimise transportation time.

Naudé and Mathee [42] describe transport cost, in a broad sense, as the costs involved with the movement of people and goods. Transportation cost occupies one-third of the total logistics costs of an organisation and is largely influenced by the formulation of transportation systems used in practice [60]. Transportation systems allows the moveability of goods and products and provides timely efficacy. A well developed transportation system could provide improved efficiency in logistics, reduce operational cost, and increase the quality of service quality [60]. The transportation cost associated with retail organisations in South Africa are significantly higher compared to other regions of the world [42]. An important factor contributing to this increase in transportation cost compared to other countries is the travel distance over which commodities have to be transported. A study performed by Martínez-Zarzoso *et al.* [37] reported that a 1% increase in travel distance translates to an approximate 0.25% increase in transportation cost. This implication emphasises the importance of transportation systems within the supply chain of retail organisations.

South African retailers may experience significant cost-savings by improving on their routing solutions. A typical activity in the supply chain of retail organisations that is not visible to the consumers, but essential in ensuring variety and availability of products at a competitive price, is called vehicle routing. This activity consist of the storage of products in a distribution centre, often referred to as a depot in the literature, and the transportation thereof from the depot to stores using a fleet of delivery vehicles that travels along a transportation network, as illustrated graphically in Figure 1.1. Each delivery vehicle must be presented with a schedule specifying the sequence of customers to be visited, and the commodities to be delivered to each store.

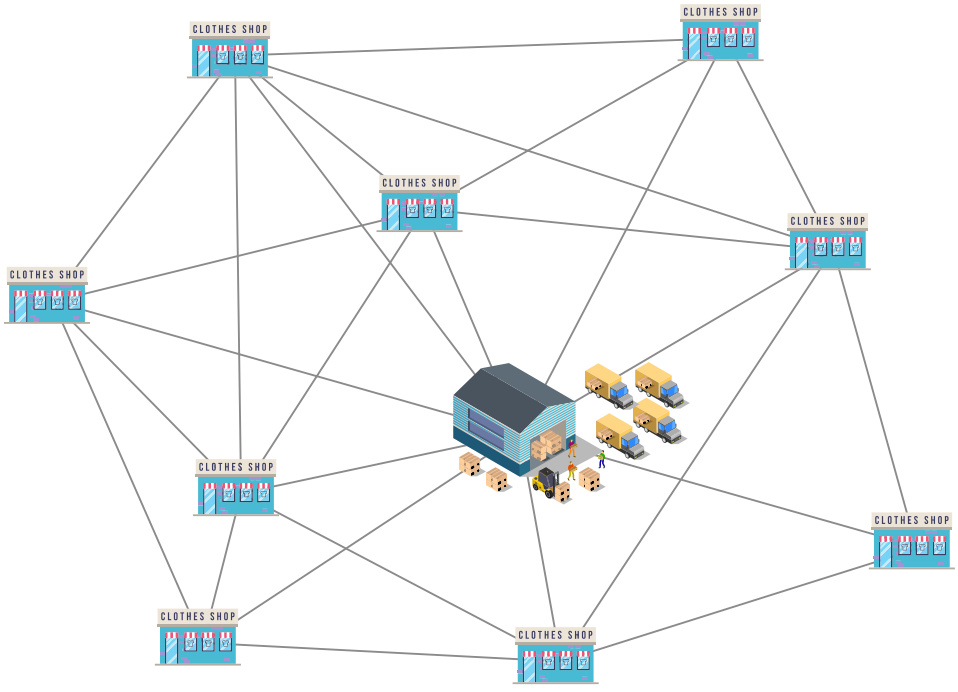


FIGURE 1.1: *A graphical illustration of the vehicle routing activity in the supply chain of a retail organisation.*

The *vehicle routing problem* (VRP) is one of the most studied and important combinatorial optimisation problems in the literature. The VRP is used to formulate the problem of distributing goods between depots and customers and it is employed within the operational decision making process of a supply chain. In a retail distribution system, products are delivered from a depot to multiple retail stores which are scattered geographically. The routing requirements of retail

organisations may therefore be modelled as a VRP to provide computerised decision support for route planning. The use of computerised vehicle routing and scheduling decision support has resulted in cost savings ranging from 5% to 20% of the cost associated with distribution planning activities in the supply chain [58]. Dantzig and Ramser [16] were the first to introduce the VRP more than 60 years ago, which led to major developments in variations on the basic VRP, as well as exact and approximate solution methodologies for solving VRP instances. The classical VRP is concerned with determining delivery routes from a depot to a set of customers at minimal cost [34]. A solution to the classical VRP is a set of routes in which each delivery vehicle starts and ends at the depot, and the demand of each customer is satisfied by exactly one delivery vehicle, while minimising the total distance travelled by all delivery vehicles [19]. The problem is central to distribution management, however, the growing cardinality and underlying complexity of modern day VRP instances render it impossible for modern day route planners to compute optimal solutions to VRP instances by hand.

The nature of the transported commodities, service level requirements, the characteristics of customers, and the fleet of delivery vehicles stationed at the depot are a few of the factors influencing the VRP formulation used to model the routing requirements of retail organisations [33]. Many variants exist on the basic notion of a VRP, each motivated by real-world applications. Furthermore, VRP variants may differ based on their objective functions. These objectives are often to either minimise the total distance travelled by all delivery vehicles, to minimise the travel time of all delivery vehicles, or to minimise the total transportation cost. Retail organisations, however, require a more complex VRP formulation than that of the basic VRP, in order to account for additional operational constraints.

A widely researched and employed VRP variant is the *capacitated VRP* (CVRP). In the CVRP, each customer exhibits a certain demand to be satisfied by a single delivery vehicle, and a homogeneous fleet of delivery vehicles with a limited capacity are available to service customers. Another popular VRP variant is the *VRP with time windows* (VRPTW) in which the service at each customer may only start within a specified time interval, known as a time-window. The VRPTW is often applied in the retail sector, due to stores being able to specify time intervals during which they prefer to receive deliveries. Another VRP variant is called the *split delivery VRP* (SDVRP), in which the demand of each customer may be satisfied by more than one delivery vehicle. Other practical variants make provision for a heterogeneous fleet of delivery vehicles, where different insurance, maintenance, and operating costs may be associated with each distinct delivery vehicle [55]. This practical consideration gives rise to the *Heterogeneous Fixed Fleet VRP* (HFVRP). Other popular variants that resulted from real-world problems include the multiple depot VRP, the periodic VRP, the stochastic VRP, the VRP with backhauls, and the VRP with pickup and delivering [19]. When multiple VRP attributes are combined into a single VRP formulation, it is called a *multi-attribute VRP* (MAVRP). MAVRPs are typically used in specific applications and may include any of the above mentioned variants, as well as others. The complexity of solving MAVRP instances increase significantly due to the combined complexities of the underlying VRP extensions.

The industry partner attached to this project is a large South African clothing retailer that seeks to improve the practical implementation of the routes stemming from solving their VRP instances, in order to increase the efficiency of activities in their supply chain. The industry partner has reported numerous challenges when attempting to implement the solutions provided by existing standard and commercial software when attempting solve their VRP instances. Many real-world problems, such as unanticipated traffic conditions and drivers travelling on roads that are not suitable for the operated delivery vehicles, result in degraded operational efficiency and increased transportation costs. These challenges may then result in a delivery vehicle driver

missing the time-window of a customer and may ultimately render the rest of the planned route infeasible. A possible solution aimed at improving the practical implementations of planned delivery routes is to improve driver-route familiarity [54]. Intini *et al.* [27] investigated the influence of route familiarity on driver behaviour. Driver-route familiarity does not only influence the road and traffic behavioural-based safety aspects, but also the travel time of drivers. A study regarding the influence of memory on driving behaviour found that an increase in the number of times a driver travels a certain route, ultimately results in a decrease in travel time [13]. Delivery vehicle drivers that are familiar with the routes on which they travel, will therefore be able to better anticipate unplanned external events and avoid driving on routes that are not suitable for the operated delivery vehicles.

A proposed method for creating driver-route familiarity proposed by King *et al.* [30], is to compute a set of standard routes, called *master routes*, for a given depot and its assigned customers. The master routes are a set of routes visiting each customer a specified number of times along different approaches. The master routes may then be used as blueprints when computing actual delivery routes. If actual delivery routes are not too dissimilar from the master routes, delivery vehicle drivers will be granted the opportunity to become familiar with travelling on theses master routes. By travelling on routes with which they are familiar with, delivery vehicle drivers will be able to increase the efficiency with which they perform deliveries. Furthermore, by continuing to drive on the master routes, the driver-route familiarity of delivery vehicle drivers with the master routes will increase, further increasing the efficiency with which they may perform deliveries. The objective of creating driver-route familiarity, however, may come at a cost. The total travel distance of delivery vehicles may increase when trying to compute actual delivery routes that are not too dissimilar from the master routes. Creating driver-route familiarity may therefore lead to additional transportation cost, however, the increase in efficiency across all activities in the supply chain resulting from the improved implementation of planned delivery routes are expected to result in an overall decrease in cost across these activities. The extent to which a retail organisation enforces driver-route familiarity is unique to each organisation and its supply chain. An organisation should therefore be able to specify a threshold, the familiarity percentage, which represents the extent to which they would like to enforce driver-route familiarity. Delivery routes will be assigned to drivers in such a way that the portion of the total distance travelled on the master routes by all delivery vehicles are a minimum of the familiarity threshold, while minimising the total transportation cost. Furthermore, the logistical operations of organisations should be taken into account by considering multiple VRP attributes, such as time-windows, split deliveries, and a heterogeneous fleet. Such an MAVRP which includes driver-route familiarity may exhibit numerous advantages that may lead to a more efficient implementation of routing solutions, ultimately guiding cost savings across the entire supply chain.

1.2 Problem statement

The aim in this project is to design and implement a computerised *decision support tool* (DST) for improved delivery routing, which allows for driver-route familiarity to be created in solutions, based on the preference of a user. The working of the DST is to be based on a combinatorial optimisation model, in the form of a *mixed-integer programming* (MIP) problem, which takes the following as input:

- the travel times and travel distances between all customers and the depot,
- a set of master routes with which delivery vehicle drivers are assumed to be familiar,

- a familiarity threshold specified by the user which governs the minimum percentage of distance to be travelled on these master routes,
- the time-window start and end times, and the demand volume associated with each customer, and
- information associated with the available fleet of delivery vehicles stationed at the depot.

The working of the DST will aid the user in making decisions regarding the delivery routes and schedules to be assigned to delivery vehicle drivers by producing as output a set of high-quality delivery routing solutions, capable of creating driver-route familiarity, whilst adhering to all operational constraints and minimising transportation cost.

1.3 Project objectives

The following objectives are pursued in this project:

I To *conduct* a thorough study of the literature related to:
   (a) the characteristics of VRPs in general,
   (b) the formulation of VRPs in the form of MIP problems,
   (c) exact methods for solving MIP problems,
   (d) methods and guidelines for the design, verification, and validation of computerised mathematical models.

II To *derive*, based on the guidelines of the literature reviewed in Objectives I(a)–(b), an MAVRP formulation in the form of an MIP problem, in support of high-quality delivery routing solutions capable of creating driver-route familiarity in solutions. The model takes as input a set of master routes with which delivery vehicle drivers are assumed to be familiar with, a specified familiarity threshold, the travel times and travel distances between customers and the depot, the time-windows and demands specified by each store, and information regarding the available fleet of delivery vehicles stationed at the depot. The model produces as output a set of delivery routes that creates driver-route familiarity, and adheres to all operational constraints, whilst minimising the total transportation cost.

III To *implement and verify* an exact solution methodology for problem instances of the model formulated in Objective II, based on the knowledge gained in pursuit of Objective I(c).

IV To *design* a user-friendly conceptual DST that is capable of computing delivery routes and ensuring driver-route familiarity according to the guidelines researched in pursuit of Objective I(b)–(c). The working of the DST should be based on the MIP model and corresponding exact solution methodology of Objectives II and III.

V To *verify and validate* the DST of Objective IV according to the guidelines researched in pursuit of Objective I(d) and generally accepted modelling guidelines.

VI To *conduct* a case study based on real-world data obtained from the industry partner, by applying the DST of Objective IV to these data.

VII To *evaluate* the ability of the concept demonstrator of Objectives II and its proposed solution approache of Objective III to compute high-quality delivery routes that creates driver-route familiarity.

VIII To *recommend* sensible follow-up work related to the work in this project which may be pursued in the future.

1.4 Project scope

Models based on real-world scenarios have a complex set of rules and constraints that govern and direct the model. Due to numerous factors that could potentially influence the successful implementation of such models, the following simplifying and unifying assumptions are made:

Heterogenoues vehicle fleet. A heterogeneous fleet of delivery vehicles is stationed at the depot and available to service customers. Each delivery vehicle may differ in terms of its capacity, fixed cost, and travel times and travel costs between locations.

Vehicle capacity. When constructing solutions, the maximum volumetric loading capabilities of delivery vehicles are considered.

A single depot. Multiple depots may be considered at once, however, this project considers depots individually.

Expected travel times and distances. The travel times and distances between vertices in the transportation network are assumed to be pre-determined and are based on average values. The model assists users to plan routes and schedules in advance and dynamic traffic conditions are therefore not taken into account.

Master routes. A set of predetermined master routes, with which delivery vehicle drivers are assumed to be familiar with, is provided as input to the MIP model and the DST. The calculation of these routes do not form part of this project, but are based on methods proposed by other researchers.

Familiarity threshold. A familiarity threshold, as a percentage, may be specified by the user and represents the extent to which driver-route familiarity are enforced in solutions.

Time-windows. Customers may specify time-windows during which service may take place. Vehicles are allowed to arrive before the start of a time-window, but must wait until the onset of the time-window before starting service at a customer.

Split deliveries. The demand volumes exhibited by customers may be satisfied by more than one delivery vehicle.

Unload time. The service time at a customer is dependent on the volume of goods that must be delivered to that customer.

Exact methods. Only exact solution methodologies will be considered for solving MIP problems.

1.5 Research methodology

A description of the methodology adopted in pursuit of the objectives listed in §1.3 is presented in this section. The research of this project is carried out in five distinct phases.

The first phase entails the execution of a literature review in pursuit of Objective I. A fundamental understanding of the characteristics of the basic VRP and the VRP variants applicable to

the work done in this project is obtained in fulfilment of Objective I(a). Furthermore, different existing attributes and variations of VRPs are explored to obtain a better understanding of how these models are formulated and how an MAVRP is derived. This includes conducting research on the CVRP, the VRPTW, the SDVRP, and the HFVRP because of the applicability of these variants to this project. Next, VRP formulations in the form of MIP problems are researched in pursuit of Objective I(b). There have been a vast number of attempts to formulate solution methodologies for solving VRP instances in the literature. A fundamental understanding of the most prolific and widely used exact methods for solving MIP problems are therefore studied, in fulfilment of Objective I(c). A study on the guidelines and methods for designing, verifying, and validating a DST follows in pursuit of Objective I(d), and in support of Objectives IV and V. The first phase furthermore includes the acquisition of technical skills required for the successful completion of this project, such as proficiency in IBM ILOG's *CPLEX Optimisation Studio* 20.1.0 (CPLEX) accessed *via* its Python programming language interface.

The second phase of the research is carried out in pursuit of Objective II and is based on the guidelines of the literature reviewed in the previous phase. A mathematical formulation for the MAVRP proposed in this project, aimed at creating driver-route familiarity in solutions, is derived in fulfilment of Objective II. Instances of the model is to be solved using an exact solution methodology in pursuit of Objective III.

During the third phase of the research, a computerised mathematical model is implemented to verify and validate the MAVRP in fulfilment of Objective III, based on the guidelines researched in fulfilment of Objective I(d). The third stage is iterative in nature, since the verification and validation of the computerised mathematical model provide feedback for further refinement resulting in a new and improved implementation. Furthermore, this phase also includes the development of a user-friendly DST, capable of computing and solving real-world problem instances specified by a user, in fulfilment of Objective IV. A generic verification and validation procedure, in line with the general practices and recommendations studied in fulfilment of Objective I(d), is employed to ensure the credibility and reliability of the DST in pursuit of Objective V.

The fourth phase of the research carried out in this project is devoted to applying the DST of Objective IV to a real-world case study to demonstrate and validate its practical application in fulfilment of Objectives VI and VII. The case study is used to evaluate real-world data and deliver numerical results in the form of delivery routes assigned to drivers. The effectiveness of the DST proposed in this report is critically assessed in terms of its ability to identify a set of high-quality delivery routes while ensuring driver-route familiarity.

The final phase exhibits recommendations made in pursuit of Objective VIII, by proposing sensible follow-up work related to the work in this project. A summary of the work documented in this project is provided, as well as suggestions for future work.

1.6 Project timeline

A Gantt chart illustrating the timeline followed in this project is provided in Figure A.1 of Appendix A. This timeline indicates the amount of time spent on each phase of the project.

1.7 Report organisation

The remainder of this report is partitioned into four parts, and are organised as follows. A review of the literature, in pursuit of Objective I, is documented in Chapter 2. The guidelines

and research reviewed in Chapter 2 serve as a basis for the remainder of this project and as guidelines for the design, development and implementation of the proposed MIP model. Chapter 3 follows with the derivation of an MAVRP mathematical model in the form of a MIP problem, capable of creating driver-route familiarity, as specified in Objectives II and III. Problem instances are solved by invoking CPLEX, which is an exact solution approach based on the branch-and-cut method. A number of test problem instances are solved to verify and validate the computerised mathematical implementation of the model in Chapter 3, in pursuit of Objective IV. Thereafter, as a final validation activity and in pursuit of Objectives V–VII, a DST is designed and executed using real-world data obtained from the industry partner attached to this project in Chapter 4. A final reflection and conclusion is delivered in Chapter 5. The report closes with a brief summary of the work done in this project, a critical evaluation of the contributions made by the project, and a discussion on what the author has learnt during the execution of this project. Sensible recommendations and follow-up work related to the research conducted in this project is also described in this chapter.

CHAPTER 2

Literature Review

Contents

In this chapter, a literature review is conducted in pursuit of Objective I. The literature review comprises three sections. The first part is presented in §2.1 and contains an in-depth discussion on the classical VRP and other VRP variants related to the work done in this project. Thereafter, exact solution methodologies for solving MIP problems are discussed in §2.2. A review of the verification and validation of linear programming problems and the computerised implementation thereof is conducted in §2.3. The chapter closes with a condensed summary of its contents in §2.4.

2.1 VRP formulations

In this section, exact formulations of the CVRP, the VRPTW, the HFVRP, and the SDVRP in the form of MIP problems are discussed in §2.1.1, §2.1.2, §2.1.3, and §2.1.4, respectively. The basic VRP is a generalisation of the *travelling salesman problem* (TSP). The TSP is concerned with determining the shortest route to visit a set of locations exactly once and then return to the starting location, whereas the basic VRP is concerned with determining a set of routes to perform transportation requests with a given fleet of delivery vehicles, at a minimum cost [35]. The VRP and its associated variants have grown in popularity, not only due to their wide range of real-world applications and practical relevance, but also due to the notorious difficulty associated with solving these combinatorial optimisation problems [59]. Variants of the VRP

also appear frequently in real-world applications that are not directly related to the delivery of goods, but some other transportation request, for example, the collection of mail from mailboxes and the scheduling of bus routes [12].

2.1.1 The CVRP

The most studied variant of the VRP, having primarily academic relevance, is the CVRP [59]. In the CVRP identical delivery vehicles are available to service customers, resulting in an increase in complexity compared to that of a TSP instance containing the same number of vertices [35]. Furthermore, each delivery vehicle is stationed at a single depot from which it departs to service customers, after which it returns to the depot. The increased complexity of the CVRP when compared to that of the TSP is not only due to the requirement of assigning customers to delivery vehicles, but also due to the need of proper sequencing of the customers visited along the route of each delivery vehicle.

The CVRP serves as a basis for understanding and describing other VRP variants, which in many instances is an extension of the CVRP. A MIP formulation of the CVRP proposed by van Vuuren [62] is described and verified by conducting small example instance in the remainder of this section. Let $\mathcal{C} = \{1, \ldots, n\}$ index the set of customers to be serviced. The depot is included in the additional supplemented set of vertices, $\mathcal{V} = \mathcal{C} \cup \{0, n+1\}$, where 0 denotes the depot upon departure and $n+1$ denotes the depot upon return. Let q_i denote the volume or weight of commodities that must be delivered to customer $i \in \mathcal{C}$, called the customer's *demand*. Each customer is assumed to exhibit a demand $q_i \geq 0$. Furthermore, let the set $\mathcal{A}$ denote the set of directed road network links, called arcs, along which delivery vehicles may travel. Let $\mathcal{G}(\mathcal{V}, \mathcal{A})$ represent the directed travel graph on which the CVRP is defined, with $\mathcal{V}$ as the vertex set and $\mathcal{A}$ as the arc set. Furthermore, let $\delta^-(i)$ denote the subset of vertices from which arcs are directed towards vertex $i \in \mathcal{V} \setminus \{0\}$, and let $\delta^+(i)$ denote the subset of vertices to which arcs are directed, from vertex $i \in \mathcal{V} \setminus \{n+1\}$ in the travel graph $\mathcal{G}$. Let the set $\mathcal{K} = \{1, \ldots, |K|\}$ index the set of identical delivery vehicles stationed at the depot and available to service customers, each having a capacity of $\mathcal{Q} > 0$. Furthermore, let d_{ij} denote the distance associated with a delivery vehicle travelling from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$.

The binary decision variables

$$x_{ijk} = \begin{cases} 1 & \text{if delivery vehicle } k \in \mathcal{K} \text{ travels from vertex } i \in \mathcal{V} \setminus \{n+1\} \text{ to vertex } j \in \mathcal{V} \setminus \{0\}, \\ 0 & \text{otherwise,} \end{cases}$$

capture all vehicle flows and are stored in row i and column j of slice k in a three-dimensional $(n+1) \times (n+1) \times |K|$ flow matrix $\boldsymbol{X}$. Furthermore, continuous auxiliary variables u_{ik}, where $i \in \mathcal{V}$ and $k \in \mathcal{K}$, denote the lower bound on the commodity load of vehicle k before reaching customer i. In the basic CVRP, the objective is to

$$\text{minimise } z = \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} d_{ij} x_{ijk}, \tag{2.1}$$

subject to the constraints

$$\sum_{k \in \mathcal{K}} \sum_{j \in \delta^+(i)} x_{ijk} = 1, \qquad i \in \mathcal{C}, \tag{2.2}$$

$$\sum_{j \in \delta^+(0)} x_{0jk} = 1, \qquad k \in \mathcal{K}, \tag{2.3}$$

$$\sum_{i \in \delta^-(j)} x_{ijk} - \sum_{i \in \delta^+(j)} x_{jik} = 0, \qquad j \in \mathcal{V}, \ k \in \mathcal{K}, \tag{2.4}$$

$$\sum_{i \in \delta^-(n+1)} x_{i,n+1,k} = 1, \qquad k \in \mathcal{K}, \tag{2.5}$$

$$u_{ik} - u_{jk} + Q x_{ijk} \leq Q - q_j, \quad (i,j) \in A, \;\; k \in K, \tag{2.6}$$

$$q_i \leq u_{ik} \leq Q, \qquad i \in \mathcal{V}, \;\; k \in \mathcal{K}, \tag{2.7}$$

$$x_{ijk} \in \{0,1\}, \quad (i,j) \in \mathcal{A}, \;\; k \in \mathcal{K}. \tag{2.8}$$

The objective function in (2.1) represents the total distance travelled by all delivery vehicles. The constraint set in (2.2) ensures that each customer is serviced exactly once. Furthermore, the constraint set in (2.3) ensures that each delivery vehicle departs from the depot exactly once, whereas the constraint set in (2.5) ensures that each delivery vehicle returns to the depot after having serviced the customers along its route. Moreover, the constraint set in (2.4) ensures that if a delivery vehicle arrives at a customer, it also departs from that customer. A delivery vehicle is said to have performed a *tour* if its route begins and ends at the depot and it visits each customer along its assigned route once. A *subtour*, on the other hand, is present if a delivery vehicle does not start and end at the depot [63]. It is therefore required to include constraint sets (2.6) and (2.7), which prevents subtour formation, and simultaneously ensures that the capacity of any delivery vehicle is not exceeded. Constraints that ensure adherence to delivery vehicle capacities are classified as the most basic type of constraint, since they can simply be imposed as an overall bound on the sum of the volume or weight of goods delivered at each customer in the route assigned to each delivery vehicle [59]. Finally, the constraint set in (2.8) restricts the decision variables in the matrix $\boldsymbol{X}$ so that these decision variables only assume binary variables.

An example of a CVRP instance and a corresponding optimal solution is illustrated on a Cartesian plane in Figure 2.1. In this figure, the volume of demand in cubic metres associated with each customer is indicated next to its vertex. In an optimal solution to this instance, three delivery vehicle are required to satisfy the demand of all the customers without exceeding the capacity of any delivery vehicle.

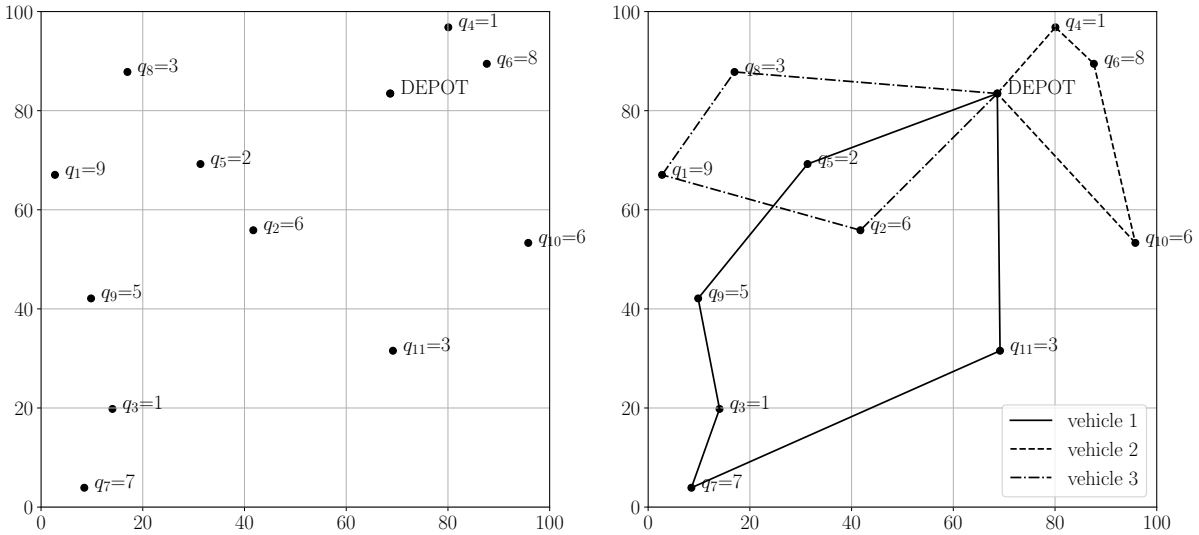


FIGURE 2.1: *A CVRP instance containing eleven customers and a single depot (on the left), and a corresponding optimal solution (on the right). The volume of commodities, q_i, in cubic metres to be delivered to each customer i is indicated next to its vertex.*

2.1.2 The VRPTW

The VRPTW is an extension of the CVRP. In the VRPTW, a time-window is specified for each customer during which service at the customer may start [14]. Delivery vehicles are permitted to arrive early at a customer, but must wait until the beginning of a customer's time-window before commencing to unload commodities. Arriving later than the specified time-window is prohibited. Furthermore, time-windows may be classified as either soft or hard. A *soft time-window* may be violated at a specified cost, whereas a *hard time-window* prohibits a vehicle from arriving at a customer after the end of its time-window [18]. The use of hard time-windows has been the most popular implementation, and is therefore considered in this project. Time-windows may further be classified as either tight or loose, and as either narrow or wide. A *loose time-window* does not influence the solution, and therefore serves as an inactive constraint, whereas a *tight time-window* influences and restricts the solution [18]. A time-window is classified as narrow or wide based on the significance of its duration compared to the duration of the planning horizon, for example, ten minutes would be considered narrow compared to a planning horizon of twelve hours, whereas six hours would be considered wide. A VRPTW reduces to a CVRP if the duration of all the time-windows are at least as large as the duration of the planning horizon [18]. Due to the VRPTW being an extension of the CVRP, similarities exist between the formulation of these two problems. Since the formulation of the VRPTW (in the form of an MIP problem) is based on the formulation of the CVRP, only the additional parameters, decision variables, and constraints required for the formulation of the VRPTW is described, as proposed by Desaulniers *et al.* [18].

The previously defined decision variables, x_{ijk}, are still applicable and capture all vehicle flows, stored in row i and column j of slice k in a three-dimensional $(n+1)\times(n+1)\times|K|$ flow matrix $\boldsymbol{X}$. Let c_{ij} denote the travel cost associated with a delivery vehicle travelling from vertex i to vertex j where $(i,j)\in\mathcal{A}$. A new parameter, the service duration, s_i, denotes the duration in minutes that it takes for a delivery vehicle to service customer $i\in\mathcal{C}$. A service time of zero minutes is associated with the depot (*i.e.* $s_0 = s_{n+1} = 0$). Furthermore, let a_i and b_i denote the start and end of the time-window at vertex $i\in\mathcal{V}$, respectively, measured in minutes from the start of the planning horizon. The time-window of the depot, denoted by $[a_0, b_0]$ and $[a_{n+1}, b_{n+1}]$, represent the earliest possible departure time and the latest possible return time at the depot. Furthermore, the expected travel time from vertex $i\in\mathcal{V}$ to vertex $j\in\mathcal{V}$ is denoted by t_{ij}.

The continuous decision variable $\mathcal{T}_{ik}$ denotes the service start time at vertex $i\in\mathcal{V}$ if serviced by delivery vehicle $k\in\mathcal{K}$. Note that the auxiliary decision variable u_{ik} defined in the mathematical formulation of the CVRP is no longer required. The objective function of the VRPTW is the same as that of the CVRP in (2.1). The vehicle flow constraint sets in (2.2)–(2.5) are also enforced in the VRPTW. Additional constraints are introduced to ensure feasibility with respect to customer time-windows. The constraint set in (2.6) is replaced with the linearised constraint set

$$T_{ik} + s_i + t_{ij} - T_{jk} \leq (1 - x_{ijk})M_{ij}, \quad k\in\mathcal{K},\ (i,j)\in\mathcal{A}, \tag{2.9}$$

where M_{ij} denotes a large constant which may be set to $\max\{b_i + s_i + t_{ij} - a_j, 0\}$, to ensure that the service start times at customers are correctly recorded and also to prevent subtour formation. The constraint set

$$a_i \leq T_{ik} \leq b_i, \quad k\in\mathcal{K},\ i\in\mathcal{V}, \tag{2.10}$$

ensures that the service at each customer starts during its specified time-window. Furthermore, to guarantee feasibility with respect to delivery vehicle capacity, the constraint set in (2.7) is

replaced with the constraint set

$$\sum_{i \in \mathcal{C}} (q_i \sum_{j \in \delta^+(i)} x_{ijk}) \leq Q, \quad k \in \mathcal{K}. \tag{2.11}$$

Finally, the constraint sets

$$x_{ijk} \in \{0,1\}, \quad (i,j) \in \mathcal{A}, \ k \in \mathcal{K}, \tag{2.12}$$

$$\mathcal{T}_{ik} \geq 0, \quad i \in \mathcal{V}, \ k \in \mathcal{K}, \tag{2.13}$$

enforce the binary and real-valued nature of the decision variables, respectively.

An optimal solution to a VRPTW instance is shown in Figure 2.2. This instance corresponds to the same problem instance considered in Figure 2.1, with the introduction of time-windows. Upon comparison of Figures 2.1 and 2.2, the effect of the addition of time-windows to the CVRP is evident. A different optimal solution was obtained in which the sequence of customers visited has changed. The time-window constraints contribute largely to the complexity of the problem due to routes being governed by either the capacity constraints or by the time-window constraints. This complexity stems from the underlying complexity of the spatial aspect of routing and the temporal aspect of scheduling, both of which must be performed to ensure adherence to time-windows [52]. The computation of a solution to a VRPTW instance by hand becomes increasingly difficult as the number of customers in the problem instance increase, as was also the case in a CVRP. Computerised solution methods therefore provide clear advantages for instances of the VRPTW that contain a large number of customers [41].

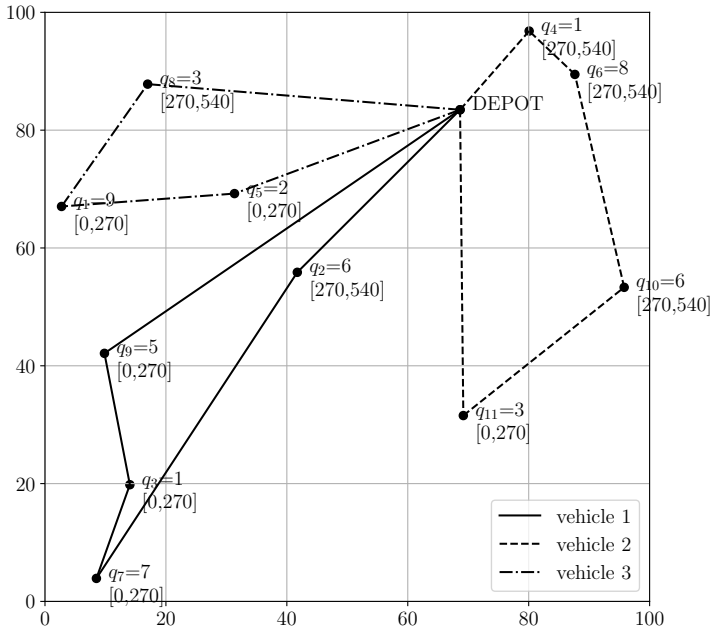


FIGURE 2.2: *An optimal solution to a VRPTW instance. The volume of commodities, q_i, in cubic metres and the time-window start and end times, $[a_i, b_i]$ in minutes from the start of the planning horizon associated with each customer are shown.*

2.1.3 The HFVRP

VRPs may be classified according to the fleet of delivery vehicles stationed at the depot and available to service customers. A homogeneous fleet consists of identical delivery vehicles that

have the same capacity and costs associated with each delivery vehicle. In real-world problems, however, retail organisations predominantly have a variety of delivery vehicles ranging in size available to service customers, partly due to the fact that some loading bays at stores may only be serviced by delivery vehicles of a certain size. Fleet dimensioning and composition selection is an important problem in industry, where retail organisations encounter a complex decision between owning and maintaining a fleet or hiring delivery vehicles from an external party [25]. As a result, a retail organisation must make an important decision concerning how many delivery vehicles of a particular type to acquire or rent. The problem may be modelled as the so-called *Fleet Size and Mix VRP* (FSM-VRP), which entails deciding on the composition of delivery vehicles to be utilised. This decision is influenced by several factors, including the delivery vehicle capacities and costs associated with utilising these delivery vehicles, as well as the expected demand to be exhibited by customers [31].

The FSM-VRP may be generalised as the HFVRP, according to the naming convention adopted by Irnich *et al.* [28]. In the HFVRP, the number of delivery vehicles available to service customers are limited. The FSM-VRP is typically employed during the strategic decision-making process of acquiring a fleet of delivery vehicles, whereas the HFVRP is concerned with the operational process of routing an available fleet of delivery vehicles, while minimising transportation cost or the total distance travelled [53]. In this project, emphasis is placed on the HFVRP, where a limited number of delivery vehicles are available to service customers.

In comparison with the previously defined CVRP, new parameters, decision variables, and constraints are introduced for the HFVRP, as proposed by Beldacci *et al* [7]. Let the set of delivery vehicle types to which each delivery vehicle may belong be indexed by $\mathcal{H} = \{1, \ldots, |\mathcal{K}|\}$. Furthermore, let m_h denote the number of delivery vehicles of type $h \in \mathcal{H}$ available to service customers. The objective of VRPs employed by retail organisations often aim to minimise the number of delivery vehicles utilised. It is therefore required to impose a fixed cost associated with utilising each delivery vehicle. This fixed cost accounts for wear and tear costs encountered in the case of owning the delivery vehicle, or the cost associated with renting the delivery vehicle in the case of hiring from an external organisation. The fixed cost associated with delivery vehicle type h is denoted $\mathcal{F}_h$. Let c_{ijh} denote the cost associated with a vehicle of type $h \in \mathcal{H}$ travelling from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$. Both the routing cost and delivery vehicle capacity vary according to the type of delivery vehicle utilised. The capacity associated with delivery vehicle type h is therefore denoted by $\mathcal{Q}_h$. As mentioned previously, in some practical applications customers have accessibility restrictions, with the implication that some customers are not able to be serviced by certain types of delivery vehicle. This is taken into consideration by simply assigning a very large value to the travel cost c_{ijh} of a delivery vehicle of type h travelling from customer i to a restricted customer j.

The decision variable y_{ijh} denotes the amount of goods carried by a delivery vehicle of type $h \in \mathcal{H}$ when travelling from vertex i to vertex j. Furthermore, the three-index binary decision variables

$$x_{ijh} = \begin{cases} 1 & \text{if delivery vehicle } h \in \mathcal{H} \text{ travels from vertex } i \in \mathcal{V} \setminus \{n+1\} \text{ to vertex } j \in \mathcal{V} \setminus \{0\}, \\ 0 & \text{otherwise.} \end{cases}$$

In the single-commodity flow formulation of the HFVRP, the objective is to

$$\text{minimise } z = \sum_{h \in \mathcal{H}} \sum_{j \in \mathcal{C}} x_{0jh} F_h + \sum_{h \in \mathcal{H}} \sum_{\substack{i,j \in \mathcal{V} \\ i \neq j}} c_{ijh} x_{ijh}, \tag{2.14}$$

subject to the constraints

$$\sum_{h\in\mathcal{H}}\sum_{i\in\mathcal{V}} x_{ijh} = 1, \qquad j \in \mathcal{C}, \tag{2.15}$$

$$\sum_{i\in\mathcal{V}} x_{iph} - \sum_{j\in\mathcal{V}} x_{pjh} = 0, \qquad p \in \mathcal{C},\ h \in \mathcal{H}, \tag{2.16}$$

$$\sum_{j\in\mathcal{C}} x_{0jh} \leq m_h, \qquad h \in \mathcal{H}, \tag{2.17}$$

$$\sum_{h\in\mathcal{H}}\sum_{i\in\mathcal{V}} y_{ijh} - \sum_{h\in\mathcal{H}}\sum_{i\in\mathcal{V}} y_{jih} = q_j, \qquad j \in \mathcal{C}, \tag{2.18}$$

$$q_j x_{ijh} \leq y_{ijh} \leq (Q_h - q_i)x_{ijh}, \qquad i,j \in \mathcal{V},\ i \neq j,\ h \in \mathcal{H}, \tag{2.19}$$

$$x_{ijh} \in \{0,1\}, \qquad i,j \in \mathcal{V},\ i \neq j,\ h \in \mathcal{H}, \tag{2.20}$$

$$y_{ijh} \geq 0, \qquad i,j \in \mathcal{V},\ i \neq j. \tag{2.21}$$

In the objective function in (2.14), the fixed cost associated with utilising delivery vehicles (the first term) and the total variable cost associated with each delivery vehicle's route (the second term) are minimised. The constraint set in (2.15) ensures that each customer is visited exactly once. Furthermore, the vehicle-flow constraint set in (2.16) ensures that if a delivery vehicle arrives at a customer, it must also depart from that customer. A route is considered feasible if the cumulative demand of customers included in the route does not exceed the capacity of the delivery vehicle utilised. The constraint set in (2.17) ensures that the number of delivery vehicles of a type utilised is no more than the number of delivery vehicles of that type stationed at the depot. The constraint sets in (2.18) and (2.19) are commodity flow constraint sets which ensure that the volume of commodities transported by each delivery vehicle across each arc does not exceed the capacity of the delivery vehicle utilised. Finally, the constraint sets in (2.20) and (2.21) enforce the real-valued and binary nature of decision variables, respectively.

An optimal solution to the instance considered in Figure 2.1, with the addition of a heterogeneous fleet of delivery vehicles available to service customers, is graphically illustrated in Figure 2.3. Information about the available fleet of delivery vehicles is presented in Table 2.1. Only four delivery vehicles were utilised in the optimal solution, two small delivery vehicles, one medium delivery vehicle, and one large delivery vehicle. Both of the small vehicles utilised 90% of their available capacities, whereas the medium and large delivery vehicles were fully utilised (100% of its available capacity was occupied). In comparison with the solution to the original CVRP instance in Figure2.1, the HFVRP returns a different set of recommended routes. This solution yields a fixed cost of R12 750.00 and a variable cost of R3 356.35, resulting in a objective function value of R16 106.00.

TABLE 2.1: *Input data related to the available fleet of delivery vehicles stationed at the depot. The number of delivery vehicles available, the capacity in cubic metres, and the fixed cost in Rand associated with each type of delivery vehicle are listed.*

Type, h	Number of vehicles, m_h	Fixed cost, $\mathcal{F}_h$	Capacity, $\mathcal{Q}_h$
Small	2	3 000	10
Medium	3	3 250	15
Large	1	3 500	18

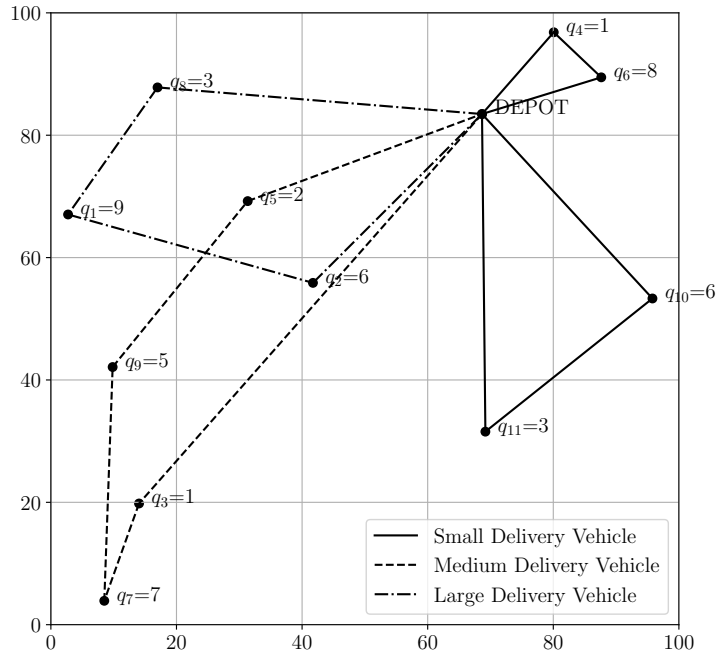


FIGURE 2.3: *An optimal solution to a HFVRP instance. The volume of commodities, q_i in cubic metres associated with each customer i is shown, as well as the type of delivery vehicles utilised for each route.*

2.1.4 The SDVRP

In the SDVRP, customers may be serviced by more than one delivery vehicle. The SDVRP is a relaxation of the CVRP in which the demand exhibited by each customer may be satisfied by *split deliveries* [28]. The CVRP and SDVRP share a common goal, which is to satisfy the demand of customers, while minimising the total transportation cost. The allowance of split deliveries may result in significant cost savings compared to the CVRP, by requiring possibly fewer delivery vehicles and, consequently, reducing the fixed cost associated with the delivery vehicles utilised [10]. Since customers may receive multiple visits, a feasible solution may be returned in the case of a customer's demand being greater than the capacity of the largest delivery vehicle, contrary to the CVRP discussed in §2.1.1 [3]. In the case of each customer being visited only once, the SDVRP reduces to the CVRP. In the SDVRP, an unlimited and homogeneous fleet of delivery vehicles is assumed to be available. The SDVRP is a complex problem that, until recently, could only be solved optimally in a systematic manner for instances with fewer than 30 customers, and is therefore mostly motivated by real-world applications [3].

In the SDVRP formulation proposed by Archetti *et al.* [4] the set of customers is denoted as $\mathcal{C} = \{1, \ldots, n\}$. The depot is included in the additional supplemented set of vertices, $\mathcal{V} = \mathcal{C} \cup 0$, as opposed to the CVRP formulation where the vertex is accounted for separately concerning the departure and return of delivery vehicles. As previously defined, q_i denotes the demand exhibited by vertex $i \in \mathcal{V}$, with $q_0 = 0$. Furthermore, $\mathcal{Q}$ denotes the capacity of each delivery vehicle that belong to the homogeneous fleet of delivery vehicles, $\mathcal{K} = \{1, \ldots, |K|\}$, stationed at the depot. A slight change in the constraints of the CVRP is required to allow for $q_i > Q$, which accounts for the fact that the SDVRP may return feasible solutions to instances in which customers exhibit larger demand volumes than the capacity of delivery vehicles. Furthermore, d_{ij} denotes the distance associated with a delivery vehicle travelling from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$.

The demand volume exhibited by customer $i \in \mathcal{C}$ and delivered by delivery vehicle $k \in \mathcal{K}$ is denoted by the decision variable r_{ik}. Once again, the binary decision variables

$$x_{ijk} = \begin{cases} 1 & \text{if delivery vehicle } k \in \mathcal{K} \text{ travels from vertex } i \in \mathcal{V} \text{ to vertex } j \in \mathcal{V}, \\ 0 & \text{otherwise,} \end{cases}$$

capture all vehicle flows, stored in row i and column j of slice k in a three-dimensional $(n+1) \times (n+1) \times |K|$ flow matrix $\boldsymbol{X}$. In the SDVRP, the objective is to

$$\text{minimise } z = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \sum_{k \in \mathcal{K}} d_{ij} x_{ijk}, \tag{2.22}$$

subject to the constraints

$$\sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{K}} x_{ijk} \geq 1, \qquad j \in \mathcal{V}, \tag{2.23}$$

$$\sum_{i \in \mathcal{V}} x_{ipk} - \sum_{j \in \mathcal{V}} x_{pjk} = 0, \qquad p \in \mathcal{V}, \ k \in \mathcal{K}, \tag{2.24}$$

$$\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} x_{ijk} \leq |C| - 1, \qquad k \in \mathcal{K}, \tag{2.25}$$

$$r_{ik} \leq q_i \sum_{j \in \mathcal{V}} x_{ijk}, \quad i \in \mathcal{C}, \ k \in \mathcal{K}, \tag{2.26}$$

$$\sum_{k \in \mathcal{K}} r_{ik} = q_i, \qquad i \in \mathcal{C}, \tag{2.27}$$

$$\sum_{i \in \mathcal{V}} r_{ik} \leq Q, \qquad k \in \mathcal{K}, \tag{2.28}$$

$$x_{ijk} \in \{0, 1\}, \qquad i \in \mathcal{V}, \ j \in \mathcal{V}, \ k \in \mathcal{K}, \tag{2.29}$$

$$r_{ik} \geq 0, \qquad i \in \mathcal{C}, \ k \in \mathcal{K}. \tag{2.30}$$

In the objective function in (2.22), the distance travelled by the delivery vehicles are minimised. The constraint sets in (2.23)–(2.25) serve as classical flow conservation constraint sets and ensure that each vertex is visited at least once, a delivery vehicle departs from a customer if it arrives at that customer, and that any subtours are eliminated, respectively. The constraint set in (2.26) imposes that a customer may only receive commodities from a delivery vehicle if it is, in fact, visited by that delivery vehicle. The constraint sets in (2.27) and (2.28) is concerned with the allocation of demand among the delivery vehicles. The constraint set in (2.27) ensures that each customer's demand is fully satisfied, whereas the constraint set in (2.28) restricts delivery vehicle capacities from being exceeded. Finally, the nature of the decision variables is enforced by the constraint sets in (2.29) and (2.30).

An optimal solution to an SDVRP instance, which relates to the same problem instance considered in Figure 2.1, is shown in Figure 2.4, but in this instance split deliveries are allowed. An example of a split delivery in this solution is the demand of Customer 10 that is satisfied by visits from Vehicles 2 and 3. Vehicle 2 delivers commodities with a volume of $4m^3$, and Vehicle 3 delivers commodities with a volume of $2m^3$. Together, these delivery vehicles fulfil the entire demand exhibited by Customer 10.

2.2 Exact solution methods

This section is devoted to a discussion on exact methods for solving MIP problems that are relevant to this project. The working of each of the discussed methods are illustrated by means

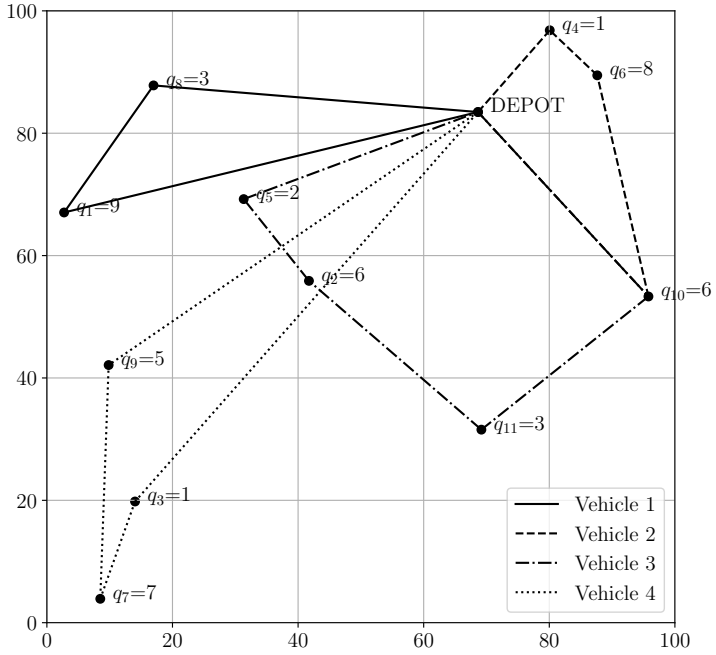


FIGURE 2.4: *An optimal solution to an SDVRP instance. The volume of commodities, q_i, exhibited by customers is shown in cubic metres next to the index of each vertex, i.*

of a small example problem instance. The section opens in §2.2.1 with a prerequisite discussion on the simplex algorithm for solving LP problems, upon which all of the methods discussed thereafter are based. Three exact solution methodologies for solving MIP problems are discussed, namely the branch-and-bound method in §2.2.2, the cutting plane method in §2.2.3, and the branch-and-cut method in §2.2.4. The example problem used to illustrate the working of each of the discussed methods is a two-variable example, as illustrated by Winston and Goldberg [63], in which the objective is to

$$\text{maximise } z = 8x_1 + 5x_2, \tag{2.31}$$

subject to constraints

$$\begin{aligned} x_1 + x_2 &\leq 6, & (2.32)\\ 9x_1 + 5x_2 &\leq 45, & (2.33)\\ x_1, x_2 &\in \mathbb{N}. & (2.34) \end{aligned}$$

2.2.1 The simplex algorithm for linear programming

Linear programming (LP) is an important tool in the field of Operations Research and may be described as an instrument for modelling optimisation problems [63]. LP was first introduced by Dantzig [17] in 1947, shortly after World War II, for solving optimisation problems related to military planning activities. Formally, LP is concerned with the maximisation or minimisation of a linear objective function, subject to linear equality and inequality constraints [17]. More recently, LP problems have been used widely to solve optimisation problems in a diverse range of industries. Although good approximate solution methodologies perform satisfactory in many applications, some scenarios require exact solution approaches. The simplex algorithm is an efficient method for solving LP instances to optimality [2]. Although it is possible to solve LP models graphically, most real-life LP problems have many decision variables making a graphical

representation of the decision space difficult, or impossible. The simplex algorithm is able to solve very large LP problems, with up to thousands of constraints and decision variables [63].

In LP problem instances, decision variables are allowed to be fractional. An *integer programming* (IP) problem, on the other hand, is an LP problem in which all of the decision variables are required to assume non-negative integer values [63]. In real-life applications, it is often the case that decision variables may only adopt integer values, and the problem must therefore be formulated as an IP problem. The classes of problems that are often referred to as sequencing, scheduling, and routing, are inherently modelled as IP problems [9]. Furthermore, an IP problem in which only some of the variables are required to have integer values, is an MIP problem [32]. In some cases, decision variables in an IP problem may only assume the values zero or one in which case it is referred to as a binary IP problem. The LP model obtained by relaxing all integer or binary constraints for decision variables is called the LP relaxation of the IP problem instance [63].

LP problems may contain both equality and inequality constraints, and may also consist of decision variables that are unrestricted in sign. The simplex algorithm requires LP problems to be in *standard form*. Standard form requires the equivalent problem to only consist of equality constraints and non-negative variables. A maximisation LP problem with m constraints and n variables, denoted by $x_1, \ldots, x_n$, may be written in standard form, which is to

$$\text{maximise } z = c_1 x_1 + \ldots + c_n x_n$$

subject to the constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ \vdots \quad &\vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m, \\ x_i &\geq 0, \qquad i = 1, \ldots, n. \end{aligned}$$

The constraints of the standard form of an LP may be written as a system of equations $\boldsymbol{A}x{=}\boldsymbol{b}$, where

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

and

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n, \end{bmatrix} \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m. \end{bmatrix}$$

The system contains m linear equations and n decision variables. If $n \geq m$, a *basic solution* to the system $\boldsymbol{A}x{=}\boldsymbol{b}$ may be obtained by assigning $n - m$ variables, called the *non-basic variables*, a value of zero. Upon solving for the remaining m variables, called the *basic variables*, a unique solution may be obtained that satisfy $\boldsymbol{A}x{=}\boldsymbol{b}$, called a *basic solution*. The different combinations of non-basic variables assigned the value zero will yield a set of different solutions. Any solution from this set of basic solutions in which all variables are non-negative, is called a *basic feasible solution*. This is an important notion, because if an LP problem instance has an optimal

solution, then the solution has to form part of the set of basic feasible solutions. Accordingly, the search space for an optimal solution is greatly simplified and the solution may be found by searching for only a finite number of points [63]. By making use of *simplex tableaus*, a set of basic feasible solutions may be produced iteratively by the simplex algorithm. During each iteration, the information of a neighbouring basic feasible solution is summarised in tableau format, illustrated in Figure 2.5. Each neighbouring basic feasible solution corresponds to an objective function value that is strictly better than the objective function value of the previous basic feasible solution, until the objective function cannot improve any further, and an optimal basic feasible solution has been reached. The detailed workings of the Simplex Algorithm is summarised in pseudo-code form in Algorithm 2.1.

		c_1	c_2	$\cdots$	c_n		
		x_1	x_2	$\cdots$	x_n	RHS	θ
Coefficients	Basic Variables	a_{11}	a_{12}	$\cdots$	a_{1n}	b_1	θ_1
		a_{21}	a_{22}	$\cdots$	a_{2n}	b_2	θ_2
		$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
		a_{m1}	a_{m2}	$\cdots$	a_{mn}	b_n	θ_m
	g	g_1	g_2	$\cdots$	g_n	Objective	
	z	z_1	z_2	$\cdots$	z_n	Function	

FIGURE 2.5: *The general format of a simplex tableau.*

Algorithm 2.1: Simplex algorithm (maximisation) [61]

Input : An LP problem in standard form.
Output: An optimal solution to the LP problem (if one exists).

1 Construct a simplex tableau from the standard form of the LP problem instance.
2 Compute the j-th entries of the g-row and z-row as $g_j = \sum_{i=1}^{m} c_i a_{ij}$ and $z_j = g_j - c_j$ for all $j = 1, \ldots, n$.
3 **if** $z_j \geq 0$ *for all* j **then**
4 An optimal solution has been found, stop.
5 **else if** $z_j < 0$ *for at least one* j, **then**
6 Continue.
7 Select the basic variable which should enter the basis (the variable, in column q, corresponding to the most negative value in the z_j-row). Column q is called the *pivot column*.
8 **if** $a_{iq} \leq 0$ *for all* i **then**
9 The objective function is unbounded, stop.
10 Select the basic variable which should leave the basis (the variable in row $i = p$, corresponding to the smallest ratio $\theta_i = b_i/a_{iq}$ when only considering positive values of a_{iq}). Row p is called the *pivot row*.
11 Perform a simplex iteration:

(a) Divide the values in the pivot row by a_{pq} (the pivot element).

(b) Compute the values of the new non-pivot row entries as:
new non-pivot row$_i$ = *old non-pivot row*$_i$ − $a_{iq}\times$ *new pivot row.*

12 Return to Step 2.

Upon applying Algorithm 2.1 to the LP problem instance formulated in (2.31)–(2.34), the problem instance is required to be in standard form before constructing a simplex tableau. The problem instance may be rewritten in standard form by introducing slack variables s_1 and s_2 so that the objective is to

$$\text{maximise } z = 8x_1 + 5x_2, \tag{2.35}$$

subject to the constraints

$$x_1 + x_2 + s_1 = 6, \tag{2.36}$$
$$9x_1 + 5x_2 + s_2 = 45, \tag{2.37}$$
$$x_1, x_2, s_1, s_2 \geq 0. \tag{2.38}$$

The feasible region of the LP problem instance in (2.31)–(2.34) is illustrated graphically by the grey area in Figure 2.6. The simplex tableau may now be constructed from the standard form of the LP problem instance. The initial simplex tableau constructed according to Step 1 of Algorithm 2.1 is

		8	5	0	0		
		x_1	x_2	s_1	s_2	RHS	θ
0	s_1	1	1	1	0	6	6
0	s_2	9	5	0	1	45	5 ←
	g	0	0	0	0	$\boxed{0}$.	
	z	-8	-5	0	0		
		↑					

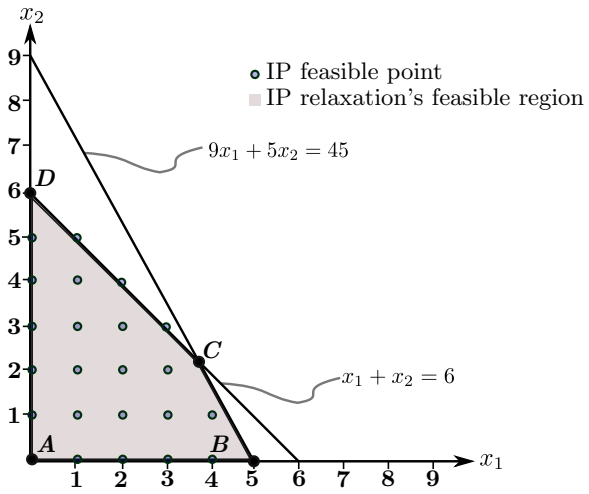


FIGURE 2.6: *The feasible region of the IP problem instance in (2.31)–(2.34).*

The first iteration of the simplex tableau shown above has obtained a feasible solution in which $x_1 = 0$, $x_2 = 0$, $s_1 = 6$ and $s_2 = 45$, with an objective function value of $z = 0$. According to the condition in Steps 3–4, an optimal solution has not been reached since not all z_j values are non-negative. Furthermore, upon consideration of Steps 5–6, it is found that the z-row contains two negative entries. During the execution of Step 7, it is found that the column associated with variable x_1 corresponds to the most negative entry in the z-row. Accordingly, the pivot column is identified as the column associated with the non-basic variable x_1. Upon execution of Steps 8–9, it is evident that the objective function is bounded and the algorithm is therefore continued. According to Step 10, the row associated with the basic variable s_2 is considered as

the pivot row. Another simplex iteration is performed according to Step 11 of the algorithm. During the next iteration, the simplex tableau

		8	5	0	0			
		x_1	x_2	s_1	s_2	RHS	θ	
0	s_1	0	$\frac{4}{9}$	1	$-\frac{1}{9}$	1	$\frac{9}{4}$	$\leftarrow$
8	x_1	1	$\frac{5}{9}$	0	$\frac{1}{9}$	5	9	
	g	8	$\frac{40}{9}$	0	$\frac{8}{9}$	$\boxed{40}$		
	z	0	$-\frac{5}{9}$	0	$\frac{8}{9}$			
			$\uparrow$					

is obtained. The basic feasible solution corresponding to this tableau, $x_1 = 5$, $x_2 = 0$, $s_1 = 1$ and $s_2 = 0$, yields an objective function value of $z = 40$. The solution obtained is once again considered to not be optimal, according to Steps 3–6 of the algorithm. Continuing, the pivot column is identified as the column associated with the non-basic variable x_2. According to Step 10, the pivot row is the row associated with the basic variable s_1. Another simplex iteration is performed and the simplex tableau

		8	5	0	0	
		x_1	x_2	s_1	s_2	RHS
5	x_2	0	1	$\frac{9}{4}$	$-\frac{1}{4}$	$\frac{9}{4}$
8	x_1	1	0	$-\frac{5}{4}$	$\frac{1}{4}$	$\frac{15}{4}$
	g	8	5	$\frac{5}{4}$	$\frac{3}{4}$	$\boxed{41.25}$
	z	0	0	$\frac{5}{4}$	$\frac{3}{4}$	

is obtained. The new basic feasible solution found is $x_1 = \frac{15}{4}$, $x_2 = \frac{9}{4}$, $s_1 = 0$ and $s_2 = 0$, and corresponds to an objective function value of $z = 41.25$. Upon consideration of Steps 3–4, this solution is an optimal solution to the LP problem instance, since the z-row consists of only non-negative entries. Points A, B and C in Figure 2.6 correspond to the solutions found in the three simplex tableaus above (in the order in which they were uncovered). The simplex algorithm is an efficient and reliable method for solving LP problem instances to optimality. Instead of considering each feasible point in the feasible region, it traverses the search space in a much more efficient manner.

2.2.2 The branch-and-bound method

If the set of feasible solutions to the related LP problem of a MIP problem is bounded, then the integer valued variables in the feasible region can only take on a finite number of values [32]. The branch-and-bound method is a technique of implicit enumeration, where a list of some of the feasible integral solutions are generated and the solution corresponding to the best objective function value is kept in memory for comparison with the objective function values of succeeding solutions generated. If all decision variables in an optimal solution to the LP relaxation of a pure IP problem instance assume integer values, then the optimal solution to the LP relaxation is also an optimal solution to the corresponding IP problem instance [63]. Furthermore, in for example a maximisation problem instance, the objective function value corresponding to an optimal solution to the LP relaxation serves as an upper bound to the optimal objective function value of the corresponding IP problem instance. The branch-and-bound method, proposed by Dakin in 1966 [15], is a popular method for solving IP problem instances. The branch-and-bound method systematically divides the feasible region of the LP relaxation corresponding to the IP problem

instance being solved, into subproblems. The goal is to divide the set of feasible solutions into several subsets and then reject many of these subsets because they are implicitly enumerated. The creation of subproblems may be represented by an enumeration tree diagram, as shown in Figure 2.7. Each subproblem is referred to as a *node*, and the lines connecting the nodes of the tree is referred to as *arcs*. The constraints associated with the subproblem corresponding to a node are the constraints for the LP relaxation and all the constraints associated with the arcs leading from the initial subproblem to the node [63].

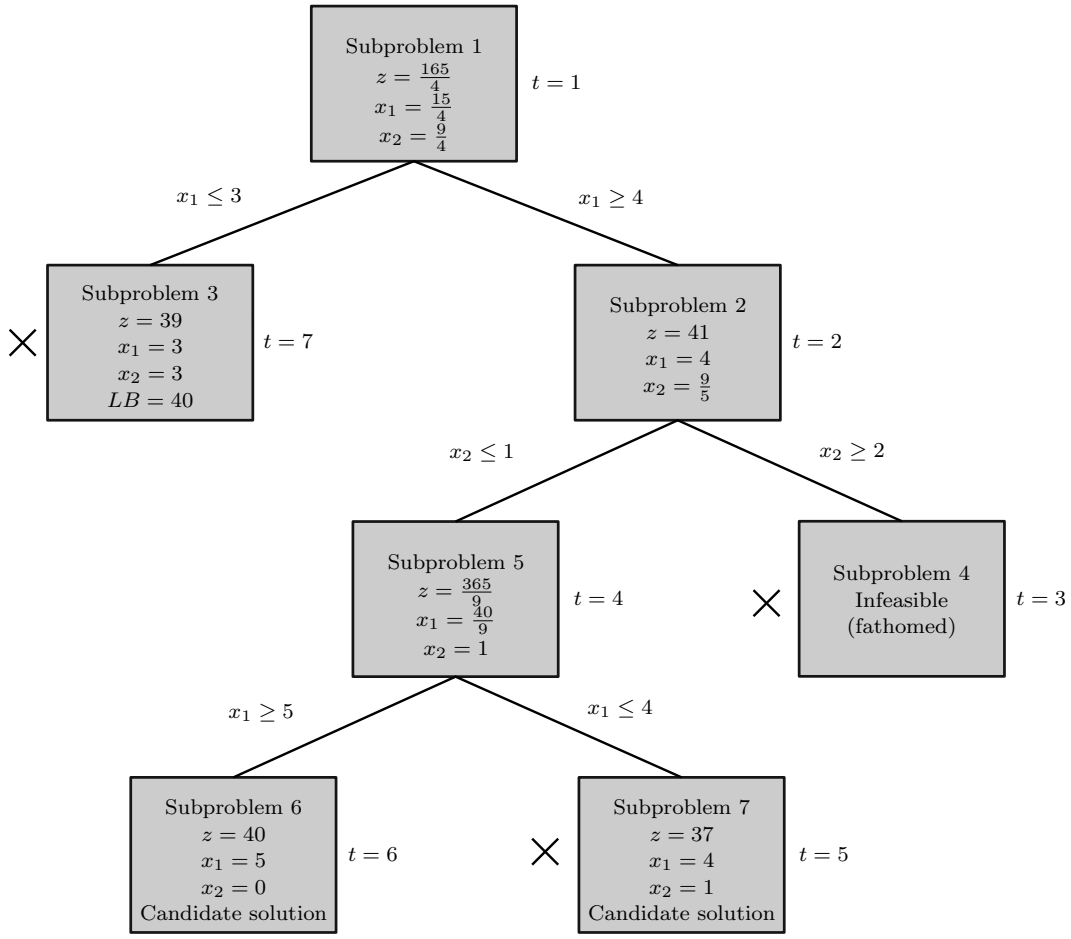


FIGURE 2.7: *The branch-and-bound tree obtained when solving the IP problem instance in (2.31)–(2.34) by means of following the LIFO rule. The order in which subproblems are considered is indicated by the t-values next to each node.*

The branch-and-bound method consists of two main phases, called *branching* and *bounding* [43]. When subproblems are formulated by adding constraints, the procedure is referred to as branching. After constructing a rooted binary tree where each node represents an LP problem and its corresponding solution, branching occurs on a fractional variable value in the parent problem. If, for example, an integer decision variable x assumes a non-integer value having an integer part a and a fractional part b, the constraints $x \leq a$ and $x \geq a + 1$ are imposed in the two subproblems, respectively. Bounding occurs when a solution that is worse than the best solution found thus far is encountered, in which case the entire branch of descendants of such an inferior solution is ignored without loss. A solution to a subproblem may be considered to be a *candidate solution* if all integer variables do in fact assume integer values. There exists the possibility that a candidate solution may be an optimal solution to the IP problem instance, therefore requiring that the candidate solution be kept in memory until a better candidate solution (if any exist)

is found. The value of the best candidate solution found while branching on a problem may be considered to be a *lower bound* on the optimal objective function value for the corresponding IP problem instance [9]. A node can not be further branched if either

1. the subproblem is infeasible,
2. the subproblem yields an optimal solution in which all variables assume integer values, or
3. the optimal objective function value of the subproblem does not exceed the current lower bound, in the case of a maximisation problem instance.

In any of these three cases, the node under consideration is deemed to be *fathomed* [63]. When all nodes in the branch-and-bound tree is fathomed, the integer solution of the subproblem corresponding to the current best (largest) lower bound is the optimal solution to the original IP problem instance. During the bounding procedure, a subproblem and all of its descendants may be eliminated from the process if

1. the subproblem is infeasible, or
2. the lower bound is at least as large as the objective function value of the solution to the subproblem (if maximising).

A minimisation IP problem instance may be solved with the same approach by simply multiplying the objective function with the value -1 and solving it as a maximisation problem instance.

Two general approaches commonly employed for governing the order in which subproblems should be visited, is the *backtracking* and *jumptracking* approaches. The approach chosen may influence the number of steps (which directly affects the computational time) required to find an optimal solution. The efficiency of each technique is dependent on the type of problem being solved. The *last-in, first-out* (LIFO) rule is the most widely adopted approach, and is referred to as backtracking in the literature. In this approach, the most recently created subproblem is solved next. One side of the branch-and-bound tree is considered first, whereafter it requires backtracking, where nodes in other branches are traversed. In the jumptracking approach, all subproblems are created and the node corresponding to the best objective function value is branched on next. As a result, consecutive nodes on opposite sides of the branch-and-bound tree may be considered. The jumptracking approach aims to produce an optimal solution relative quickly, but may create more subproblems and is more memory intensive than the backtracking approach [63].

The working of the branch-and-bound method when adopting the LIFO approach is demonstrated for the example problem instance in (2.31)–(2.34). The branch-and-bound tree obtained when solving this problem instance is illustrated graphically in Figure 2.7. Subproblem 1, represented by the first node, corresponds to an optimal solution to the LP relaxation of the original IP problem instance which is obtained by invoking the simplex algorithm discussed in §2.2.1. The optimal objective function value of this subproblem (which is $z = \frac{165}{4}$) serves as an upper bound to the original IP problem instance. The variable x_1 assumes a non-integer value and is associated with the largest coefficient in the objective function, and is therefore chosen to branch on. Two new constraints are imposed to restrict the variable x_1 and are added as two new branches to the node representing Subproblem 1.

Subproblem 2 is arbitrarily chosen to be solved next. The optimal solution to Subproblem 2 does not yield an all-integer solution. It is therefore required to continue branching on the

current node. The variable x_2 (assuming a non-integer value), is branched on to create two new subproblems, Subproblem 4 and Subproblem 5. The annotations $t = 1, t = 2, \ldots$ indicate the order in which the tree is traversed. Subproblem 4 is chosen arbitrarily to be solved next. Subproblem 4 is infeasible and considered to be fathomed. A cross next to the subproblem indicates the abandonment of a candidate solution due to one of the aforementioned reasons. Subproblem 5 is solved and yields a solution corresponding to $x_1 = \frac{40}{9}$, $x_2 = 1$ and an objective function value of $z = \frac{365}{9}$. The variable x_1, which assumes a non-integer value, is branched on next to create Subproblem 6 and Subproblem 7.

Subproblem 7 is chosen to be solved first, and results in a candidate solution with and objective function value of $z = 37$. This solution is kept in memory as the current best feasible solution. Subproblem 6 is considered next, and results in a feasible objective function value of $z = 40$. The solution is better than the previously obtained objective function value of Subproblem 7, and is considered to be the current best feasible solution. The optimal objective function value of Subproblem 7 does not exceed the current lower bound, and a cross next to the subproblem indicates that it does not need to be considered further. Finally, Subproblem 3 is considered when backtracking up the tree. Subproblem 3 yields an objective function value of $z = 39$, which does not exceed the current best feasible solution in memory. Since all subproblems have been considered, an optimal solution corresponding to $x_1 = 5$, $x_2 = 0$ and $z = 40$ is returned by the algorithm, which correspond to the solution obtained upon solving Subproblem 6.

2.2.3 The cutting plane method

Methods for solving IP problem instances are not as general in their application as methods for solving LP problem instances, since there is no single algorithm that works well for all IP problem instances [32]. Furthermore, the difficulty of solving these problems are mainly due to the methods' inefficiency for solving even medium-sized problems. An approach that aims to address this problem by efficiently solving IP problem instances, is the *cutting plane method*, proposed by Ralph Gomory [23] in 1958. The cutting plane method has had a significant role in the evolution of IP solution methods. It is the first IP solution method that was proven to converge to an optimal solution within a finite number of steps. Furthermore, it has provided significant insights into the field of IP solution methods that have led to other, more efficient, methods [9].

The workings of the cutting plane algorithm relies on invoking the simplex algorithm for solving the LP relaxation of IP problem instances. If all variables in an optimal solution to an LP relaxation assumes integer values, an optimal solution has been found and the process is terminated, otherwise, a new constraint is added which eliminates some of the non-integral solutions obtained previously by the simplex algorithm. The newly imposed constraint should be constructed so as to not eliminate any feasible integer solutions. The cutting-plane algorithm does not divide the feasible region into subdivisions, as in the branch-and-bound approach, but rather considers a single LP problem instance that is refined by iteratively imposing new constraints, which are solved by invoking the simplex algorithm. The constraints are added incrementally reduce the feasible region of the LP relaxation until an optimal solution is found in which all required variables assume integer values [9].

The cutting plane method comprises three high-level steps, which is to

1. find the optimal simplex tableau for the LP relaxation of the IP problem instance,
2. formulate an additional constraint to form a cutting plane, and

3. construct a dual simplex tableau with the cutting plane inserted as an additional constraint to find an optimal solution to the LP relaxation.

The method invoked for generating the cutting plane constraint is called the *Glomory fractional cut* [36]. A cut generated according to this method has an important characteristic – any feasible point for the IP problem instance will satisfy the cut, whereas the current optimal solution to the LP relaxation does not satisfy the cut. This implies that the cut will eliminate the current optimal solution to the LP relaxation, but not any feasible solutions to the IP problem instance [63]. If the solution to the LP relaxation does not consist of only integer valued variables, then at least one of the rows of the simplex tableau contains a non-integer value. A row containing a non-integer value may be chosen arbitrarily to deduce the next cut from. By convention, the row with a binding constraint and with the closest fractional value to $\frac{1}{2}$ is chosen, which may result in faster convergence to an optimal solution. To formulate the cutting constraint, the chosen equation is split into an integer part and a non-negative decimal part. The equation is rewritten so that the fractional values are represented by the summation of an integer value and a non-negative decimal portion, for example $2.25x = 2x + 0.25x$, while $-2.75x = -3x + 0.25x$. The cut is formulated by rearranging the resulting equation to have all the terms with integer coefficients on the left-hand side of the equality and all the terms with fractional coefficients on the right-hand side. Set the right-hand side (all the fractional coefficients) less than or equal to zero. This ensures that all the feasible points of the IP problem instance satisfies the cut, whereas the current optimal solution to the LP relaxation does not.

A pseudo-code description of the cutting plane algorithm is provided in Algorithm 2.2, and it is illustrated by means of the example IP problem instance in (2.31)–(2.34). After rewriting the original problem instance in standard form (as in (2.35)–(2.38)), the simplex algorithm is invoked to obtain an optimal tableau for the LP relaxation of the problem instance. Following Steps 2–3 of the cutting plane algorithm, the second constraint, corresponding to the equality equation $x_1 - 1.25s_1 + 0.25s_2 = 3.75$, is chosen to construct a cutting plane from. Steps 4–5 of the algorithm instructs rearranging the equation in the manner explained above, where the fractional values are written as the summation of the integer value and a non-negative decimal portion. The corresponding equality is

$$x_1 - 2s_1 + 0.75s_1 + 0.25s_2 = 3 + 0.75. \tag{2.39}$$

To generate the cut, according to Step 5, the terms with integer coefficients are separated from the terms with fractional coefficients. Consequently, the constraint in (2.39) is rearranged as

$$x_1 - 2s_1 - 3 = 0.75 - 0.75s_1 - 0.25s_2. \tag{2.40}$$

Finally, the cut is generated, according to Step 6 of the algorithm, by setting the right-hand side of (2.40) smaller or equal to zero, so that

$$0.75 - 0.75s_1 - 0.25s_2 \leq 0. \tag{2.41}$$

This cut successfully eliminates the current optimal solution to the LP relaxation while maintaining all feasible solutions to the IP problem instance. The cut may be depicted graphically by removing the slack variables and rewriting the cut in terms of x_1 and x_2. The constraints in (2.36) and (2.37) may be used to replace s_1 and s_2 in (2.41). The cut is rewritten as $3x_1 + 2x_2 \leq 15$, as shown in Figure 2.8. Continuing with Step 7 of the algorithm, the inequality constraint in (2.41) is rewritten in standard form and slack variable s_3 is introduced. The formulated cut,

$$-0.75s_1 - 0.25s_2 + s_3 = -0.75, \tag{2.42}$$

Algorithm 2.2: The cutting plane algorithm (maximisation) [63]

Input : An IP problem instance in standard form.
Output: An optimal solution to the IP problem instance.

1 Find the optimal solution to the corresponding LP relaxation.
2 If all variables in the optimal solution assume integer values, then an optimal solution to the IP problem has been found. Stop.
3 Else, pick a binding constraint in the LP relaxation optimal tableau with a fractional valued right-hand side which will be used to generate a cut. For possible faster convergence, generate the cut by using the binding constraint with a right-hand side closest to $\frac{1}{2}$.
4 Rewrite the each variable's coefficient as an integer part and non-negative part.
5 Rewrite the constraint used to generate the cut as

all terms with integer coefficients = all terms with fractional coefficients.

6 The cut is formulated by setting

all terms with fractional coefficients ≤ 0.

7 With the cut as an additional constraint, employ the dual simplex algorithm to find an optimal solution to the LP relaxation.
8 If all variables assume integer values, an optimal solution to the IP problem has been found. Stop.
9 Else, return to step 3.

is added as an additional constraint. The simplex tableau is updated and solved as

		8	5	0	0	0	
		x_1	x_2	s_1	s_2	s_3	RHS
5	x_2	0	1	$\frac{9}{4}$	$-\frac{1}{4}$	0	$\frac{9}{4}$
8	x_1	1	0	$-\frac{5}{4}$	$\frac{1}{4}$	0	$\frac{15}{4}$
0	s_3	0	0	$-\frac{3}{4}$	$-\frac{1}{4}$	1	$-\frac{3}{4}$ ←
	g	8	5	$\frac{5}{4}$	$\frac{3}{4}$	0	$\boxed{41.25}$.
	z	0	0	$\frac{5}{4}$	$\frac{3}{4}$	0	
	θ'	–	–	$\frac{5}{9}$	3	–	
				↑			

Upon imposing the new constraint in (2.42), the resulting simplex tableau corresponds to the infeasible region, and an iteration of the dual simplex algorithm is required to obtain feasibility, according to Step 7. In the dual simplex algorithm, the pivot row is identified first, in contrast to the simplex method explained in Algorithm 2.1. The pivot row corresponds to the row containing the basic variable assuming the most negative value in the right-hand side column (therefore, the row associated with the variable s_3 in the tableau above) [63]. After the pivot row is identified, the ratio

$$\theta' = \frac{\text{coefficient of } x_j \text{ in the } z\text{-row}}{\text{coefficient of } x_j \text{ in the pivot row}} \tag{2.43}$$

is calculated for each column in the tableau. The column resulting in the smallest absolute value of θ' is identified as the pivot column (the column associated with the variable s_1 above).

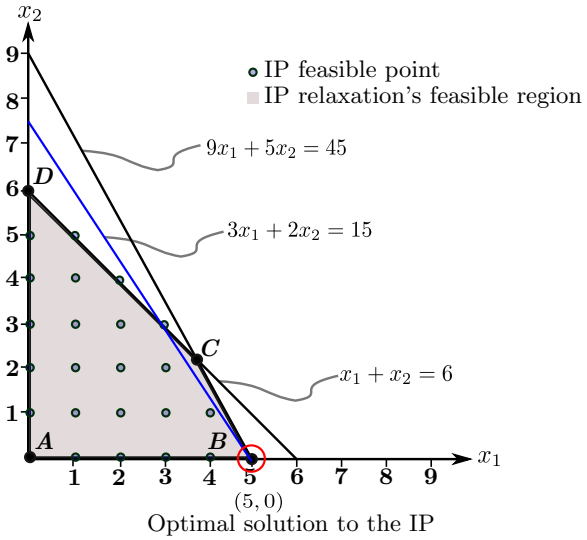


FIGURE 2.8: *A cutting plane (represented by the blue line) inserted to reduce the feasible region for an IP problem instance.*

Finally, Gauss-Jordan elimination is invoked to obtain feasibility. Upon assessment the tableau, according to Steps 8–9, an optimal solution to the IP problem instance has not been found and the tableau contains variables $x_1 = \frac{15}{4}$, $x_2 = \frac{9}{4}$ and $s_3 = -\frac{3}{4}$ assuming non-integer values. According to the dual simplex algorithm, the variable s_3 is associated with the resulting pivot row (since it is the only variable that consists of a negative right-hand side entry) and s_1 is associated with the resulting pivot column (since it is associated with the smallest entry in the z-row). Another iteration of the simplex algorithm is performed, and the resulting tableau is

		8	5	0	0	0	
		x_1	x_2	s_1	s_2	s_3	RHS
5	x_2	0	1	0	-1	3	0
8	x_1	1	0	0	$\frac{2}{3}$	$-\frac{5}{3}$	5
0	s_1	0	0	1	$\frac{1}{3}$	$-\frac{4}{3}$	1
	g	8	5	0	$\frac{1}{3}$	$\frac{5}{3}$	$\boxed{40.}$
	z	0	0	0	$\frac{1}{3}$	$\frac{5}{3}$	

There are no negative values in the z-row, all decision variables assume integer values, and feasibility has been regained, indicating that the tableau has resulted in an optimal solution to the IP problem instance. The optimal solution is $x_1 = 5$ and $x_2 = 0$, which yields an objective function value of $z = 40$. This optimal solution corresponds to point B in Figure 2.8. Note that if the first cut does not produce an optimal solution, more cuts are inserted until an optimal tableau emerges.

Although the cutting plane method returns an optimal solution, it has several drawbacks. Firstly, the integral solution does not appear until the very last step, and if the algorithm is stopped before the end, no integral solution will be provided (unlike the branch-and-bound method which returns feasible solutions through iteratively solving different subproblems). Furthermore, since fractional parts of the coefficients are used to generate the cutting planes, rounding errors may cause the method to converge slowly [32]. Another drawback of the cutting plane method is that the algorithm may involve very lengthy computations. These problems and difficulties presented by the cutting plane method may be avoided by employing the branch-and-cut method discussed next in §2.2.4.

2.2.4 The branch-and-cut method

The branch-and-cut method for solving IP problem instances was first introduced in 1991 by Padberg and Rinaldi [46]. The algorithm was applied to the TSP to propose the notion of generating cutting planes at the nodes of a branch-and-bound tree. The branch-and-cut method proves to be a robust procedure, and is significantly more efficient than both the branch-and-bound and cutting plane methods discussed previously [6]. As the name implies, the branch-and-cut method is a combination of the branch-and-bound method and the cutting plane method. The combined efforts of the two approaches has the ability to solve far larger instances of IP problems to optimality than other methods [38]. The branch-and-cut method is generally able to obtain an optimal solution to an IP problem instance while producing a smaller branch-and-bound tree.

The first step, as with both of the previous aforementioned methods, is to solve the LP relaxation using the simplex algorithm. An important decision that should be addressed is deciding on whether to proceed by adding a cutting plane to the current node or to rather branch on a decision variable. The computational cost of inserting cutting planes may be prohibitively expensive. As a result, it is common practice not to insert cutting planes at every node of the tree. Multiple alternative approaches exist, such as inserting a cutting plane at every eighth node, or at every node at a depth of a multiple of eight in the tree [38]. An alternative approach is to generate cuts only to tighten the initial LP relaxation, and cutting planes are added only at the root node of the tree. Generally, the branch-and-cut method branches on a decision variable to create multiple nodes, solves the LP relaxation at each node, generates cutting planes, and adds the constraint imposing the cutting plane to the node.

The algorithm is applied to the problem instance in (2.31)–(2.34) to demonstrate its working. Similar to the branch-and-bound method in §2.2.2, the algorithm starts off by solving the LP relaxation of the original IP problem instance by invoking the simplex algorithm. An enumeration tree depicts the different subproblems considered. As discussed in §2.2.2, the simplex algorithm may be invoked to obtain an optimal solution to the LP relaxation which corresponds to Subproblem 1 (illustrated graphically in Figure 2.9(a)). The solution to Subproblem 1 consists of the fractional variables $x_1 = \frac{15}{4}$ and $x_2 = \frac{9}{4}$, corresponding to an objective function value of $z = 41.25$. Branching is applied to the subproblem according to the LIFO rule, resulting in Subproblems 2 and 3. By following the branch-and-bound method described in §2.2.2, branching occurs on the decision variable assuming a non-integer value having the largest coefficient in the objective function. Therefore, the constraints $x_1 \geq 4$ and $x_1 \leq 3$ are imposed, resulting in Subproblems 2 and 3, respectively. In the solution to Subproblem 3, all decision variables assume integer values ($x_1 = 3$ and $x_2 = 3$) which yields a candidate solution with an objective function value of $z = 39$. As can be seen from the formulated tree in Figure 2.9(a), Subproblem 2 is considered next. Subproblem 2 is solved by invoking the simplex algorithm, resulting in the simplex tableau

		8	5	0	0	0	
		x_1	x_2	s_1	s_2	s_3	RHS
0	s_1	0	0	1	$-\frac{1}{5}$	$-\frac{4}{5}$	$\frac{1}{5}$
5	x_2	0	1	0	$\frac{1}{5}$	$\frac{9}{5}$	$\frac{9}{5}$
8	x_1	1	0	0	0	-1	4
	g	8	5	0	1	1	$\boxed{41}$.
	z	0	0	0	1	1	

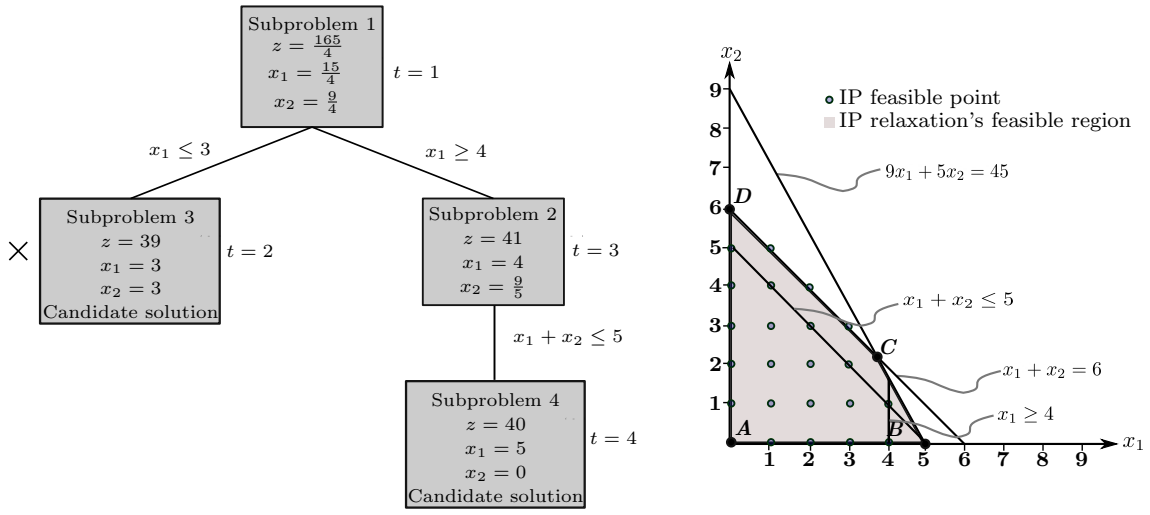
In this solution, $x_1 = 4$ and $x_2 = \frac{9}{5}$, which yields an objective function value of $z = 41$. Subproblem 2 is therefore not feasible, since it contains a non-integer decision variable. Contrary to the working of the branch-and-bound method, a Glomory fractional cut is inserted according to the methodology described in §2.2.3. The second constraint (corresponding to the equality equation $x_2 + 0.2s_2 + 1.8s_3 = 1.8$) is selected to generate the cutting plane from. The equation is rearranged so that the non-integer values are written as a summation of the integer values and a non-negative decimal portion. The equality is rearranged by writing all the terms with integer coefficients on one side, and the terms with fractional coefficients on the other, such as

$$x_2 + s_1 + 1 = 0.8 - 0.2s_2 - 0.8s_3. \tag{2.44}$$

To formulate the cut, the fractional coefficients should be less than or equal to zero, that is

$$0.8 - 0.2s_2 - 0.8s_3 \leq 0. \tag{2.45}$$

The inequality in (2.45) may be rewritten in terms of x_1 and x_2 as $x_1 + x_2 \leq 5$. The cut is applied to the decision space, as illustrated graphically in Figure 2.9(b). The cutting plane is inserted as an additional constraint into the simplex tableau to perform a single iteration of the dual simplex algorithm, in order to regain feasibility. The resulting optimal solution is $x_1 = 5$ and $x_2 = 0$, yielding an overall objective function value of $z = 40$. This solution corresponds to the integer point B in Figure 2.9(b). The solution obtained is greater than that of Subproblem 3. Since all branches are fathomed, an optimal solution has been found and the algorithm is terminated.



(a) *A branch-and-cut tree* (b) *The feasible regions of subproblems*

FIGURE 2.9: *A graphical representation of the branch-and-cut tree formulated for an IP problem instance in (a), and the corresponding feasible region in (b).*

CPLEX utilises the branch-and-cut method when solving IP and MIP problem instances [26]. CPLEX also manages the search space by invoking a tree, consisting of nodes (subproblems). A node is processed by solving the subproblem without integrality constraints, which corresponds to the relaxation of the IP problem. *Active* nodes are nodes which have not yet been solved, whereas already processed nodes are no longer considered to be active. Branching in CPLEX works similarly to branching explained in the branch-and-cut method. Branching therefore

modifies the bounds on a single variable, which will result in the creation of two new nodes from the parent node. Cuts are also formulated similarly to the branch-and-cut method explained previously, where the purpose of adding a cutting plane is to limit the size of the solution search space [26].

CPLEX starts by initialising the branch-and-cut tree to contain the root node as the only active node. Potential cuts are then generated for the root node, but not all cuts are applied to the problem immediately in order to maintain a reasonable problem size. If possible, an incumbent solution (the best known solution that satisfies all the integrality constraints) is established for comparison with other candidate solutions which might arise later in the algorithm. When processing a node, CPLEX solves the LP relaxation of the node. If the solution violates any cuts, CPLEX may add some of them, or all of them, to the node problem and solve it again. This procedure is repeated until no more violated cuts are detected. If the node becomes infeasible at any point during the addition of cuts, it is removed from the tree, otherwise, CPLEX determines whether the solution to the subproblem satisfies the integrality constraints. If this is the case, and if its objective function value exceeds that of the current incumbent solution, the solution of the subproblem is used as the new current best solution. If the solution does not satisfy the integrality constraints, branching is performed. Branching is applied to a variable that does not assume an integer value in the current solution, but should. Two new nodes are added to the tree as a result of this procedure.

After solving the LP relaxation at each node, a new objective function value is found. The node corresponding to an objective function value superior to that of all nodes at any point in the algorithm, may be compared to the incumbent solution's objective function. The process is repeated until all nodes are traversed and an optimal solution is found. The optimality gap, which is the percentage difference between the best lower bound and upper bound found throughout execution of the algorithm, is reported by CPLEX. The optimality gap is defined as an indicator of comparison between these solution values and serves as a measure of progress toward finding an optimal solution. In the situation when all nodes are fathomed, and the upper and lower bound have converged, an optimality gap of zero indicates that the incumbent solution is optimal. It is possible to instruct CPLEX to end the branch-and-cut process before optimality has been reached *via* specifying a run time limit or a memory limit, after which the best solution found throughout the search is returned [26].

2.3 The verification and validation of mathematical models

An important activity in ensuring the correctness of mathematical models and their corresponding computerised implementations, is the systematic and thorough testing of the designed system before releasing the system to users. Failure to perform this step may result in the user relying on a system that delivers outputs of questionable quality. The need for thorough systems and software testing is motivated by the necessity to verify that the system meets the design requirements specified, to build confidence in the modelling approach adopted, and to find discrepancies between the desired and observed behaviour of a system [48]. Furthermore, the developers of mathematical models and corresponding solution method implementations should critically assess the computational science and engineering tools utilised [44]. The output obtained from mathematical models should be credible and ascertain confidence in the user. Obtaining such confidence necessitates a well-planned and implemented *verification and validation* (V&V) program. The output of a verified and validated mathematical model is intended to provide a technically acceptable basis to support decisions based on the obtained results [57].

The process of ensuring that a model implementation accurately represents the developer's conceptual description of the model and the solutions to instances of the model is termed *verification* [57]. The goal of verification is to find and eliminate errors in the model by ensuring the model is implemented correctly and performs as expected. In contrast, *validation* is defined as the process of determining the extent to which a model accurately represents the real world from the perspective of the intended uses of the model [57]. The validation phase is concerned with quantifying the model's accuracy by comparing numerical solutions obtained by the model to experimental data.

It is important to note that V&V processes verify a model's accuracy and quality with regard to specific scenarios, and it does not necessarily prove that a model is correct for all possible instances (except for trivial models, which are not of interest). Furthermore, the V&V process is a continuous process that does not have a clearly defined point of completion. Since the correctness and accuracy of a model cannot be calculated for all possible instances, practical concerns such as cost limitations and the intended use complexity of the model generally govern completion or sufficiency of the V&V process. The V&V processes may, however, provide evidence that a model is sufficiently accurate [45][49][57]. Intuitively, a model may be valid for a set of experimental conditions and may be invalid for another. A model's validity is determined by it's accuracy within an acceptable range, which is dependent on the model's intended purpose and should be determined early in the developmental process of the model [50].

An adapted version of the well known *Sargent Circle*, constructed by the Society for Computer Simulation [51] in 1979, is shown in Figure 2.10. The Sargent Circle is a simplistic model used to illustrate the V&V process. The model differentiates between modelling and simulation activities (black solid lines), and assessment activities (red dashed lines). Furthermore, the Sargent Circle differentiates between three different components that comprise the model, which are the reality of interest, the mathematical model and the computer model. The *reality of interest* component represents the physical system, the particular problem, or the real-life process that is of importance and for which data are being collected. The *mathematical model* component is composed of mathematical expressions, equations, and the conceptual model which describe the reality of interest or physical system. The *computerised model* component is the implementation of the mathematical and conceptualised model or code, which may comprise, but is not limited to, the computer program, the conceptual and mathematical modelling assumptions and code inputs [57].

The *modelling* activity is concerned with extracting important features and mathematical expressions that represent the reality of interest best in the mathematical model. *Confirmation* involves assessing whether the modelling activity was correctly performed. Furthermore, the software implementation in the computer model is assessed by the *verification* activity. Identification, quantification, and reduction of errors in the computational model and its numerical solutions constitute the fundamental verification strategy [45]. Moreover, the verification activity may be divided into two activities, called *code verification* and *calculation verification* [47]. The goal of code verification is to identify and solve problems in the computer programming code, whereas calculation verification is concerned with quantifying errors introduced by the application of the code to a specific problem instance.

Experimental data form the reference against which competing model solutions are compared during the validation process [39]. Experimental data includes data from the target system, whether from historical sources or from experiments that were specifically designed, that may offer important information which may be used to validate a mathematical model's formulation. The *validation* activity aims to measure the model's accuracy by means of comparing experimental data with solution generated by the computer model. Validation is a continuous

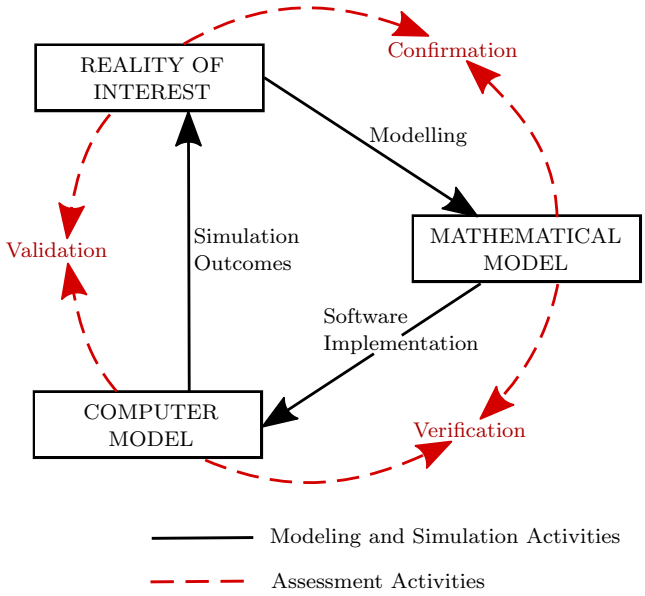


FIGURE 2.10: *The Sargent Circle used to model the V&V process (adapted from [51]).*

process that occurs as operations are improved and/or parameter ranges of input parameters are expanded. The fundamental steps in the validation process are to identify and quantify the errors and uncertainties in the conceptual and computational models, to quantify numerical errors in the computational solution, to estimate the uncertainties in the experimental data, and to compare the computational results with the experimental data [45]. Application of the validation strategy, however, does not assume that the experimental data are more accurate than the computational results, but rather, it suggests that the experimental results are a better representation of reality for the purposes of validation. A widely recommended validation method is to employ a *building-block* approach, due to the infeasibility and impracticality of conducting actual validation studies on complex or large scale systems [45]. The aforementioned approach suggests that the complex system be divided into (at least three) progressively simpler tiers, where each tier's computational results are compared with experimental data. It is recommended that the complete system be decomposed into subsystem cases, benchmark cases, and unit problems [57].

A testing process, suggested by Kendall and Kendall [29], supports this approach by recommending that systems testing should be done throughout the system's development and not subsequent to its completion. The testing process suggests that testing is performed on subsystems, modules, and unit problems as modelling progresses. Four tiers of testing is recommended, namely:

1. Program testing with test data,
2. Link testing with test data,
3. Full systems testing with test data, and
4. Full systems testing with live data.

The first step, program testing with test data, directs the programmer to desk check the algorithm or program logic before executing the program. Desk checking is executed by the developer, wherein each step in the program on paper is followed to check whether the model

logic makes sense, and the algorithm works as it is written [29]. Furthermore, both valid and invalid test data are generated and used to detect errors. Link testing with test data is used to verify whether the interdependent programs, modules, and units communicate and cooperate properly. When the individual programs and links perform satisfactorily, the system as a complete entity is tested with valid and invalid test data. Finally, the full system is tested using live data (data that have been successfully processed through the existing system). Completion of this step serves as validation, since it requires the accurate comparison of the created system's output with what is considered to be the output known to be correctly processed.

2.4 Chapter summary

A comprehensive literature review on topics relevant to this project was discussed in this chapter. The chapter opened in §2.1 with concise descriptions and exact mathematical formulations of the VRP variants related to the work done in this project. Differences between the CVRP, the VRPTW, the HFVRP, and the SDVRP were elucidated by means of a small example instance which was solved for each problem. Exact solution approaches for solving VRP instances were considered next in §2.2. A methodology for solving LP problem instances, called the Simplex Algorithm, was described. Furthermore, three solution methodologies to solving IP problem instances were considered, namely the branch-and-bound method, the cutting plane method, and the branch-and-cut method. Finally, the chapter closed in §2.3 with a brief discussion on guidelines for verifying, validating, and testing mathematical models and its computerised implementations.

CHAPTER 3

Mathematical Model

Contents

This chapter is devoted to the formulation of a computerised mathematical model in the form of a MIP problem for the MAVRP considered in this project. The MAVRP must be able to facilitate a heterogeneous fleet of delivery vehicles stationed at the depot and available to service customers, which is allowed to perform split deliveries, while adhering to the time-windows associated with customers. Furthermore, the routes assigned to delivery vehicles must adhere to a specified familiarity threshold, thereby increasing driver-route familiarity. The routes with which delivery vehicle drivers are familiar with, or the so-called master routes, must be provided as input to the model. A derivation of the model is provided in §3.1. Thereafter, the exact solution approach and the implementation thereof in CPLEX is described in §3.2, which facilitates the verification of the MAVRP. Further verification and validation of the computerised mathematical model is initiated next in §3.3. A description of a *graphical user interface* (GUI), which allows the computerised mathematical model to be utilised as a DST, is provided in §3.4. The chapter closes in §3.5 with a brief summary of its contents.

3.1 Model derivation

An MAVRP model for increased driver-route familiarity is derived in this section. First, in §3.1.1, the required input data are described in terms of the model parameters. These parameters are used to configure an instance of the MAVRP. The decision variables of the model are declared in §3.1.2, followed by a derivation and discussion on the constraints of the model in §3.1.3. The objective function is finally derived in §3.1.4.

3.1.1 Model parameters

Let $\mathcal{C} = \{1, \ldots, n\}$ index the set of customers to be serviced. The depot is included in the additional supplemented set of vertices, $\mathcal{V} = \mathcal{C} \cup \{0, n+1\}$, where 0 denotes the depot upon departure and $n+1$ denotes the depot upon return. Let $\mathcal{G}(\mathcal{V}, \mathcal{A})$ denote the directed travel graph on which the MAVRP is defined, where the arc set, denoted by $\mathcal{A}$, represents the directed road network links which delivery vehicles may traverse, and $\mathcal{V}$ denotes the vertex set. Furthermore, let $\delta^-(i)$ denote the subset of vertices from which arcs are directed towards vertex $i \in \mathcal{V} \setminus \{0\}$, and let $\delta^+(i)$ denote the subset of vertices to which arcs are directed from vertex $i \in \mathcal{V} \setminus \{n+1\}$ in the travel graph $\mathcal{G}$.

Let a_i and b_i denote the earliest and latest possible service start time (measured in minutes after some fixed reference time) allowed at customer $i \in \mathcal{C}$, respectively. The time-window associated with the depot, denoted by $[a_0, b_0]$ and $[a_{n+1}, b_{n+1}]$, represents the earliest possible departure time and the latest possible return time at the depot. The average demand volume (measured in cubic metres) exhibited by customer $i \in \mathcal{C}$ is denoted by q_i. The average service time duration at customer $i \in \mathcal{C}$ (measured in minutes), is denoted by s_i. The average service time duration is derived as a function based on the average demand volume per planning period associated with each customer, and includes a fixed service setup time. The service time associated with the depot is $s_0 = s_{n+1} = 0$.

The various types of delivery vehicles to which each delivery vehicle may belong are indexed by $\mathcal{H} = \{1, \ldots, |\mathcal{H}|\}$. Accordingly, let m_h denote the number of delivery vehicles of type $h \in \mathcal{H}$ available for service. Moreover, delivery vehicles of the same type $h \in \mathcal{H}$ are indexed successively in the set of (heterogeneous) delivery vehicles that are available at the depot to service customers, denoted by $\mathcal{K} = \{1, \ldots, |\mathcal{K}|\}$. Intuitively, it can be shown that $\sum_{h \in \mathcal{H}} m_h = |\mathcal{K}|$. Let $\mathcal{Q}_k$ and $\mathcal{F}_k$ denote the loading capacity (measured in cubic metres) and fixed cost (measured in Rand) associated with utilising delivery vehicle $k \in \mathcal{K}$, respectively. To account for delivery vehicle size restrictions which may prohibit the visitation of certain delivery vehicles at certain customers, the binary parameter

$$g_{ik} = \begin{cases} 1 & \text{if delivery vehicle } k \in \mathcal{K} \text{ is able to visit customer } i \in \mathcal{C}, \\ 0 & \text{otherwise,} \end{cases}$$

is defined. Different types of delivery vehicle may have different travel costs and travel times associated with them. Let c_{ijk} and t_{ijk} denote the cost (in Rand) and expected travel time (in minutes), respectively, associated with delivery vehicle $k \in \mathcal{K}$ travelling from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$. These travel costs and travel times associated with each delivery vehicle $k \in \mathcal{K}$ may be determined by multiplying the travel cost matrix and travel time matrix with a variable cost factor v_k and speed factor σ_k, respectively. Furthermore, let α_k denote the cost of increasing the travel duration of delivery vehicle $k \in \mathcal{K}$ by one minute.

The model takes as input a set of master route arcs which represents a set of arcs with which delivery vehicle drivers are assumed to be familiar with. A user-specified familiarity threshold, denoted by β, represents the minimum percentage of distance that must be travelled on these master routes. The set of master route arcs are represented by the binary parameter

$$m_{ij} = \begin{cases} 1 & \text{if the arc } (i, j) \text{ forms part of the set of master route arcs,} \\ 0 & \text{otherwise.} \end{cases}$$

3.1.2 Model variables

The binary decision variables

$$x_{ijk} = \begin{cases} 1 & \text{if delivery vehicle } k \in \mathcal{K} \text{ travels from vertex } i \in \mathcal{V} \setminus \{n+1\} \text{ to vertex } j \in \mathcal{V} \setminus \{0\}, \\ 0 & \text{otherwise,} \end{cases}$$

capture all vehicle flows and are stored in row i and column j of slice k in a three-dimensional $(n+1) \times (n+1) \times |\mathcal{K}|$ flow matrix $\boldsymbol{X}$. The decision variable y_{ik} records the volume (in cubic metres) of goods delivered to customer i by delivery vehicle k. Furthermore, define the binary decision variable

$$r_k = \begin{cases} 1 & \text{if delivery vehicle } k \in \mathcal{K} \text{ is used,} \\ 0 & \text{otherwise.} \end{cases}$$

The continuous decision variable $\mathcal{T}_{ik}$ is defined as the time (measured in minutes since some reference time) that delivery commences at vertex $i \in \mathcal{V}$, by delivery vehicle $k \in \mathcal{K}$. Intuitively, $\mathcal{T}_{0k}$ denotes a delivery vehicle's departure time from the depot and $\mathcal{T}_{n+1,k}$ denotes the time that the delivery vehicle will return to the depot upon completing its assigned route. A delivery vehicle's total time in spent away from the depot (if utilised) may therefore be calculated as $\mathcal{T}_{n+1,k} - \mathcal{T}_{0k}$.

3.1.3 Model constraints

A number of constraints are imposed on the model to guarantee that solutions are practically feasible. The constraints are formulated based on those imposed in the CVRP, the VRPTW, the HFVRP, and the SDVRP, discussed in §2.1. To ensure that each customer is visited at least once, the constraint set

$$\sum_{k \in \mathcal{K}} \sum_{j \in \delta^+(i)} x_{ijk} \geq 1, \quad i \in \mathcal{C}, \tag{3.1}$$

is imposed. The constraint sets

$$\sum_{j \in \delta^+(0)} x_{0jk} = r_k, \quad k \in \mathcal{K}, \tag{3.2}$$

$$\sum_{i \in \delta^-(n+1)} x_{i,n+1,k} = r_k, \quad k \in \mathcal{K}, \tag{3.3}$$

are imposed to ensure that each delivery vehicle utilised departs from the depot and returns to the depot after completing its assigned route. In order to ensure that each delivery vehicle departs from a customer if it arrives at that customer, the constraint set

$$\sum_{i \in \delta^-(j)} x_{ijk} - \sum_{i \in \delta^+(j)} x_{jik} = 0, \quad j \in \mathcal{V}, \ k \in \mathcal{K}, \tag{3.4}$$

is imposed. The constraint set

$$\sum_{j \in \delta^+(i)} x_{ijk} \leq g_{ik}, \quad i \in C, \ k \in \mathcal{K}, \tag{3.5}$$

ensures that customers are only serviced by delivery vehicles that are, in fact, able to visit them, based on size restrictions. Delivery vehicles must adhere to the time-windows that are associated

with customers. The constraint sets

$$\begin{aligned} \mathcal{T}_{ik} + s_i + t_{ijk} - \mathcal{T}_{jk} &\leq (1 - x_{ijk})M_{ijk}, && k \in \mathcal{K},\ (i,j) \in \mathcal{A}, && (3.6)\\ a_i \sum_{j \in \delta^+(i)} x_{ijk} \leq\ \mathcal{T}_{ik} &\leq b_i \sum_{j \in \delta^+(i)} x_{ijk}, && k \in \mathcal{K},\ i \in \mathcal{V}\backslash\{n+1\}, && (3.7)\\ a_{n+1} \sum_{i \in \delta^-(n+1)} x_{i,n+1,k} \leq\ \mathcal{T}_{n+1,k} &\leq b_{n+1} \sum_{j \in \delta^-(n+1)} x_{j,n+1,k}, && k \in \mathcal{K}, && (3.8) \end{aligned}$$

are therefore imposed. The constraint set in (3.6) implicitly prevents subtour formation, where M_{ij} denotes a large constant which may be set to $\max\{b_i + s_i + t_{ijk} - a_j, 0\}$, as well as ensures correct bookkeeping of the time that delivery vehicles commence at vertices. Imposition of the constraint set in (3.7) enforces that the service at each customer starts within its specified time-window. The constraint set in (3.8) ensures that each utilised delivery vehicle departs from and arrives back at the depot within its specified time-window. In order to ensure that the capacity of any delivery vehicle is not exceeded, the commodity flow constraint sets

$$\begin{aligned} \sum_{k \in \mathcal{K}} y_{ik} &= q_i, \qquad i \in \mathcal{C}, && (3.9)\\ \sum_{i \in \mathcal{C}} y_{ik} &\leq Q_k r_k, \quad k \in \mathcal{K}, && (3.10) \end{aligned}$$

are imposed. The constraint set in (3.9) ensures that the total volume of commodities delivered to a customer corresponds to the demand exhibited by the customer. Moreover, the constraint in (3.10) restricts the volume of commodities carried by a utilised delivery vehicle to not exceed its capacity. Furthermore, imposition of the constraint set

$$\sum_{(i,j) \in \mathcal{A}} x_{ijk} \leq (n+1) r_k, \quad k \in \mathcal{K}, \tag{3.11}$$

ensures that a vehicle is considered to be utilised if it services at least one customer. To ensure that a delivery only takes place at a customer if a delivery vehicle actually visits that customer, the constraint set

$$y_{jk} \leq \mathcal{O}_k \sum_{i \in \delta^-(j)} x_{ijk}, \quad j \in \mathcal{C}, \quad k \in \mathcal{K}, \tag{3.12}$$

is imposed, where $\mathcal{O}_k$ is a large integer (at least as large as the largest delivery vehicle type's capacity Q_k). Furthermore, the constraint set

$$\beta \leq \frac{\sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} x_{ijk} m_{ij} d_{ij}}{\sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} x_{ijk} d_{ij}}, \tag{3.13}$$

is imposed to ensure that the portion of the total distance which is travelled on the master route arcs is more than the specified familiarity threshold β. To enforce the binary and real-valued nature of the decision variables, the domain constraints

$$\begin{aligned} x_{ijk} &\in \{0,1\}, && (i,j) \in \mathcal{A},\ k \in \mathcal{K}, && (3.14)\\ r_k &\in \{0,1\}, && k \in \mathcal{K}, && (3.15)\\ \mathcal{T}_{ik} &\geq 0, && i \in \mathcal{V},\ k \in \mathcal{K}, && (3.16)\\ y_{ik} &\geq 0, && i \in \mathcal{C},\ k \in \mathcal{K}, && (3.17) \end{aligned}$$

are imposed.

3.1.4 Objective function

The model objective is to

$$\text{minimise } z = \sum_{k \in \mathcal{K}} r_k F_k + \sum_{k \in \mathcal{K}} \sum_{i,j \in \mathcal{A}} c_{ijk} x_{ijk} + \sum_{k \in \mathcal{K}} \alpha_k (T_{n+1,k} - T_{0,k}). \tag{3.18}$$

The first term represents the fixed cost associated with utilising a delivery vehicle, the second term represents the variable cost associated with the arcs traversed, whereas the third term represents the total duration of each delivery vehicle's trip.

The first term only makes a contribution to the objective function value if the delivery vehicle is in fact utilised (*i.e.* if $r_k = 1$). In the second term, a contribution is made to the objective function value only if delivery vehicle k travels from vertex i to vertex j (*i.e.* if $x_{ijk} = 1$). As previously mentioned in §3.1.2, a delivery vehicle's total travel time (if utilised) may be calculated as $\mathcal{T}_{n+1,k} - \mathcal{T}_{0k}$. In the third term the trip duration of each delivery vehicle is multiplied with its cost coefficient α_k in order to compact routes.

3.2 Model implementation

This section is devoted to an implementation of the MAVRP derived in §3.1 in CPLEX *via* its Python programming language interface for verification and validation purposes. As discussed in §2.2.4, CPLEX applies the branch-and-cut method to solve MIP problem instances. The model implementation comprises three sections.

In the first section inputs specified by the user are stored as a set of parameters to the model. These parameters configure the model instance and may be imported *via* an *Microsoft Office Excel* (Excel) file comprising multiple Excel spreadsheets. The data in the Excel file should be organised according to eight different spreadsheets. The first spreadsheet contains general information about the problem instance, including the number of customers, the variable service rate and the fixed service rate for deliveries. The second spreadsheet lists the longitude and latitude coordinates of the depot and its customers. Furthermore, the third spreadsheet contains the vehicle-customer compatibility lists. If a customer may not be visited by a certain delivery vehicle type, the delivery vehicle type is not included in the list of compatible delivery vehicle types for that specific customer. The fourth spreadsheet contains the time-window associated with each customer, whereas the fifth spreadsheet contains the demand volume specified by each customer. The sixth and seventh spreadsheets contain matrices of travel distances and expected travel times between the customers and the depot, respectively. Details about the available fleet and each type of delivery vehicle is captured in the final spreadsheet.

In the second section of the model implementation, the master route arcs are computed for a given depot and its customers. King *et al.* [30] proposed a master route generator for computing suitable routes for delivery vehicle drivers to become familiar with in the context of a given depot and its customers. This master route generator was made available to the author to generate master route arcs to be given as input to the MAVRP model. Any set of master route arcs, however, may be provided as input to the model.

The final section of the model implementation comprises the MAVRP model derived in §3.1. In this section, the model is encoded in CPLEX, accessed *via* its Python interface. The implementation of the model derived in (3.1)–(3.18) is shown in Listing 3.1. The decision variables are defined first, followed by the declaration of the objective function, and then the implementation

of the constraints. The final two lines of the model implementation provide the command for the problem instance to be solved, and then prints the solution returned by CPLEX.

```
mdl = Model("MAVRP")
# Decision variable declaration
# Equations (3.14)-(3.17)
keys_x = [(i,j,k) for (i,j) in A for k in vehicles]
x = mdl.binary_var_dict(keys_x, name = 'x')
keys_y = [(i,k) for i in C for k in vehicles]
y = mdl.continuous_var_dict(keys_y, lb = 0, name = 'y')
keys_r = [k for k in vehicles]
r = mdl.binary_var_dict(keys_r, name = 'r')
keys_T = [(i,k) for i in V for k in vehicles]
T = mdl.continuous_var_dict(keys_T, lb = 0, name = 'T')

# Equation (3.18) Objective function
mdl.minimize(mdl.sum(r[k]*F[k] for k in vehicles)+mdl.sum(c[i,j,k]*x[i,j,k]
  for i,j in A for k in vehicles)+mdl.sum(alpha[k]*(T[n+1,k]-T[0,k]) for k in
  vehicles))

# Constraints
# Equation (3.1)
mdl.add_constraints(mdl.sum(x[i,j,k] for j in delta_plus[i] for k in vehicles)
   >= 1 for i in C)
# Equation (3.2)
mdl.add_constraints(mdl.sum(x[0,j,k] for j in delta_plus[0]) == r[k] for k in
  vehicles)
# Equation (3.3)
mdl.add_constraints(mdl.sum(x[i,n+1,k] for i in delta_minus[n+1]) == r[k] for
  k in vehicles)
# Equation (3.4)
mdl.add_constraints(mdl.sum(x[i,j,k] for i in delta_minus[j]) - mdl.sum(x[j,i,
  k] for i in delta_plus[j]) == 0 for j in C for k in vehicles)
# Equation (3.5)
mdl.add_constraints(mdl.sum(x[i,j,k] for j in delta_plus[i]) <= g[i,k] for i
  in C for k in vehicles)
# Equation (3.6)
mdl.add_constraints(T[i,k] + s[i] + t[i,j,k] - T[j,k] <= M[i,j,k]*(1-x[i,j,k])
   for i,j in A for k in vehicles)
# Equation (3.7)
mdl.add_constraints(a[i]*mdl.sum(x[i,j,k] for j in delta_plus[i]) <= T[i,k]
  for k in vehicles for i in V if i!=n+1)
mdl.add_constraints(T[i,k] <= b[i]*mdl.sum(x[i,j,k] for j in delta_plus[i])
  for k in vehicles for i in V if i!=n+1)
# Equation (3.8)
mdl.add_constraints(a[n+1]*mdl.sum(x[i,n+1,k] for i in delta_minus[n+1]) <= T[
  n+1,k] for k in vehicles)
mdl.add_constraints(T[n+1,k] <= b[n+1]*mdl.sum(x[j,n+1,k] for j in delta_minus
  [n+1]) for k in vehicles)
# Equation (3.9)
mdl.add_constraints(mdl.sum(y[i,k] for k in vehicles) == q[i] for i in C)
# Equation (3.10)
mdl.add_constraints(mdl.sum(y[i,k] for i in C) <= Q[k]*r[k] for k in vehicles)
# Equation (3.11)
mdl.add_constraints(mdl.sum(x[i,j,k] for i,j in A) <= (n+1)*r[k] for k in
  vehicles)
# Equation (3.12)
mdl.add_constraints(y[j,k] <= O[k]*mdl.sum(x[i,j,k] for i in delta_minus[j])
  for j in C for k in vehicles)
# Equation (3.13)
```

```
44  mdl.add_constraint(beta*(mdl.sum(x[i,j,k]*d[i,j] for i,j in A for k in
      vehicles)) <= mdl.sum(x[i,j,k]*m[i,j]*d[i,j] for i,j in A for k in vehicles)
      )
45  solution = mdl.solve(log_output = True)
46  print(solution)
```

Listing 3.1: An implementation of the MAVRP model derived in (3.1)–(3.18) in CPLEX *via* its Python interface.

The remainder of this section is dedicated towards describing a small hypothetical problem instance containing ten customers to illustrate the input parameters and output of the MAVRP model derived in §3.1. A random problem instance may be generated by randomly assigning demand volumes and locations (in the form of coordinates) to customers. Information about the fleet of available delivery vehicles stationed at the depot is listed in Table 3.1. Eight delivery vehicles are available, comprising two small delivery vehicles, three medium delivery vehicles, and three large delivery vehicles which may be utilised. Details about each type of delivery vehicle is stipulated in the rows accordingly. The input data corresponding to the customers and the depot is shown in Table 3.2. The horizontal and vertical coordinates are used to calculate the Euclidean distances between the vertices, which represent the travel distances between vertices. Multiplication of these distances by the speed parameter of the relevant delivery vehicle results in a travel time matrix calculated for each delivery vehicle. The delivery vehicle compatibility depicts which delivery vehicles are able to service a specific customer, in order to account for possible size restrictions at customers. Furthermore, each customer's time-window start and end times are indicated in minutes since 08:00. The demand volume exhibited by each customer in cubic metres is listed in the final column.

TABLE 3.1: *Input data related to the available fleet of delivery vehicles in a small hypothetical problem instance.*

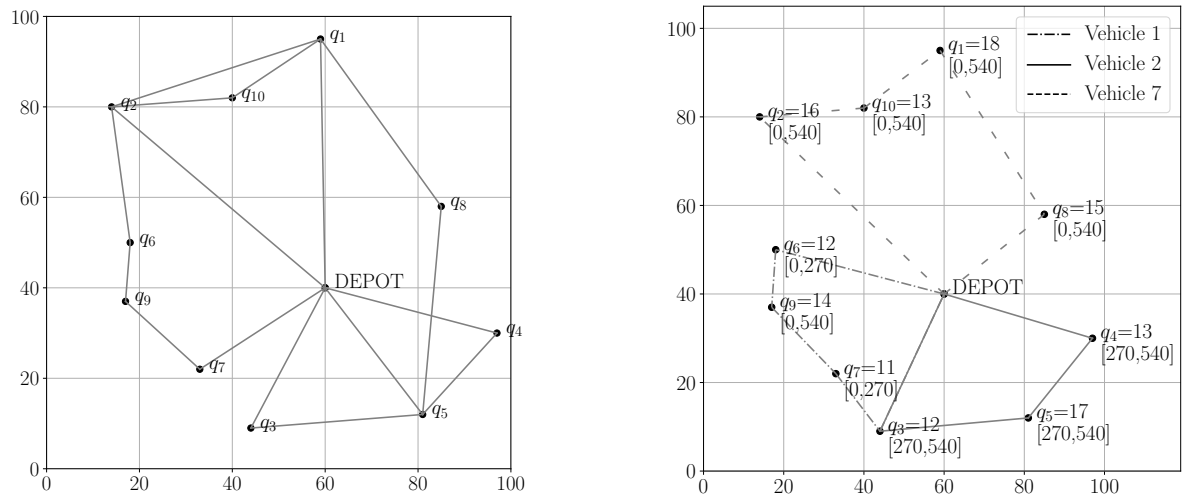
Type, k	Number of vehicles, m_h	Fixed cost, $\mathcal{F}_k$	Capacity, $\mathcal{Q}_k$	Variable cost, v_k	Speed factor, σ_k
Small	2	3 000	40	5.850	0.900
Medium	3	3 250	55	6.175	1.045
Large	3	3 500	65	6.500	1.200

The master routes computed for the small hypothetical problem instance by invoking the master route generator proposed by King *et al.* [30] are illustrated graphically in Figure 3.1(a). A corresponding optimal solution based on a minimum familiarity threshold of 60% is shown in Figure 3.1(b). The model constraints may be validated by recording the model's decision variables in tabular format, as shown in Table 3.3. The sequence of customers to be visited, the service start times at these customers, and the volume of commodities delivered to each customer for each utilised delivery vehicle are shown for validation purposes. In the solution returned, only three delivery vehicles are utilised. Two small delivery vehicles were utilised, whereas the other is a larger delivery vehicle. Upon analysis of the solution returned, it is clear that each customer's demand is satisfied and no delivery vehicle's capacity is exceeded. Furthermore, the service at each customer starts within its specified time-window. Customer 3 is visited twice, which serves as an example of a split delivery taking place. The actual familiarity percentage obtained by the solution is 72.04%, which is more than the minimum threshold of 60% that was specified. Furthermore, the objective function value and the values of its comprised terms (in Rand) of the solution returned are summarised in Table 3.4. The objective function value of the returned solution is R13 583.00. Upon inspection of these output values, it is confirmed that

the solution returned by CPLEX satisfies every constraint of the model derived in §3.1, and is therefore feasible.

TABLE 3.2: *Input data related to the small hypothetical problem instance. For each vertex, the coordinates, vehicle-compatibilities, time-window start and end times (measured in minutes since the start of the day), and demand volumes in cubic metres are tabulated.*

Vertex, i	Coordinates	Vehicle compatibility, g_{ik}	Time-window, $[a_i, b_i]$	Demand, q_i
0	(60,40)	N/A	[0, 600]	N/A
1	(59,95)	3	[0, 540]	18
2	(14,80)	1,2,3	[0, 540]	16
3	(44,9)	1	[270, 540]	12
4	(97,30)	1,2,3	[270, 540]	13
5	(81,12)	1,2,3	[270, 540]	17
6	(18,50)	1,2,3	[0, 270]	12
7	(33,22)	1,2,3	[0, 270]	11
8	(85,58)	2,3	[0, 540]	15
9	(17,37)	1,2,3	[0, 540]	14
10	(40,82)	1,2,3	[0, 540]	13



(a) *Master routes* (b) *An optimal solution*

FIGURE 3.1: *An example problem instance consisting of ten customers and a single depot. The master routes are shown on the left in (a), and an optimal solution to the corresponding problem instance, based on a minimum familiarity threshold of 60%, is shown on the right-hand side in (b). The volume of demand exhibited by each customer in cubic metres is indicated next to its vertex, as well as the time-window associated with each customer underneath its vertex.*

3.3 Model verification

Similar to the verification strategy followed for the small hypothetical problem instance in §3.2, the implementation of the MAVRP derived in §3.1 is verified in this section by following the same process

TABLE 3.3: *An optimal solution to the example problem instance. The delivery vehicle type, its corresponding index, delivery vehicle visitation sequence, volume of demand delivered, service start times, and arrival times back at the depot are given.*

Vehicle type	Index, k	From i	To j	Volume delivered, y_{ik}	Start time, T_{ik}	Return time, T_{jk}
Small	1	0	6	12	43.344	
		6	9	14	82.200	
		9	7	11	139.935	
		7	3	2	211.674	
		3	0	0	270.000	347.397
Small	2	0	4	13	235.505	
		4	5	17	270.000	
		5	3	10	340.675	
		3	0	0	435.084	512.481
Large	7	0	2	16	0.000	
		2	10	13	73.151	
		10	1	18	162.443	
		1	8	15	239.069	
		8	0	0	357.335	449.302

TABLE 3.4: *The objective function value and its comprising terms of the solution returned for the small hypothetical problem instance in Rand.*

Fixed cost	R9 500.00
Variable cost	R2 756.74
Service cost	R1 326.26
Total cost	R13 583.00

for multiple randomly generated problem instances. Model verification includes the assessment of the implementation of the computer model, as discussed in §2.3. The verification activity was initiated by performing code verification, in which any problems encountered throughout the model's development were addressed. This included technical errors made, errors in the input data, and data formatting issues. Once these problems were addressed and the implementation of the model in CPLEX was able to return solutions successfully, the calculation verification was completed.

Further model verification and validation is performed in this section by following the testing methodology suggested by Kendall and Kendall [29], which is described in §2.3. The model implementation is invoked to solve randomly generated test instances. The test instances contain both valid and invalid data to detect errors or unusual behaviour. Once an infeasible problem instance is incurred, the problem instance is discarded and the next problem instance is considered. The input parameters for instances of the MAVRP derived in §3.1 are randomised according to the following rules:

- A dataset number is provided as the seed number to ensure that each test instance can be replicated in the future.
- The number of customers in each problem instance is assigned an integer-valued number from eight to twelve. This range was chosen so as to limit the solution duration of each instance, while not over simplifying the instance.
- The coordinates of customers and the depot are drawn from a uniform distribution with values ranging from zero to a hundred in the Cartesian plane.
- Customers having an even numbered index are assigned a morning time-window [0,270], whereas the remainder of the customers are assigned a latter time-window [270,540]. The time-window associated with the depot is [0,600].

- The three types of delivery vehicles described in Table 3.1 are stationed at the depot and available to service customers.
- Each customer is allowed to be serviced by any type of delivery vehicle (to simplify the model for testing purposes).
- The master routes provided as input to the model are generated by invoking the master route generator proposed by King *et al.* [30]. A minimum familiarity threshold of 40% is specified for each instance.

Five problem instances were generated for each number of customers, $n = 8, \ldots, 12$, resulting in a total of $5 \times 5 = 25$ testing problem instances. For each problem instance, the objective function value, required run time, and resulting optimality gap obtained when invoking CPLEX are summarised in Table 3.5. In order to confirm the feasibility of each solution, the following was validated:

- Each delivery vehicle's assigned route departs from and returns to the depot.
- The sum of all the delivery quantities assigned to a delivery vehicle does not exceed its capacity.
- The cumulative volume of commodities delivered to a customer is equal to the demand exhibited by that customer.
- The volume of commodities transported by a delivery vehicle decreases according to the corresponding delivered volume as deliveries are made to each customer along its route.
- The service at each customer starts within its specified time-window.
- The service start time at each customer is later than the departure time from the depot if it is the first customer along the route, or later than the service start time at the previous customer visited by the corresponding delivery vehicle.

All solutions returned for the testing problem instances satisfied the constraints and were validated according to the above mentioned criteria. The testing problem instances also provide some valuable insight. The run time of the model is clearly dependent on the problem instance size, and an increasing number of customers results in an increase in run time. If the number of customers in a VRP instance is increased, the number of possible arcs which may form part of a feasible solution also increases. The mathematical formulation is verified through the successful implementation thereof in CPLEX.

3.4 Decision support tool

This section is devoted to a detailed description of a DST that encapsulates the MAVRP derived in §3.1 and its exact solution approach implemented in §3.2. The DST is made accessible to users by developing a user-friendly *graphical user interface* (GUI) for users to interact with. The GUI was developed in Python by invoking the *Tkinter* package. Tkinter is a framework that is built into the Python standard library, resulting in several advantages, such as the development of cross-platform GUI frameworks that use visual elements from native operating system elements to blend in with the platform on which it is being operated [1]. The GUI serves as a toolkit within the DST for users with varying technical skills to employ the computerised mathematical model proposed in this project to their use-cases. Four functional requirements that are fulfilled by the DST is that (1) it is able to import a given dataset from an Excel file to generate a problem instance, (2) it is capable of generating a set of high-quality routing solutions from the problem instance provided, (3) it has the ability to display and export the routing solutions computed, and (4) it is able to generate and solve random problem instances to demonstrate the capabilities of the MAVRP model.

The GUI is comprised of two main pages, namely the *Import Instance* page, which is shown in Figure 3.2, and the *Random Instance* page shown in Figure 3.3. Both pages consist of a standard layout, which consists of a *Navigation panel*, an *Input panel* and a *Solution panel*. The Import Instance page is displayed when the tool is launched and the frame on the left-hand side (the Navigation panel) enables navigation between the Import Instance or Random Instance page, or enables the user to terminate the program. The

TABLE 3.5: *The instance number, number of customers (n), seed number, objective function value (z) in Rands, run time in seconds, and remaining optimality gap returned for each testing instance solved.*

Instance	n	Seed number	z	Time	Gap
1	8	1	10 714.88	6.33	0.00
2	8	2	12 755.41	13.81	0.00
3	8	3	15 915.22	2.59	0.00
4	8	4	13 961.90	4.63	0.00
5	8	5	10 267.45	2.44	0.00
6	9	1	13 673.87	62.16	0.00
7	9	2	13 480.13	38.00	0.00
8	9	3	13 495.88	237.44	0.00
9	9	4	14 276.97	157.34	0.00
10	9	5	14 538.24	34.52	0.00
11	10	1	14 890.86	66.55	0.00
12	10	2	13 466.96	53.56	0.00
13	10	3	14 557.02	128.99	0.00
14	10	4	16 590.04	156.41	0.00
15	10	5	13 885.57	49.31	0.00
16	11	1	17 114.26	6 178.69	0.00
17	11	2	14 205.28	259.52	0.00
18	11	3	17 072.29	9 540.97	0.00
19	11	4	17 291.26	754.52	0.00
20	11	5	16 028.46	99.69	0.00
21	12	1	17 849.16	6 786.67	0.00
22	12	2	17 083.68	39 491.13	0.00
23	12	3	17 662.69	3 974.59	0.00
24	12	4	19 402.31	2 249.08	0.00
25	12	5	17 780.68	1 141.75	0.00

DST allows for the importation of previously formulated master routes, or it is able to generate completely new master routes by invoking the master route generator proposed by King *et al.* [30]. Furthermore, the DST invokes the computerised mathematical model (which employs CPLEX) to compute high-quality routing solutions. The remainder of this section is devoted to a high-level overview of how the user may interact with the DST through its GUI by providing inputs.

The Import Instance page enables the user to generate a new set of routing solutions based on a problem instance they have specified as an Excel file, with each sheet arranged according to the standard format described in §3.2. The input panel (the frame on the bottom right-hand side), allows the user to specify additional parameters to the DST, such as the familiarity threshold and run time. The first input textbox serves two purposes: it allows the user to specify the name of a previously formulated set of master route arcs which may then be imported, or to specify the name of an Excel file containing information about the instance, based on which the master routes should be generated. If the user generates a new set of master routes, the master routes are automatically stored within the current working directory and the user must load the file before executing the problem instance. The user must next specify the familiarity threshold and solution run time. Both of these parameters have default values that may be used. The final textbox requires the user to specify the problem instance issued from a location on their computer, which is in the standard Excel file format. Once the execution button is selected, a loading bar appears while routes are being generated in the background. Following successful execution, the user can access the routes and information about the solution in the Solution panel. This includes a basic visualisation of the top view of the transportation network, the computed routes, the solution run time, the obtained familiarity percentage, the resulting optimality gap, and the objective function values. The display of a solution

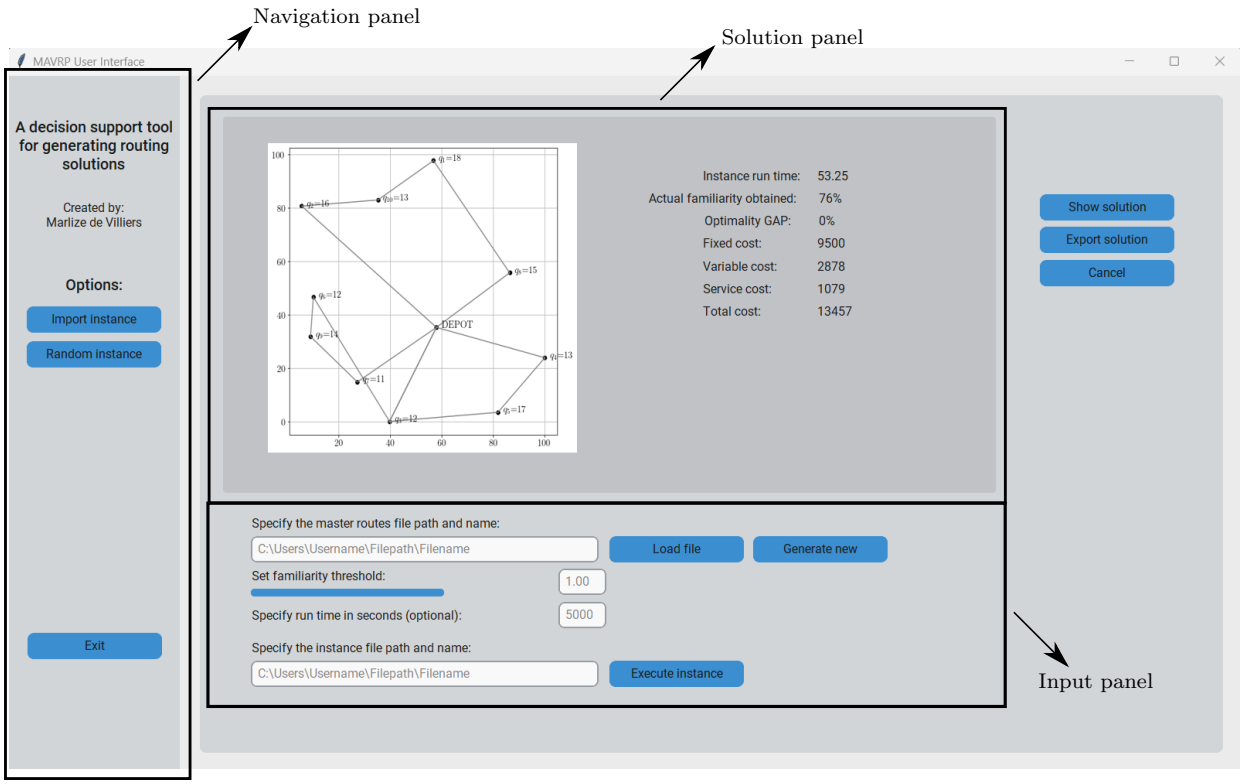


FIGURE 3.2: *The Import Instance page illustrating a solution obtained by solving an example problem instance.*

returned for a small problem instance is depicted in Figure 3.2. Finally, the user may export the solution to an Excel file if they would like to save the solution and its associated decision variables.

The Random Instance page is dedicated towards generating and solving a random problem instance in order to experiment with the MAVRP proposed in this project. In the Input panel, the user may adapt and specify some of the parameters to the random problem instance. The first input textbox requires the user to provide a name for the problem instance specified. If the *Export to Excel file* option is selected, the exported file containing the routing solutions and decision variables is named after the problem instance's specified name and stored within the current working directory. In contrast to the Import page previously discussed, the user able to specify whether they would like to include or exclude time-windows. When time-windows are included, random time-windows are assigned to customers, whereas if it is excluded, all customers assume a large time-window. After the required parameters have been provided, the problem instance may be solved and a loading bar appears in order to indicate that the problem is being solved. The execution includes the calculation of new master routes, based on the master generator proposed by King *et al.* [30], which is then provided as an input to the MAVRP model and solved using CPLEX. Once the instance is solved, the loading bar disappears and the user is able to visualise the solution returned by selecting the *Show solution* button. This process may be iterated by clearing the solution and generating a new problem instance by specifying different input parameters.

The verification process of the tool includes determining whether the DST is implemented correctly. To ensure the DST's credibility and reliability, the development of the tool was documented using commented sections within the implementation code as well as the descriptions and diagrams presented in this chapter. This also allows future users to improve or expand on the functionality of the DST. Logic errors are often difficult to detect, which is why the testing procedure, which was previously discussed in §2.3, was invoked. The DST was successfully verified after performing program testing with test data, link testing with test data, and full systems testing with test data.

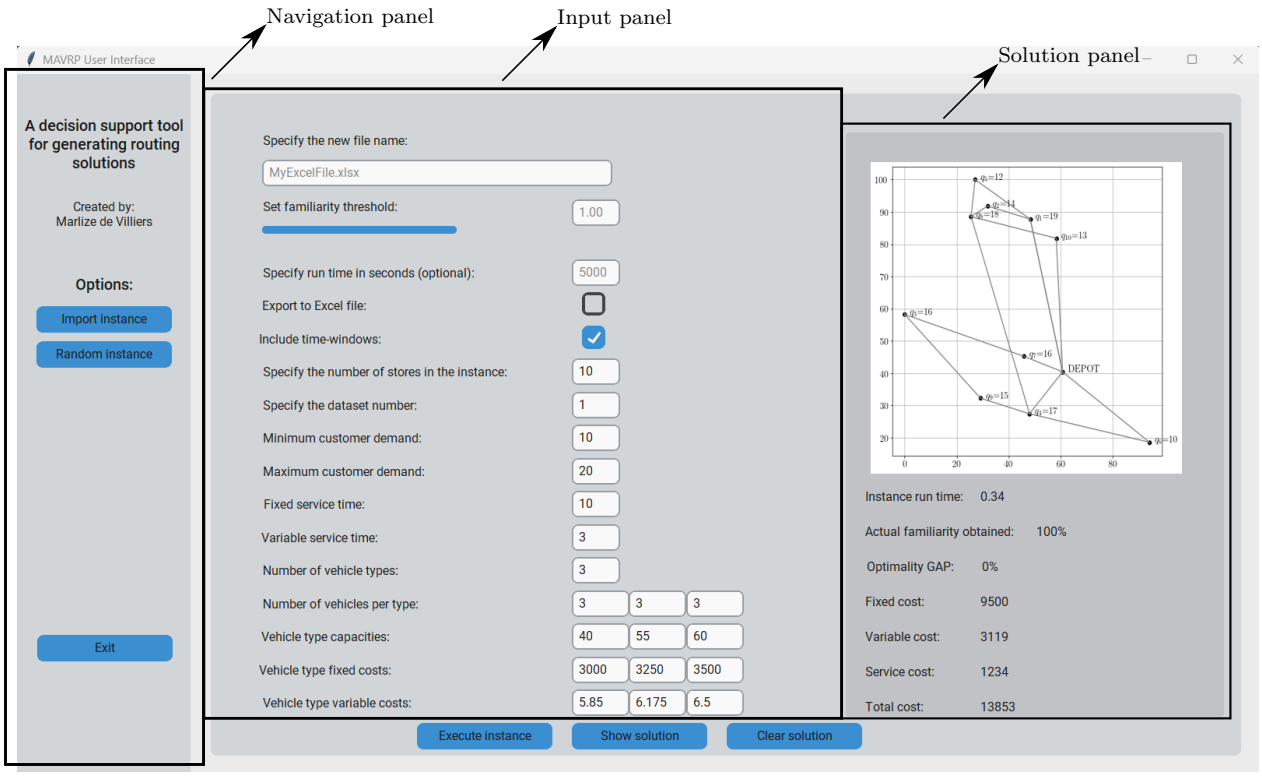


FIGURE 3.3: *The Random Instance page illustrating a solution to a randomly generated example problem instance.*

3.5 Chapter summary

In this chapter, a mathematical model was derived for increased driver-route familiarity. The model's parameters, decision variables, constraints, and objective function were discussed in §3.1. A computer implementation of the model in CPLEX served as a verification of the mathematical model. The validation of an example problem instance was performed in §3.2 for further verification purposes of the computerised mathematical model and solution implementation. The model was also successfully verified based on the solutions returned for 25 testing problem instances in §3.3. A DST was proposed in §3.4, which acts as a user-friendly tool for users to invoke the MAVRP model.

CHAPTER 4

Case Study

Contents

This chapter is devoted to the execution of a case study by providing real-world data as input parameters to the DST developed in Chapter 3, in an effort to showcase its practical applicability. The input data include historic demand data of stores, provided by the industry partner attached to this project. The execution of this case study also attempts to confirm that the computerised mathematical model accurately addresses the real-world problem that it is attempting to model, and serves as a final validation in accordance with the validation strategy described in §2.3. To provide context for the data employed in the case study, a discussion on the background of the industry partner associated with this project is presented in §4.1. The case study is conducted in §4.2 by providing the data obtained from the industry partner as input to the DST. A detailed discussion on the results and the interpretation thereof is presented in §4.3, whereafter the chapter closes in §4.4 with a brief summary of its contents.

4.1 Industry partner background

The industry partner attached to this project is a large South African clothing retailer that owns 5 470 stores located across ten African countries. The industry partner reports an annual revenue of R77.3 billion and the competitiveness of its supply chain is largely dependent on its ability to distribute retail commodities from its depots to its stores efficiently. The industry partner reported numerous challenges when implementing the solutions stemming from invoking standard commercial vehicle routing and scheduling software for the computation of routing schedules for delivery vehicles. When daily delivery routes differ significantly from one schedule to the next, drivers tend to get lost or travel on roads that are not suitable for the delivery vehicles [54]. These problems often lead to a degradation in the supply chain operational efficiency, which may ultimately increase the costs of activities across the entire supply chain. Furthermore, unanticipated increase in travel times may result in delivery vehicle drivers missing the time-windows of certain stores, which may consequently cause the next delivery to also fall behind schedule, ultimately rendering the rest of the planned schedule infeasible. As a result of these practical issues with the implementation of planned delivery routes, the industry partner is interested in increasing driver-route familiarity in their planned delivery routes. Delivery vehicle drivers should be afforded the opportunity to become familiar with the routes along which they travel, which is anticipated to increase the efficiency with which they perform deliveries.

Each store of the industry partner is serviced by a single depot and commodities are distributed by road using delivery vehicles. The industry partner does not own a fleet of delivery vehicles, but hires them from a third party logistics provider. It may therefore be assumed that an fleet of unlimited delivery vehicles is available to service stores, and that the number of delivery vehicles required for each planning period must be minimised. The industry partner has two types of delivery vehicles available, each having a different size and volumetric capacity associated with it. Each store may receive deliveries from any type of delivery vehicle, resulting in no vehicle-customer compatibilities to be adhered to. Stores may be modelled as customers that exhibit a certain quantity of demand and the service at stores must start within specified time-windows. Deliveries at stores are only allowed to take place between 09:00–17:00 on weekdays, while the depot operates between 08:00–18:00. As a result, the industry partner's scenario may be modelled as a problem instance of the MAVRP derived in §3.1. The solution returned for the case study should adhere to the aforementioned operational constraints, while minimising the overall transportation cost and adhering to a familiarity threshold specified. Permitting split deliveries may result in a reduction in the number of delivery vehicles required, as the demand of stores may be split among multiple delivery vehicles, allowing an increase in the utilisation of delivery vehicle capacities. Moreover, by ensuring that planned routes adhere to a familiarity threshold, the efficiency with which deliveries are carried out may be increased.

4.2 Input data

The case study performed in this chapter is based on a depot of the industry partner, the *Oshakati hub* (OSK), located close to the northern border of Namibia and servicing 25 stores. Unlike the geographical locations of customers in the testing problem instances implemented in §3.2 and §3.3, the locations of stores are not uniformly distributed. A map indicating the OSK depot and its assigned stores is shown in Figure 4.1. The Google Maps application programming interface was invoked to determine the travel distances and expected travel times between all pairs of coordinates and was stored in a distance matrix and travel time matrix, respectively. Each value in the travel time matrix and distance matrix is multiplied by the speed factor and cost factor of the corresponding delivery vehicle to produce a travel time matrix and cost matrix for each type of delivery vehicle, respectively. Another

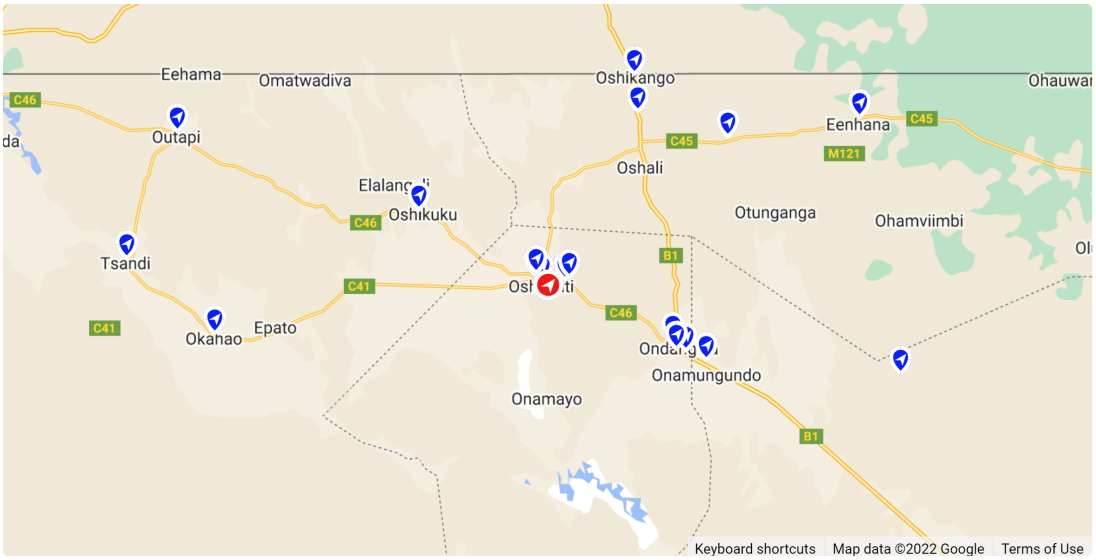


FIGURE 4.1: *The locations of the OSK depot (red vertex) and its assigned stores (blue flags).*

important difference between the case study data set and the previously implemented test instances is the magnitude of the travel distances and travel times between customers and the depot. The distances and travel times between vertices in the testing instances is much smaller than that of the case study. In the case study, the largest travel distance between a store and the depot is 226.4km, corresponding

to a travel time of 155 minutes, which is much further than the maximum travel distance of 102km and travel time of 102 minutes assumed in the test data sets. Furthermore, the ratio of the demand volumes exhibited by customers to the capacities of the delivery vehicles are also much larger when compared to the test instances. The maximum volume of demand exhibited by a store in the case study compared to the capacities of the two types of delivery vehicles available are 3.20 and 1.60, whereas the maximum demand exhibited by a customer in the test instances compared to the capacities of the three types of delivery vehicles available are 0.50, 0.36 and 0.33. Furthermore, due to the large number of stores and available delivery vehicles in the case study, a larger specified run time and computer memory is required for the instance to be solved exactly.

Information about the fleet of delivery vehicles available is summarised in Table 4.1, whereas information pertaining to the depot and its stores is provided in Table 4.2. In particular, the latitude and longitude coordinates, the average demand volumes, the demand volumes for the period under consideration, and the time-window start and end times of each store are provided. The average demand volumes for stores are based on the demand volumes exhibited for the 16-week period of 13/09/2021–27/12/2021, and are given as input to the master route generator proposed by King *et al.* [30]. The actual demand column contains the demand volumes exhibited by each store that must be satisfied during the period for which delivery routes are computed (for the week of 13/12/2021) and are given as input to the MAVRP proposed in Chapter 3.

TABLE 4.1: *Information about the types of delivery vehicles available to rent by the industry partner. The number of delivery vehicles are assumed to be unlimited since a third party logistics provider is used. The fixed cost in Rand, the capacity in cubic metres, the variable cost in Rand per kilometre travelled, and the speed factor associated with each type of delivery vehicle is shown.*

Type, k	Number of vehicles, m_h	Fixed cost, $\mathcal{F}_k$	Capacity, $\mathcal{Q}_k$	Variable cost, v_k	Speed factor, σ_k
Small	Unlimited	2 600	25	14	0.9
Large	Unlimited	3 600	50	20	1.0

The master routes were generated using the master route generator mentioned previously. The master routes returned are provided in Table 4.3, and illustrated graphically in Figure 4.2. These master routes are to be provided as input to the MAVRP model.

4.3 Results

The case study was executed on a computer with an Intel Core i7 CPU operating at 2.90GHz with 16GB of memory, and a run time limit of 90 000 seconds (25 hours) were specified. The DST discussed in §4.2 was invoked by specifying varying familiarity thresholds to observe the effect thereof on the solutions and corresponding objective function values returned. A familiarity threshold of 100% was specified first to return a solution in which delivery vehicle drivers only travel along routes with which they are familiar (the master routes generated in §4.2). Thereafter, the familiarity threshold was gradually reduced in increments of 0.1 (10%) to generate multiple trade-off solutions that may be considered when deciding on a familiarity threshold.

Information about the solutions returned when specifying each familiarity threshold is presented in Table 4.4. In particular, a brief comparison between the costs, solution run times and optimality gaps that resulted from solving each problem instance is provided. The information provided in Table 4.4 is illustrated graphically in Figure 4.3, which highlights the trends resulting from the solutions returned. It is evident that as the familiarity threshold is decreased, the objective function value of each solution also decreases (since there are less restriction on the routes that may be travelled). This may be ascribed to the fact that the total distances travelled by delivery vehicles increase when computing actual delivery routes that are not too dissimilar from the master routes. Although the optimality gap resulting from the solution found when a familiarity threshold of 90% was specified is much larger than that resulting when specifying a familiarity threshold of 100%, the solution returned corresponds to a significantly

TABLE 4.2: *Information about the OSK depot (denoted by vertex zero) and its assigned stores. The exact coordinates, the average demand volumes in cubic metres associated with stores over the period of 13/09/2021–27/12/2021, actual demand volume exhibited by these stores for the week of 13/12/2021, and the time-windows start and end times, measured in minutes from 08:00, are given.*

	Coordinates				Time-windows	
Vertex, i	Longitude	Latitude	Average demand	Actual demand, q_i	Start, a_i	End, b_i
0	-17.787085	15.720548	-	-	0	600
1	-17.772491	15.696698	16.41	17.89	60	540
2	-17.914604	15.972792	12.69	9.19	60	540
3	-18.060008	13.841303	28.90	16.88	60	540
4	-17.914436	15.972232	40.58	22.51	60	540
5	-17.771891	15.696894	48.08	68.46	60	540
6	-18.360067	16.579870	29.61	36.77	60	540
7	-17.507285	14.990581	59.40	80.86	60	540
8	-17.776238	15.694143	32.91	47.88	60	540
9	-17.469609	15.896958	16.52	26.39	60	540
10	-17.480208	16.333409	37.84	49.24	60	540
11	-17.897854	15.965865	19.61	28.31	60	540
12	-17.787026	15.708539	38.02	48.07	60	540
13	-17.918119	15.991506	41.05	56.43	60	540
14	-17.744961	14.891141	20.51	27.15	60	540
15	-17.653872	15.465963	25.05	29.08	60	540
16	-17.780009	15.762075	21.95	32.19	60	540
17	-17.935591	16.031504	20.77	25.62	60	540
18	-17.885739	15.064255	23.69	31.11	60	540
19	-17.579456	17.223150	29.55	39.53	60	540
20	-17.784844	15.753508	15.13	18.46	60	540
21	-17.434280	14.430490	20.32	27.29	60	540
22	-17.517048	16.074027	10.79	14.50	60	540
23	-17.399487	15.890360	35.80	51.73	60	540
24	-17.961166	16.414084	9.70	16.09	60	540
25	-18.356879	16.575624	12.95	10.49	60	540

TABLE 4.3: *The master routes returned for the OSK depot of the industry partner. The visitation sequence of stores in each route is provided. The depot upon departure is indexed by 0, and the depot upon return is indexed by 26.*

Route	Store visitation sequence	Route	Store visitation sequence
1	(0, 23, 9, 2, 26)	9	(0, 22, 10, 23, 26)
2	(0, 5, 1, 26)	10	(0, 6, 25, 17, 26)
3	(0, 3, 21, 20, 26)	11	(0, 3, 7, 15, 26)
4	(0, 10, 19, 6, 26)	12	(0, 18, 7, 26)
5	(0, 11, 4, 26)	13	(0, 13, 4, 26)
6	(0, 15, 12, 8, 26)	14	(0, 16, 26)
7	(0, 7, 14, 26)	15	(0, 19, 24, 26)
8	(0, 8, 5, 26)	16	(0, 13, 26)

better objective function value. Since the routes along which delivery vehicles may travel are restricted less as the familiarity threshold is decreased, there exists the possibility that a solution corresponding to

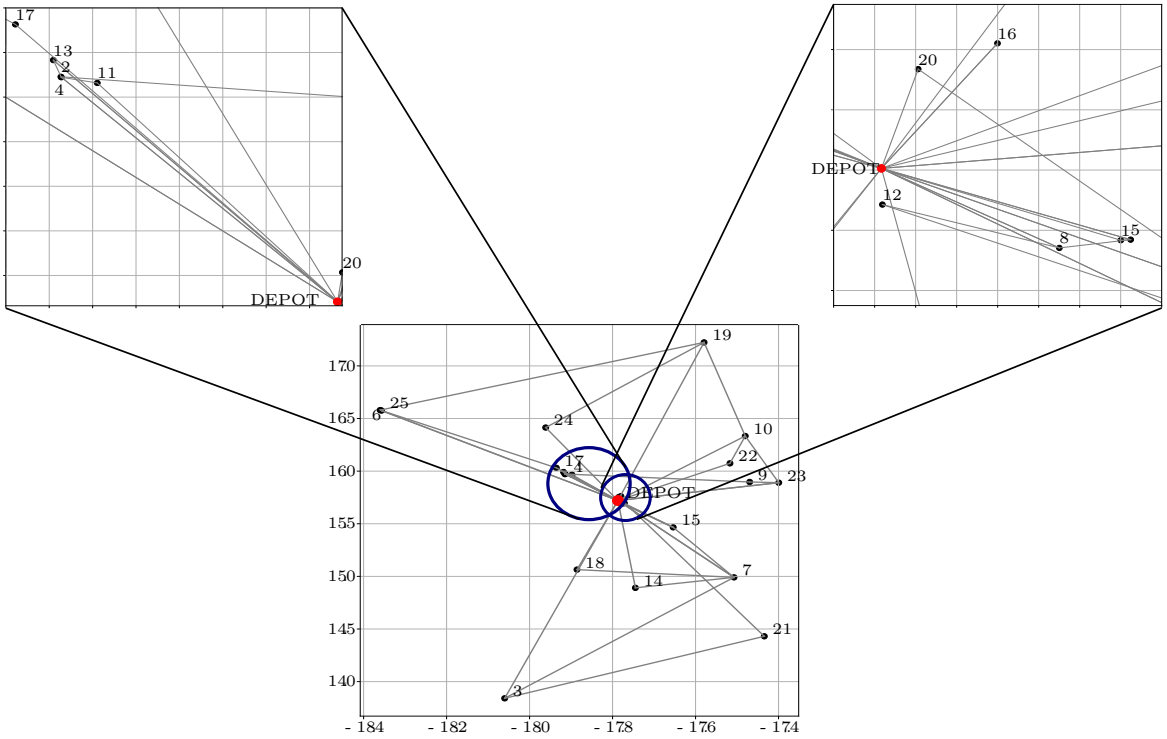


FIGURE 4.2: *The master routes generated for the OSK depot of the industry partner. The index of each store is shown next to its location, and the depot is represented by a red dot. Two enlarged figures reveal the clustered stores within the blue circles.*

a lower objective function value may be found. The subsequent improvement in the objective function values as the familiarity threshold is decreased further (smaller than 90%) are not as significant as the initial improvement, as observed in Figure 4.3. Furthermore, the number of delivery vehicles utilised also decreased as the familiarity threshold decreased (from 22 delivery vehicles when the familiarity threshold specified was 100% to 17 delivery vehicles when a familiarity threshold of 60% was specified) which also contributed to the decrease in objective function values. Consequently, there exists a clear trade-off between the transportation cost and the degree of familiarity associated with the solutions.

TABLE 4.4: *The objective function values (in Rand) of the solutions returned for the case study. The total run time (in seconds) and the optimality gap is also provided.*

Familiarity threshold	Achieved familiarity	Fixed cost	Variable cost	Service cost	Total cost	Run time	Optimality gap
100%	100%	73 200.00	71 580.88	8 064.27	152 845.15	90 030.39	20.24%
90%	90.33%	61 200.00	53 198.30	6 372.44	120 770.70	90 012.88	25.50%
80%	81.21%	61 200.00	52 568.34	6 012.41	119 780.70	90 011.61	26.44%
70%	71.54%	61 200.00	52 491.26	5 981.05	119 672.30	90 009.89	28.77%
60%	64.61%	61 200.00	52 416.08	6 135.98	119 752.10	90 010.52	29.26%

During the first execution, corresponding to a specified familiarity threshold of 100%, CPLEX was found to execute slower than expected, and the problem instance could not be solved optimally within the specified time limit. As a result, an optimality gap of 20.24% was returned for the solution, which indicated that the difference between the upper and lower bound of the CPLEX search tree was 20.24% and the feasible solution returned by CPLEX is not proven to be optimal. An increase in the solution run time is observed upon comparing the results obtained from the case study and the results of the test problem instances in §3.3. This increase is primarily due to an increase in the number of stores in the

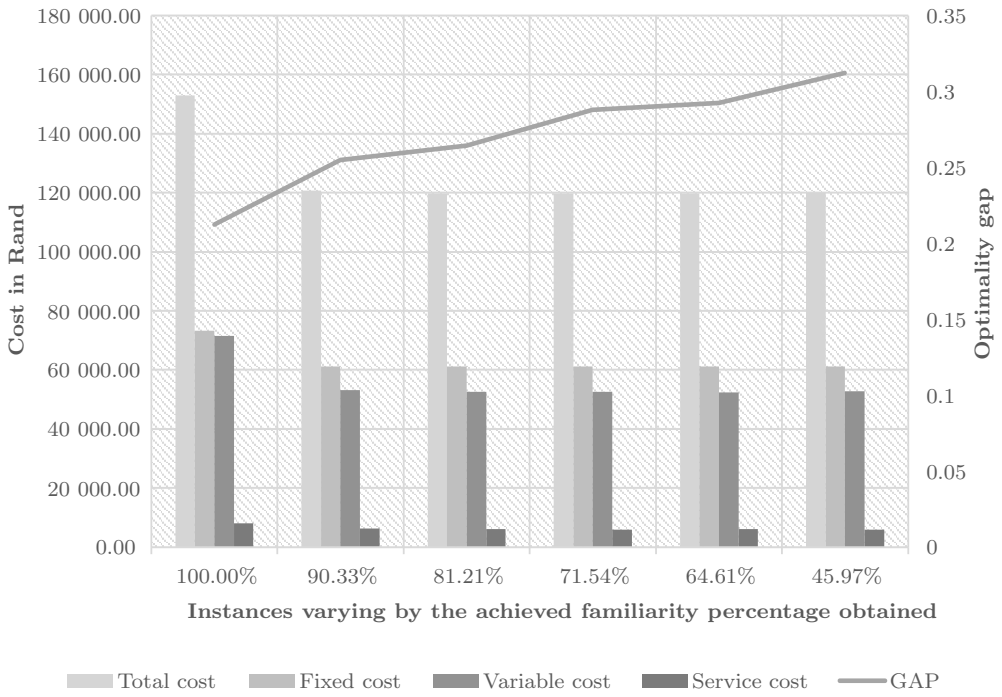


FIGURE 4.3: *A graphical representation of the solution data tabulated in Table 4.4, based on various familiarity threshold percentages. The total run time in seconds and the optimality gap is provided for each solution.*

problem instance and the number of delivery vehicles available. A larger vertex set results in a larger number of arcs which may be traversed, and an increase in the number of vehicles implies more possible ways in which these arcs may be traversed *via* different delivery vehicle combinations, which may result in a larger number of possible feasible solutions. Nevertheless, the solution returned by CPLEX may still be implemented in practice and provides valuable insight into the workings of the model. The solutions returned by CPLEX for the initial familiarity threshold of 100% is summarised in Table 4.5. In total, 22 routes were created, and the sequence of stores visited in each route is shown. The service start time at each of these stores (and the departure time at the depot) is provided and may be used to confirm adherence to the time-window constraints. Furthermore, 16 large delivery vehicles and six small delivery vehicles are utilised. The utilisation of delivery vehicles vary between 50.53% and 100%.

The DST is able to return high-quality routing solutions to a real-world problem instance. It is possible, as proven in this chapter, to invoke the computerised model of Chapter 3 for solving problem instances containing 25 customers, however, due to computing power and memory restrictions, it may not be possible to solve these instances to optimality within a realistic time-frame. It is recommended that delivery routes be computed well in advance to allow for a longer run time of the exact solution approach. A larger run time (accompanied by an increase in available computing power and memory) may allow for a larger number of iterations of the branch-and-cut algorithm (employed by CPLEX) to be performed, which may consequently result in higher quality solutions returned.

4.4 Chapter summary

In this chapter, the DST developed in Chapter 3 was applied to a case study using real world data provided by the industry partner attached to this project, in order to showcase its practical applicability. Furthermore, the verification and validation procedure was concluded in this chapter, as a full systems testing with live (real-world captured) data was performed on the computerised model. In §4.1, a brief

TABLE 4.5: *A feasible solution returned for the OSK depot of the industry partner upon specifying a familiarity threshold of 100%. The sequence of stores and the type of delivery vehicle performing the deliveries are provided. The depot upon departure is indexed by 0, and the depot upon return is indexed by 26. Service start times are measured in minutes after 08:00. The distances travelled for each route it specified in kilometres, and the utilisation of each delivery vehicle is specified.*

Vehicle type	Route	Start times	Travel distance	Vehicle utilisation
Large	(0, 19, 24, 26)	(129.50, 252.50, 466.73, 600.00)	400.60	0.79
Large	(0, 23, 26)	(332.02, 383.42, 600.00)	116.65	1.00
Large	(0, 5, 26)	(369.27, 377.06, 600.00)	8.19	1.00
Large	(0, 3, 21, 20, 26)	(13.57, 170.04, 320.08, 527.31, 600.00)	533.50	1.00
Large	(0, 7, 15, 26)	(39.12, 108.07, 400.97, 526.20)	184.92	1.00
Large	(0, 13, 26)	(273.13, 310.18, 526.20)	71.13	1.00
Large	(0, 10, 23, 26)	(31.04, 102.25, 309.62, 526.20)	211.82	0.81
Large	(0, 11, 4, 26)	(285.35, 316.43, 415.32, 526.20)	67.08	0.65
Large	(0, 16, 26)	(400.35, 409.35, 526.20)	10.60	0.64
Large	(0, 5, 1, 26)	(229.64, 237.43, 445.57, 526.20)	8.76	0.73
Large	(0, 18, 7, 26)	(6.80, 60.00, 205.30, 526.20)	221.90	0.83
Large	(0, 15, 12, 8, 26)	(37.15, 65.66, 189.26, 347.77, 507.68)	71.61	1.00
Large	(0, 7, 14, 26)	(0.00, 68.98, 344.65, 507.68)	221.85	1.00
Large	(0, 6, 25, 17, 26)	(72.14, 163.96. 284.89, 377.27, 507.68)	230.58	0.96
Large	(0, 8, 26)	(341.63, 347.77, 507.68)	6.89	0.96
Large	(0, 23, 9, 2, 26)	(81.31, 132.71, 307.57, 436.52, 507.68)	152.66	0.71
Small	(0, 6, 26)	(311.98, 395.45, 600.00)	230.17	1.00
Small	(0, 15, 26)	(451.40, 477.32, 600.00)	67.07	1.00
Small	(0, 13, 4, 26)	(274.28, 307.96, 492.15, 600.00)	71.59	1.00
Small	(0, 19, 24, 26)	(155.28, 267.10, 473.55, 600.00)	400.60	0.64
Small	(0, 22, 10, 23, 26)	(67.00, 112.50, 185.24, 388.10, 600.00)	211.82	1.00
Small	(0, 3, 21, 20, 26)	(47.07, 189.32, 331.23, 527.97, 600.00)	533.50	0.51

background discussion on the industry partner attached to this project was presented. Thereafter, a discussion on the data provided by the industry partner was presented in §4.2. The results obtained when invoking the DST of Chapter 3 was documented in §4.3, which included an overview of the practical considerations concerning the computerised model. The solutions returned by CPLEX were found to be satisfactory, however, solving instances containing a larger number of customers may result in the return of lower quality solutions.

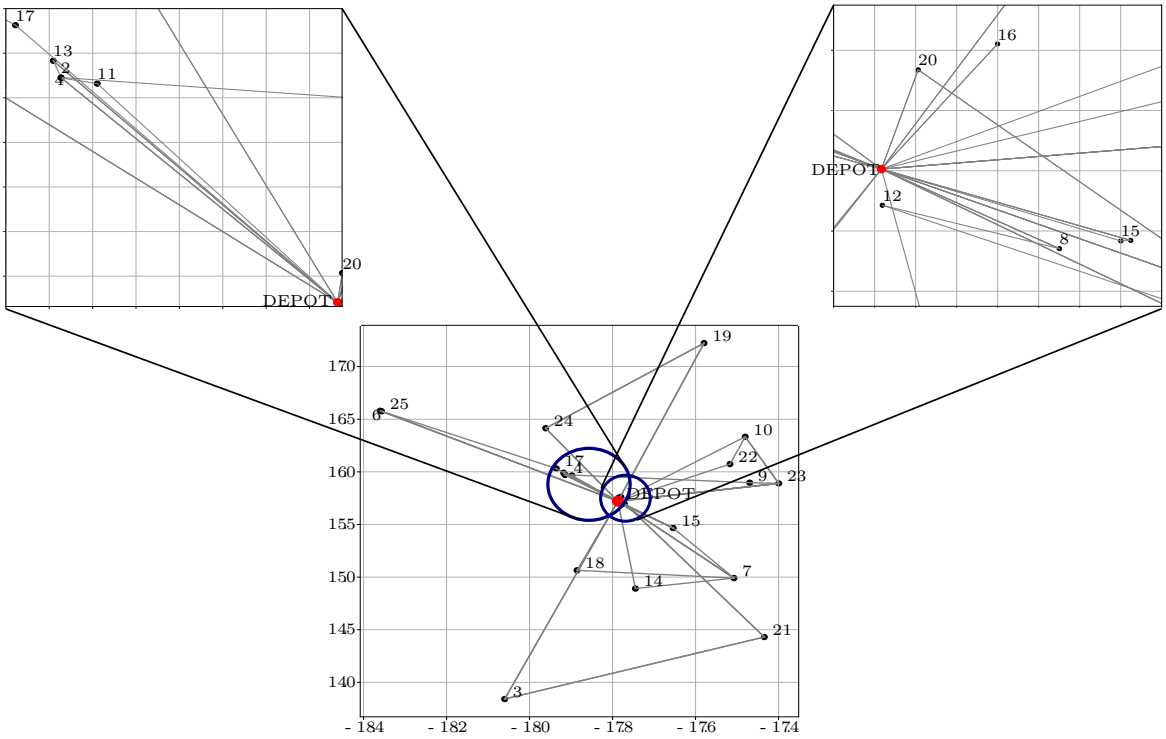


FIGURE 4.4: *The routes generated for the case study problem instance, corresponding to a familiarity threshold of 100%. The index of each vertex is shown next to its location and the depot is represented by a red vertex. Two additional enlarged plots reveal the clustered stores within the blue circles.*

CHAPTER 5

Conclusion

Contents

This conclusion chapter consists of five distinct sections. It opens in §5.1 with an overview of this project and what has been achieved. The contributions made by this project are critically appraised in §5.2, whereafter recommendations for future follow-up work based on the work performed in this project are provided in §5.3. The project's potential societal impact is discussed in §5.4, and followed by a discussion on the lessons learnt by the author during the course of this project in §5.5.

5.1 Project summary

This section provides a brief overview of what was achieved in this project, by presenting a summary of each chapter's contents. This report has four supplementing chapters in addition to this concluding one, which is an introduction in Chapter 1, a literature review in Chapter 2, a DST in Chapter 3, and a case study in Chapter 4.

An overview on the context in which this project was executed, was detailed in Chapter 1. The chapter served as a justification for the research conducted in this project. The chapter opened with a brief background description of the problem and highlights the significance of transportation costs in logistics when an organisation wants to sustain a competitive advantage. This concept was explained in further detail concerning retail organisations in the context of South Africa. A critical activity within these organisations, known as routing and scheduling support, was introduced. Thereafter, a brief introduction to VRPs as important combinatorial optimisation problems was provided, which was followed by a discussion on some of the general difficulties that often arise when retail organisations attempt to implement the solutions stemming from solving VRP instances. Thereafter, a concise description of the problem addressed by this project was provided, followed by a list of the objectives pursued, and a delimitation of the scope boundaries concerning the research conducted in this project. The research methodology adopted in pursuit of the project objectives was also explained, whereafter the chapter closed with a description of the organisation of the report.

Chapter 2 was devoted to a concise review of the literature relevant to the work done in this project, in fulfilment of Objective I in §1.3. An in-depth discussion of the classical VRP and other VRP variants relevant to the problem addressed in this project was presented in pursuit of Objective I(a). In order

to fulfil Objective I(b), formulations of the CVRP, the VRPTW, the HFVRP, and the SDVRP in the form of MIP problems were presented, and a small example problem instance was solved for each variant. Thereafter, a discussion on exact solution methodologies for solving MIP problem instances were presented in pursuit of Objective I(c). This section comprised a discussion on the simplex algorithm for solving LP problems, upon which three subsequently discussed exact solution approaches (the branch-and-bound method, the cutting plane method, and the branch-and-cut method) for solving MIP problem instances are based. Finally, in fulfilment of Objective I(d), a review on the verification and validation of mathematical models and the accompanying computer software implementation thereof was delivered.

In Chapter 3, an MAVRP in the form of a MIP problem was derived to obtain high-quality routing solutions capable of creating driver-route familiarity. The mathematical model formulated was inspired by previously researched VRP variants, in pursuit of Objective II. The parameters, decision variables, constraints and objective function of the model were derived, after which an exact solution approach and the implementation thereof in the CPLEX environment were described in partial fulfilment of Objective III. This section also initiated the model verification and validation strategy adopted by including an illustration of the workings of the computerised mathematical model *via* a small hypothetical problem instance. Randomly formulated test data were provided as input to the computerised mathematical model as a continuation of the model verification process. A user-friendly DST, capable of computing high-quality delivery routes and demonstrating the capabilities of the computerised mathematical model, was proposed in pursuit of Objective IV. Through the incorporation of a GUI within the DST, users with varying technical skills are able to employ the computerised mathematical model to customised problem instances. The verification and validation strategy was employed once more to ensure the reliability and credibility of the DST, in pursuit of Objective V.

Chapter 4 was devoted to conducting a proof-of-concept case study in which the computerised mathematical model was applied to a real-world problem instance supplied by the industry partner attached to this project, in pursuit of Objectives VI and VII. A background discussion on the industry partner attached to this project was presented, describing the logistical operations of the industry partner and how the MAVRP proposed in this project is capable of taking these operations into account. Thereafter, details about the input data of the case study were described, along with some of the key characteristics of the data set and a brief description of the environment in which the case study was conducted. A critical evaluation on the results returned after conducting the case study was presented, in fulfilment of Objective VII. Even though the case study could not be solved to optimality, the DST generated a set of high-quality routing solutions capable of creating driver-route familiarity.

5.2 Project appraisal

The literature review of VRP variants performed in §2.1 not only serves as an introduction to the concept of VRPs, but it also presents a MIP model for each variant. Each of these variants was implemented in CPLEX and invoked to generate solutions for a small example problem instance. Furthermore, in §2.2, exact methodologies for solving VRPs were discussed in great detail and demonstrated using an example problem instance. As a result, the reader is given the opportunity to thoroughly understand the formulation of various VRP variants and the workings of different exact solution methods for solving these variants.

The MAVRP proposed in this project is intended to aid a retail organisation in vehicle routing decisions with the aim of increasing driver-route familiarity. The model makes use of a set of standard delivery routes with which drivers are familiar with to address the lack of driver-route familiarity prevalent in current modelling approaches employed in industry. Aside from providing a user-friendly GUI through which the user can export the recommended delivery vehicle routes and corresponding objective function value to an Excel file, the DST can instantaneously display the recommended routes for review purposes after they have been computed. The DST also allows the user to generate new master routes or to import master routes that were created previously. The DST is able to import data sets from Excel files, allowing the user to create problem instances in a widely used software suite. The DST furthermore enables the user to specify their preferred familiarity threshold and run time limit. This versatility enables

retail companies to experiment and find routing and scheduling solutions that best suit their operational environment.

An advantage of the DST is that it may help to maximise the utilisation of a retail company's available delivery vehicles by providing acceptable solutions to VRP instances that are not possible to achieve using human intuition. An exact solution approach was implemented by employing CPLEX. CPLEX was able to find feasible solutions quickly, proving the model's functionality. Furthermore, as a proof-of-concept, the proposed DST was employed to demonstrate its applicability to practical problem instances by conducting a real-world case study. The results of the case study confirmed that by invoking the MAVRP when planning delivery routes, it is possible to achieve a desired level of driver-route familiarity, while optimising the suggested routing solutions and delivery vehicle fleet composition.

The MAVRP formulation proposed in this project takes multiple VRP attributes into account, which enables it to be applied to a wide range of practical problems. The model takes a heterogeneous fleet of delivery vehicles into account, it enforces adherence to customer time-windows, and it enables split deliveries, all while creating driver-route familiarity within solutions. The model successfully serves as a proof-of-concept for creating driver-route familiarity in planned delivery routes. Depending on the operational requirements of the user, the MAVRP may easily be adapted to include other VRP variants such as, for example, considering multiple depots. Including additional variants, however, may increase the complexity of the problem. CPLEX requires a significant amount of computer memory and solution run time to solve a realistically sized problem instance optimally. In practice, retailers having depots that serve a large number of customers may have to invoke a different solution methodology.

5.3 Suggestions for future work

This section is devoted to a discussion of suggestions for potential future work related to the work done in this project, in pursuit of Objective VIII. There are several potential future research avenues that may be pursued to improve and expand on the work accomplished by this project, but were not pursued due to time or scope limitations.

Proposal I *Proposing a metaheuristic solution approach for obtaining high-quality solutions within a shorter time-frame.*

The use of CPLEX as a solution method limits the number of customers within problem instances that may be solved within a realistic time-frame. Due to the inherent complexity of real-world problems, implementing an exact solution approach is not always feasible, and an optimal solution may not be attainable in practice [24]. In these cases, a good feasible solution being reasonably close to optimality may be obtained by invoking a *heuristic* method — a method that is likely to uncover a good feasible solution for the particular problem under consideration, but not necessarily an optimal solution. Although the quality of the solution obtained cannot be guaranteed, a well-designed heuristic method can typically deliver a close-to-optimal solution (or conclude that no such solutions exist) [24]. Heuristic methods, however, is designed to solve a specific type of problem rather than being applicable to a variety of applications. A solution methodology, termed *metaheuristics*, addresses these shortcomings.

A metaheuristic is an approach towards problem solving that offers a general framework and strategic guidelines for tailoring a particular heuristic method to a particular class of problems. In order to perform a robust search of the feasible region of a problem instance, a metaheuristic is a general solution method that governs the interaction between local improvement procedures and higher-level exploration strategies. This enables the search process to bypass local optima and, ideally, approach a global optimum [22]. Particularly, metaheuristics have served as a popular solution method for solving VRP instances [21]. Metaheuristics are widely employed and generally generate feasible solutions much faster than exact solution methods. It is therefore recommended that the MAVRP developed in this project may rather be solved by invoking a metaheuristic solution method when problem instances containing a large number of customers must be solved.

Proposal II *Including the periodic VRP variant.*

In a *periodic* VRP (PVRP), customers require service during one or more decision periods within a planning period and routes are constructed over a longer period of time [20]. The PVRP was first introduced by Beltrami and Bodin [8] in 1974 and entails generating routes to be assigned over multiple decision periods. The objective of the PVRP is to find a set of routes for available delivery vehicles that minimises total travel costs across multiple decision periods, while adhering to operational constraints (for example delivery vehicle capacities and customer requirements). Instead of considering each decision period individually, the PVRP takes all decision periods into account simultaneously, which may reduce the operational costs of satisfying the demand of customers over the entire planning period.

Proposal III *Incorporating a component that assists the user in determining an appropriate familiarity threshold and cost trade-off solution.*

Depending on the operational environment of an organisation, there may exist confusion on which familiarity threshold to specify. The incorporation of increasing driver-route familiarity may be better suited as an additional objective function, thereby modifying the MAVRP to be a multi-objective VRP. In multi-objective optimisation problems, the optimal objective function value of each objective is not necessarily suggested by the same solution, and there may exist conflict between objectives. In these cases, a so-called *satisfactory solution*, which is a feasible solution that meets or exceeds the decision maker's minimum expected level of achievement of objective values, may be accepted [40]. Incorporation of such a notion would require a solution approach where multiple non-dominated solutions are returned, and the user is able to choose their preferred solution according to a specific criteria. Rules and guidelines may assist users to make an informed decision. By incorporating such a mechanism that may guide the user in choosing a familiarity threshold and cost trade-off solution, the benefit of utilising master routes is still obtained, but at a lower expense.

Proposal IV *Including data collection from the Google Maps application programming interface within the DST.*

Problem instances include distance and travel time matrices that represent the travel distances and expected travel times between each pair of vertices in the transportation network. For real-world problems, exact travel distances and travel times may be obtained by providing the coordinates of customers and the depot as an input to the Google Maps application programming interface. In the DST proposed by this project, the user currently performs this process manually before providing it as input to the computerised mathematical model. The incorporation of data collection from the Google Maps application programming interface to determine route distances and travel times will eliminate this manual process and enable the DST to execute this task during the calculation of routing solutions.

5.4 Project contribution to society

The MAVRP proposed in this project aims to improve the problems that the retail industry currently faces when implementing the routes stemming from solving VRP instances using standard and commercially available software. Current delivery vehicle routing software does not allow drivers to become familiar with the recommended routes. When unplanned external events occur during the execution of planned delivery routes, human planners and schedulers must often make uninformed and rapid decisions. If drivers are familiar with the routes on which they must travel, it is expected to increase the efficiency with which they perform deliveries. The DST proposed by this project addresses these problems by providing delivery vehicle drivers with the opportunity to become familiarised with regularly travelled routes. Moreover, driver-route familiarity has the potential to reduce driving errors while also increasing driver confidence.

Besides the increased familiarity, the requirement for efficient routing solutions is essential for retail companies whose expenses are heavily reliant on transportation costs. Significant cost savings are possible if computerised vehicle routing and scheduling decision support is implemented, by ensuring the shortest routes are assigned to delivery vehicles. Significant cost savings may also result from more efficient utilisation of the capacities of delivery vehicles. By utilising the fleet of available delivery vehicles optimally, the number of delivery vehicles required may be reduced, resulting in a further reduction in cost. As a

result, the model proposed in this project may assist human planners and schedulers in making decisions pertaining to the routing solutions and fleet of delivery vehicles that should be made available or hired.

There are no ethical considerations pertaining to the contributions of this project, as the industry partner attached to this project and the data provided for the case study were anonymised. The DST does not seek to replace the role of human schedulers and planners in an organisation, but allows employees to shift their focus from generating optimal routing and scheduling solutions by hand, to rather focusing on decisions related to disruptions or unforeseen operational events. In particular, this project has contributed a DST as a concept demonstrator to retail organisations while creating driver-route familiarity.

5.5 Reflection on what was learnt

This section is dedicated to highlighting the technical and personal skills the author has acquired throughout the execution of this project. Continuous exposure to new knowledge domains in the field of industrial engineering was obtained. A practical overview of the industrial engineering field as a whole was obtained, as well as a deeper understanding of the field of Operations Research. Furthermore, the development and improvement of technical writing skills, the opportunity to practice time management skills, and a gain in personal development and confidence was acquired. The culmination of this knowledge base and skill set applied together resulted in the successful completion of this project. This wide range of expertise obtained included theoretical knowledge and practical skills acquired through the undergraduate degree at Stellenbosch University, but also extends further through self-taught courses and research conducted by the author.

Being an industrial engineer requires one to constantly look for innovative ways to improve processes or redesign systems to make them more efficient. This includes streamlining processes to save money and time. At the core of every successful industrial engineer is practical experience and knowledge. The execution of this project provided the author with the opportunity of applying the skills and knowledge obtained through the undergraduate industrial engineering curriculum at Stellenbosch University to gain practical experience. The complex process of conducting research, investigating a complex real-world problem, following a methodology, and synthesizing the results, among other things, were accomplished. Furthermore, a comprehensive and original solution was required, which necessitated the use of independent thinking and intuitive reasoning, which are skills that were not necessarily taught in the undergraduate course. On the technical side, exposure to complex algorithms and mathematical formulations improved the author's understanding of how such algorithms are implemented in practice. The author realised, through the help of the industry partner attached to this project, the practical considerations that are involved during the modelling of real-world systems.

The author was awarded the opportunity to improve proficiency in the field of Operations Research. The author acquired insight into the broad field of vehicle routing and combinatorial optimisation. The multifaceted nature of VRPs in general allows for a diverse set of methodologies and solution approaches to be implemented. The author was exposed to a variety of exact solution methodologies and was given the opportunity to master the CPLEX environment through its Python programming language interface. The author derived a mathematical formulation of a VRP and was able to verify and validate the derivation through the development of a computerised mathematical version. Invaluable support and expertise was received from the *Stellenbosch Unit for Operations Research in Engineering* (SUnORE) through the guidance and cooperation of its members. The author was exposed to the fields of data science, decision science, optimisation, computer science, and many more through regular meetings with both the supervisor and the group. The author developed a keen interest in the field of Operations Research, and an eagerness and desire to explore the field in more depth was prompted. An appreciation for the specialised fields within the Operations Research literature was developed by attending regular and interesting research feedback sessions from peers, master's and doctoral students, and academic personnel.

The ability to conduct professional and scientific research and of writing a technical report was exercised through this project. The author has been granted the opportunity to master the technical software environment, LaTeX, wherein this professionally formatted report was produced. Within the SUnORE research group, the author was granted an opportunity to present on the research performed in this project. This presentation was prepared by utilising the LaTeX accompanying presentation package,

Beamer. The supervisor's regular feedback and support on what constitutes a good scientific research report had a significant impact on the author's confidence in technical writing. Moreover, the group's commitment towards achieving excellence and their passion for learning inspired the author to consistently give her all in the pursuit of a higher quality deliverable.

Effective time management was critical towards completing this project and the author had to meticulously plan and develop a detailed schedule to ensure that all in-between deliverables were met. A thorough, in-depth study of a real-world problem had to be conducted, while attending to responsibilities other than those related to this project. Due to the demanding nature of the other modules pursued this year, the author had to prioritise them alongside this project. The author realised the responsibility of producing professional research outputs, and that each inclusion had to be planned rigorously to ensure the outputs are reproducible and extendable. Through successfully producing benchmark deliverables in time, the author gradually gained trust in her own competence.

In closing, the author has realised and learnt how little knowledge they actually possess, and a desire to improve on this has been prompted. A new appreciation for the available literature as a foundation for knowledge development has emerged, and the author is inspired to make a valuable contribution towards the existing Operations Research literature. It is for this reason that the author has decided to conduct further research by continuing with postgraduate studies.

References

[1] Amos D, 2022, *Python GUI programming with Tkinter*, [Online], [Cited October 2022], Available from `https://realpython.com/python-gui-tkinter/`.

[2] Applegate DL, Cook W, Dash S & Espinoza DG, 2007, *Exact solutions to linear programming problems*, Operations Research Letters, **35(6)**, pp. 693–699.

[3] Archetti C & Speranza MG, 2012, *Vehicle routing problems with split deliveries*, International Transactions in Operational Research, **19(2)**, pp. 3–22.

[4] Archetti C, Speranza MG & Hertz A, 2006, *A tabu search algorithm for the split delivery vehicle routing problem*, Transportation Science, **40(1)**, pp. 64–73.

[5] Ayers JB & Odegaard MA, 2017, *Retail supply chain management*, 2nd Edition, CRC Press, Boca Raton (FL).

[6] Balas E, 2001, *Integer programming*, pp. 492–499 in Floudas CA & Pardalos PM (Eds), *Encyclopedia of optimisation*, Kluwer Academic Publishers, Dordrecht.

[7] Baldacci R, Battarra M & Vigo D, 2008, *Routing a heterogeneous fleet of vehicles*, pp. 3–27 in Golden B, Raghavan S & Wasil E (Eds), *The vehicle routing problem: Latest advances and new challenges*, Springer, New York (NY).

[8] Beltrami EJ & Bodin LD, 1974, *Networks and vehicle routing for municipal waste collection*, Networks, **4(1)**, pp. 65–94.

[9] Bradley SP, Hax AC & Magnanti TL, 1977, *Applied mathematical programming*, Addison-Wesley, San Francisco (CA).

[10] Chen S, Golden B & Wasil E, 2007, *The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results*, Networks: An International Journal, **49(4)**, pp. 318–329.

[11] Chopra S & Meindl P, 2001, *Supply chain management: Strategy, planning, and operation*, 7th Edition, Pearson, New York (NY).

[12] Christofides N, 1976, *The vehicle routing problem*, RAIRO: Operations Research-Recherche Opérationnell, **10(1)**, pp. 55–70.

[13] Colonna P, Intini P, Berloco N & Ranieri V, 2016, *The influence of memory on driving behavior: How route familiarity is related to speed choice*, Safety Science, **82**, pp. 456–468.

[14] Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM & Soumis F, 2002, *The VRP with time windows*, pp. 176–213 in Toth P & Vigo D (Eds), *The vehicle routing porblem*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[15] Dakin RJ, 1965, *A tree-search algorithm for mixed integer programming problems*, The Computer Journal, **8(3)**, pp. 250–255.

[16] Dantzig GB & Ramser JH, 1959, *The truck dispatching problem*, Management Science, **6(1)**, pp. 80–91.

[17] Dantzig GB & Thapa MN, 1997, *Linear programming 1: Introduction*, Springer, New York (NY).

[18] Desaulniers G, Madsen OBG & Ropke S, 2014, *The vehicle routing problem with time windows*, pp. 131–171 in Toth P & Vigo D (Eds), *Vehicle routing: Problems, methods and applications*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[19] Dhara S, Sarkar S, Roy S, Mandal D & Tunga H, 2016, *Methods of capacitated vehicle routing problem based on constraints*, Proceedings of the 2^{nd} National Conference on Recent Innovations in Computer Science & Communication Engineering, Kolkata, pp. 175–180.

[20] Francis PM, Smilowitz KR & Tzur M, 2008, *The period vehicle routing problem and its extensions*, pp. 73–102 in Golden B, Raghavan S & Wasil E (Eds), *The vehicle routing problem: Latest advances and new challenges*, Springer, New York (NY).

[21] Gendreau M, Potvin J, Bräumlaysy O, Hasle G & Løkketangen A, 2008, *Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography*, pp. 143–169 in Golden B, Raghavan S & Wasil E (Eds), *The vehicle routing problem: Latest advances and new challenges*, Springer, New York (NY).

[22] Glover FW & Kochenberger GA, 2006, *Handbook of metaheuristics*, Springer Science, New York (NY).

[23] Gomory RE, 1958, *Outline of an algorithm for integer solutions to linear programs*, Bulletin of the American Mathematical Society, **64(5)**, pp. 275–278.

[24] Hillier FS & Lieberman GJ, 2014, *Introduction to operations research*, 10^{th} Edition, McGraw-Hill, New York (NY).

[25] Hoff A, Andersson H, Christiansen M, Hasle G & Løkketangen A, 2010, *Industrial aspects and literature survey: Fleet composition and routing*, Computers and Operations Research, **37(12)**, pp. 2041–2061.

[26] IBM Corporation, 2019, *Branch and cut in CPLEX*, [Online], [Cited June 2022], Available from `https://www.ibm.com/docs/en/icos/12.10.0?topic=concepts-branch-cut-in-cplex`.

[27] Intini P, Colonna P & Ryeng EO, 2019, *Route familiarity in road safety: A literature review and an identification proposal*, Transportation Research Part F: Traffic Psychology and Behaviour, **62**, pp. 651–671.

[28] Irnich S & Schneider M, 2014, *Four variants of the vehicle routing problem*, pp. 241–271 in Toth P & Vigo D (Eds), *Vehicle routing: Problems, methods, and applications*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[29] Kendall KE & Kendall JE, 2019, *Systems analysis and design*, 10^{th} Edition, Pearson Higher, London.

[30] King JCP, van Vuuren JH & Toth P, *A new vehicle routing problem for increased driver-route familiarity*, Submitted to Computers & Operations Research.

[31] Koç Ç, Bektaş T, Jabali O & Laporte G, 2016, *Thirty years of heterogeneous vehicle routing*, European Journal of Operational Research, **249(1)**, pp. 1–21.

[32] Kolman B & Beck RE, 1995, *Elementary linear programming with applications*, 2^{nd} Edition, Gulf Professional Publishing, San Diego (CA).

[33] Kumar SN & Panneerselvam R, 2012, *A survey on the vehicle routing problem and its variants*, Intelligent Information Management, **4(3)**, pp. 66–74.

[34] Laporte G, 2009, *Fifty years of vehicle routing*, Transportation Science, **43(4)**, pp. 408–416.

[35] Laporte G & Nobert Y, 1987, *Exact algorithms for the vehicle routing problem*, pp. 147–184 in Martello S, Laporte G, Minoux M & Ribeiro C (Eds), *Surveys in combinatorial optimisation*, Elsevier, New York (NY).

[36] Lavrov M, 2019, *The Cutting Plane Method*, Lecture Notes, Department of Mathematics, College of Liberal Arts & Sciences, UIUC, Urbana (OH).

[37] Martinez-Zarzoso I, Garcia-Menéndez L & Suárez-Burguet C, 2003, *Impact of transport costs on international trade: The case of Spanish ceramic exports*, Maritime Economics and Logistics, **5(2)**, pp. 179–198.

[38] Mitchell JE, 2001, *Integer programming: Branch and cut algorithms*, pp. 519–525 in Floudas CA & Pardalos PM (Eds), *Encyclopedia of optimisation*, Kluwer Academic Publishers, Dordrecht.

[39] Murray-Smith DJ, 2019, *The use of experimental data in simulation model validation*, pp. 357–382 in Beisbart C & Saam NJ (Eds), *Computer simulation validation: Fundamental concepts, methodological frameworks, and philosophical perspectives*, Springer, New York (NY).

[40] Nahum OE, Hadas Y, Spiegel U, Cohen R & Natan-Manor S, 2014, *Multi-objective stochastic VRP–Genetic algorithm and fitness complexity*, International Journal of Computer System, **1(2)**, pp. 30–44.

[41] Nasir A, Masrom S & Ahmad N, 2011, *Evaluation of vehicle routing problem with time windows by using metaheuristics algorithm*, Proceedings of the 13$^{\text{th}}$ ASME/WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering Applied Computing, Shah Alam, pp. 82–87.

[42] Naudé W & Matthee M, 2007, *The significance of transport costs in Africa*, The United Nations University World Institute for Development Economics Research Policy Briefs, **5**, pp. 1–7.

[43] Nel S, 2021, *Integer programming: The branch-and-bound method*, Lecture Notes, Department of Industrial Engineering, US, Stellenbosch.

[44] Oberkampf WL & Trucano TG, 2008, *Verification and validation benchmarks*, Nuclear Engineering and Design, **238(3)**, pp. 716–743.

[45] Oberkampf WL, Trucano TG & Hirsch C, 2004, *Verification, validation, and predictive capability in computational engineering and physics*, Applied Mechanics Reviews, **57(5)**, pp. 345–384.

[46] Padberg M & Rinaldi G, 1991, *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Review, **33(1)**, pp. 60–100.

[47] Roache PJ, 1998, *Verification of codes and calculations*, The American Institute of Aeronautics and Astronautics. Journal, **36(5)**, pp. 696–702.

[48] Sailors RM, East TD, Wallace CJ, Carlson DA, Franklin MA, Heermann LK, Kinder AT, Bradshaw RL, Randolph AG & Morris AH, 1996, *Testing and validation of computerised decision support systems*, Proceedings of the 1st American Medical Informatics Association Annual Fall Symposium, Bethesda, pp. 234–238.

[49] Sargent RG, 1982, *Verification and validation of simulation models*, pp. 159–169 in Cellier FE (Ed), *Progress in modelling and simulation*, Academic Press, London.

[50] Sargent RG, 2001, *Some approaches and paradigms for verifying and validating simulation models*, Proceedings of the 1^{st} 2001 Winter Simulation Conference, Syracuse (NY), pp. 106–114.

[51] Schlesinger S, 1979, *Terminology for model credibility*, Simulation, **32(3)**, pp. 103–104.

[52] Solomon MM & Desrosiers J, 1988, *Time-window constrained routing and scheduling problems*, Transportation Science, **22(1)**, pp. 1–13.

[53] Soonpracha K, Mungwattana A, Janssens GK & Manisri T, 2014, *Heterogeneous VRP review and conceptual framework*, Proceedings of the 2^{nd} International Multi-Conference of Engineers and Computer Scientists, Hong Kong, pp. 1052–1059.

[54] Stray J, 2022, Lead data scientist at *a large South African clothing retailer*, [Personal Communication], Contactable at `jonasstray@gmail.com`.

[55] Subramanian A, Penna P, Uchoa E & Ochi L, 2011, *A hybrid algorithm for the fleet size and mix vehicle routing problem*, Proceedings of the 1^{st} International Conference on Industrial Engineering and Systems Management, Metz, pp. 414–427.

[56] Sukati I, Hamid ABA, Baharun R, Alifiah MN & Anuar MA, 2012, *Competitive advantage through supply chain responsiveness and supply chain integration*, International Journal of Business and Commerce, **1(7)**, pp. 1–11.

[57] Thacker BH, Doebling SW, Hemez FM, Anderson MC, Pepin JE & Rodriguez EA, 2004, *Concepts of model verification and validation*, (Unpublished) Technical Report LA–14167–MS, Los Alamos.

[58] Toth P & Vigo D, 2002, *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[59] Toth P & Vigo D, 2014, *Vehicle routing: Problems, methods, and applications*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[60] Tseng Y, Yue WL & Taylor MA, 2005, *The role of transportation in logistics chain*, Proceedings of the 5^{th} Eastern Asia Society for Transportation Studies, Adelaide, pp. 1657–1672.

[61] Van Vuuren JH, 2021, *Linear Programming: Simplex algorithm*, Lecture Notes, Department of Industrial Engineering, US, Stellenbosch.

[62] Van Vuuren JH, 2022, *The vehicle routing problem*, Lecture Notes, Department of Industrial Engineering, US, Stellenbosch.

[63] Winston WL & Goldberg JB, 2004, *Operations research: Applications and algorithms*, 4^{th} Edition, Duxbury Press, Boston (MA).

APPENDIX A

Project Timeline

The expected timeline is given in Figure A.1 (on the next page) in Gantt-chart form.

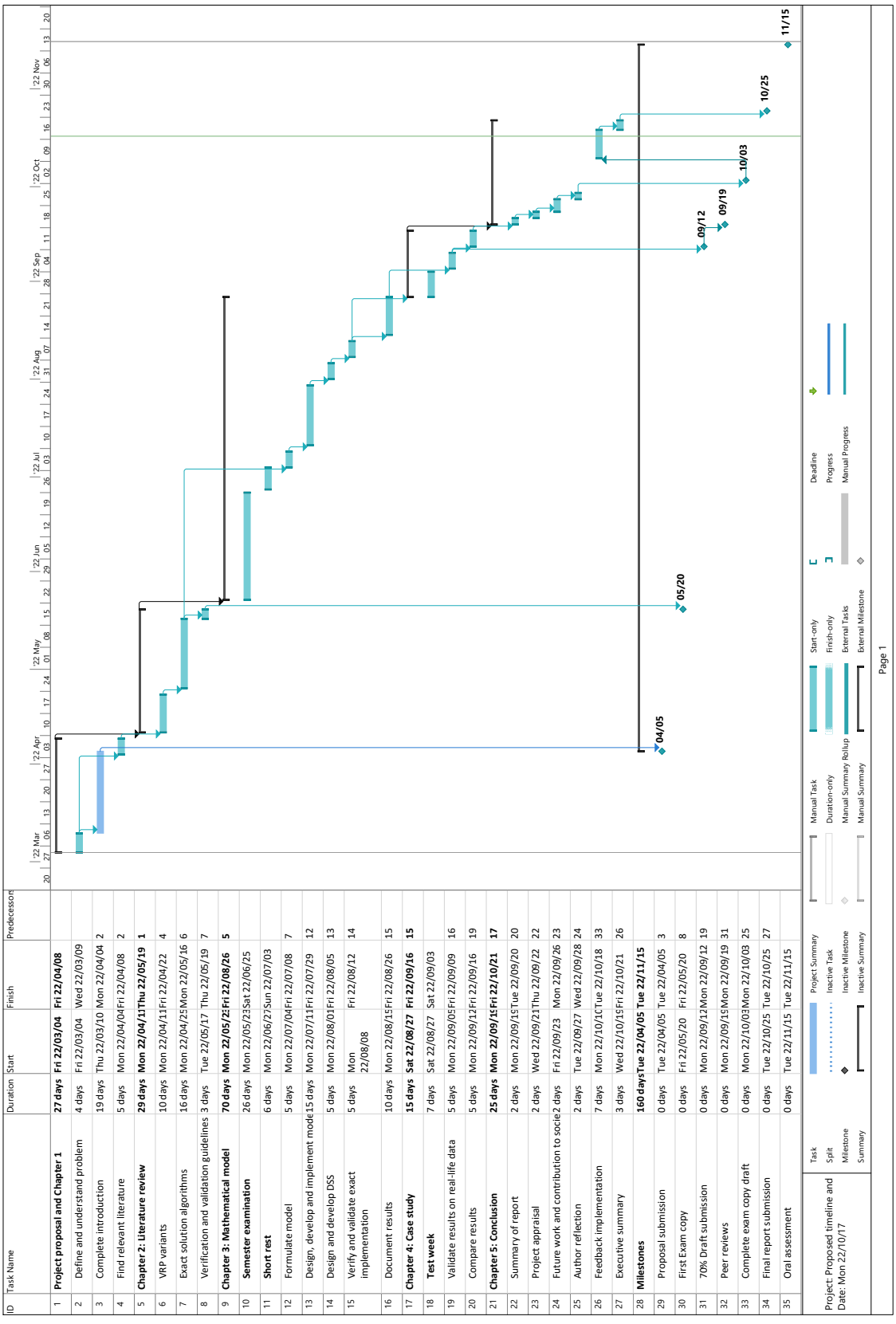


FIGURE A.1: *Expected timeline in Gantt-chart form.*