A new vehicle routing problem for increased driver-route familiarity

by

Jacobus Coenraad Petrus King

Thesis presented in fulfilment of the requirements for the degree of
Master of Engineering (Industrial Engineering)
in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof JH van Vuuren
Advisor: Prof P Toth

December 2022

Declaration

Abstract

Practical challenges often arise when implementing solutions that stem from solving vehicle routing problem instances. Unplanned external events can result in increased vehicle travel times and subsequent degradations in supply chain operational efficiency. Moreover, drivers tend to get lost and/or often travel on roads that are not suitable for the delivery vehicles utilised when they are unfamiliar with delivery routes, especially when these routes differ significantly from one day to the next. A possible solution, aimed at streamlining the practical implementation of planned delivery routes, is therefore to attempt to increase driver-route familiarity.

A novel problem, called the *familiarity vehicle routing problem* (FVRP), is proposed in this thesis for improving the practical implementation of planned delivery routes by introducing increased driver-route familiarity into vehicle delivery routes. The FVRP consists of two phases — a strategic phase and an operational phase. During the strategic phase, a set of standard delivery routes visiting each customer along a specified number of different approaches is generated for a depot and the customers it services. These routes are called *master routes* and are then used as blueprints for daily planning purposes when actual delivery routes are computed during the subsequent operational phase. Delivery vehicle drivers are thus be afforded the opportunity to become familiar with the master routes, which is anticipated to increase the efficiency with which they are able to perform deliveries (if the actual delivery routes do not deviate too much from these master routes).

Two novel mathematical models and accompanying approximate solution approaches are proposed for the different phases of the FVRP. The (single-objective) mathematical model for the strategic phase is concerned with generating a minimum-cost the set of master routes for a given depot and the customers it services. The set of arcs that form these master routes represent road links with which delivery vehicle drivers may become increasingly familiar as they continue to travel along them during future deliveries. The set of master route arcs are provided as input to the (bi-objective) mathematical model proposed for the operational phase of the FVRP. This model is concerned with computing multiple trade-off solutions which can serve as actual delivery routes along which the objectives are to minimise transportation cost and to maximise the portion of the total distance travelled along the master route arcs.

The two proposed models and their approximate solution approaches are finally applied to a special case study, involving real-world data, in order to demonstrate the practical applicability of the work in this thesis.

Opsomming

Praktiese uitdagings ontstaan dikwels wanneer oplossings van voertuigroeteringsprobleemgevalle geïmplementeer word. Onbeplande eksterne gebeurtenisse kan lei tot verlengde voertuigreistye en 'n gevolglike agteruitgang in die bedryfsdoeltreffendheid van die voorsieningsketting. Boonop is bestuurders geneig om te verdwaal en/of dikwels op paaie te reis wat nie geskik is vir die afleweringsvoertuie wat gebruik word wanneer hulle nie vertroud is met afleweringsroetes nie, veral wanneer hierdie roetes aansienlik van een dag na die volgende verskil. 'n Moontlike oplossing, wat daarop gemik is om die praktiese implementering van beplande afleweringsroetes te verbeter, is dus om te poog om bestuurders se vertroudheid met afleweringsroetes te verhoog.

'n Nuwe probleem, genaamd die *vertroudheidsvoertuigroeteringsprobleem* (VVRP), word in hierdie tesis voorgestel vir die verbetering van die praktiese implementering van beplande afleweringsroetes deur verhoogde bestuurdervertroudheid met voertuigafleweringsroetes in te voer. Die VVRP bestaan uit twee fases — 'n strategiese fase en 'n operasionele fase. Tydens die strategiese fase word 'n stel standaard afleweringsroetes vir 'n depot en die kliënte wat daardeur bedien word, gegenereer wat elke kliënt langs 'n gespesifiseerde aantal verskillende benaderings besoek. Hierdie roetes word meesterroetes genoem en dien dan as 'n bloudruk vir daaglikse beplanningsdoeleindes wanneer werklike afleweringsroetes tydens die daaropvolgende operasionele fase bereken word. Afleweringsvoertuigbestuurders word sodoende die geleentheid gebied om vertroud te raak met die meesterroetes, wat na verwagting die doeltreffendheid waarmee hulle aflewerings kan doen, sal verhoog (indien die werklike afleweringsroetes nie te veel van hierdie meesterroetes afwyk nie).

Twee nuwe wiskundige modelle en gepaardgaande benaderde oplossingsbenaderings word vir die verskillende fases van die VVRP daargestel. Die (enkeldoelige) wiskundige model vir die strategiese fase is daarop gemik of 'n minimum-koste stel meesterroetes vir 'n gegewe depot en die kliënte wat daardeur bedien word, te bereken. Die versameling boë wat in hierdie meesterroetes voorkom verteenwoordig padverbindings waarmee afleweringsvoertuigbestuurders al hoe meer vertroud kan raak soos wat hulle toekomstige aflewerings maak. Die versameling meesterroeteboë word as toevoer tot die (twee-doelige) wiskundige model verskaf wat vir die operasionele fase van die VVRP voorgestel word. Hierdie model is daarop gemik om verskeie afruiloplossings te bereken wat as werklike afleweringsroetes kan dien waarvolgens vervoerkoste geminimeer word en dáárdie gedeelte van die totale afstand langs die meesterroeteboë afgelê, gemaksimeer word.

Die twee voorgestelde modelle en hul benaderde oplossingsbenaderings word uiteindelik op 'n spesiale gevallestudie toegepas, wat op werklike data gebaseer is, om ten einde die praktiese toepaslikheid van die werk in hierdie tesis te demonstreer.

Acknowledgements

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

- First, I give praise and thanks to our Lord, Jesus Christ, for the gifts and abilities that He has given me. I thank Him for the privilege of being where I am today, and for His presence through each step of this project.
- My parents, James and Christel King, and the rest of my family, for their love, support, and belief in my abilities.
- Janél van Jaarsveld, for always being there for support and motivation during difficult times and celebrations during good times.
- My supervisor, Prof JH van Vuuren, for your invaluable mentorship and contribution towards this thesis. Thank you for inspiring me to be passionate about my work and for transferring your pursuit of excellence onto others.
- Dr Stephan Nel, for your time and assistance. Thank you for nurturing an interest in research since my undergraduate journey.
- The *Stellenbosch Unit for Operations Research in Engineering* (SUnORE) and my colleagues there, thank you for your support, assistance, and teatime conversations. This group has created a motivational environment for students to learn and grow in multiple aspects.
- Dr Jonas Stray and Prof Paolo Toth, for your time, assistance, and input into this thesis. Thank you for your willingness to allow me to take on this project with you.
- My friends for their support and contribution towards an unforgettable postgraduate journey.

Table of Contents

List of Acronyms

2E-VRP: 2-Echelon vehicle routing problem

3L-CVRP: Capacitated vehicle routing problem with three-dimensional loading constraints

ACO: Ant colony optimisation

ALNS: Adaptive large neighbourhood search

ARP: Arc routing problem

BBO: Biogeography-based optimisation

BCO: Bee colony optimisation

BFS: Basic feasible solution

CA: Cultural algorithm

CDD: Crowding distance density

CIPS: Chartered Institute of Procurement and Supply

CMA-ES: Evolutionary strategy with covariance matrix adaptation

CMT: Benchmark instances proposed by Christofides *et al.* [24]

CoEA: Co-evolutionary algorithm

CPLEX: IBM ILOG CPLEX Optimization Studio 20.1.0

CPP: Chinese postman problem

CPTP: Capacitated profitable tour problem

CVRP: Capacitated vehicle routing problem

DARP: Dial-a-ride problem

DC: Distribution centre

DCVRP: Distance-constrained capacitated vehicle routing problem

DE: Differential evolution

EA: Evolutionary algorithm

EDA: Estimation of distribution algorithm

EP: Evolutionary programming

EPH: Empangeni hub

ES: Evolutionary strategy

FSMVRP: Fleet size and mix vehicle routing problem

FVRP: Familiarity vehicle routing problem

GA: Genetic algorithm

GIS: Geographic information system

GLS: Guided local search

GP: Genetic programming

GRASP: Greedy randomised adaptive search procedure

GRP: General routing problem

GWKC: Benchmark instances proposed by Golden *et al.* [60]

HGSADC: Hybrid genetic search with advanced diversity control

HV: Hypervolume

HVRP: Heterogeneous vehicle routing problem

ILS: Iterated local search

IP: Integer programming

IRP: Inventory routing problem

LIFO: Last-in-first-out

LNS: Large neighbourhood search

LP: Linear programming

MAVRP: Multi-attribute vehicle routing problem

MDVRP: Multiple depot vehicle routing problem

MTVRP: Multi-trip vehicle routing problem

MVCTP: Multi-vehicle covering tour problem

MVRPB: Mixed vehicle routing problem with backhauls

OVRP: Open vehicle routing problem

PCVRP: Prize-collecting vehicle routing problem

PDP: Pickup-and-delivery problem

PI: Pattern improvement

PIX: Periodic crossover with insertions

PR: Path relinking

PS: Pareto set

PSO: Particle swarm optimisation

PVRP: Periodic vehicle routing problem

RI: Route improvement

RPP: Rural postman problem

SA: Simulated annealing

SDVRP: Split delivery vehicle routing problem

SI: Swarm intelligence

SME: Subject matter expert

SS: Scatter search

ST: Single truck

TOP: Team orienteering problem

TSP: Travelling salesman problem

TTC: Truck-and-trailer combination

TTRP: Truck-and-trailer routing problem

VNS: Variable neighbourhood search

VRP: Vehicle routing problem

VRPB: Vehicle routing problem with backhauls

VRPDDP: Vehicle routing problem with devisable deliveries and pickups

VRPMS: Vehicle routing problem with multiple synchronisation constraints

VRPPC: Vehicle routing problem with private fleet and common carrier

VRPRB: Vehicle routing problem with route balancing

VRPSPD: Vehicle routing problem with simultaneous pickup and delivery

VRPSTW: Vehicle routing problem with soft time-windows

VRPTT: Vehicle routing problem with trailers and transshipments

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 the collection of activities performed by an organisation with a view to deliver goods or services from a point of origin (*i.e.* raw material extraction) to a point of consumption (*i.e.* product or service delivery to an end-user) [20]. Although customer satisfaction — typically expressed in terms of a so-called customer service level — is determined at the point of consumption, it depends sensitively on the upstream (preceding) activities in the supply chain. When customers visit a clothing store, for example, it is expected that they will be allowed to try on various items of clothing. The shopping experience of the customer can, to a large extent, depend on the variety and availability of clothing items and sizes at the store. Upstream activities that are not visible to the customer, but are essential to ensuring variety and availability of products, include the sourcing of raw materials, the storage of materials in warehouses, the manufacturing of clothing products, the transportation and storage of clothing products in a *distribution centre*[1] (DC), and the delivery of clothing products to stores. The competitive advantage and success of such a retail organisation (and organisations in many other industries) depend largely on the organisation's ability to manage and reduce costs effectively across all activities constituting the supply chain [128].

The *Chartered Institute of Procurement and Supply* (CIPS) has proposed a three-phase model of supply chain management [20]. The three phases and their constituent stages are illustrated graphically in Figure 1.1. In a traditional manufacturing supply chain, the procurement phase consists of interactions with suppliers. This involves sourcing, transportation (inbound logistics), and storage of raw materials in warehouses. During the operations management phase, the stored

[1]A facility at which large quantities of products are stored and from which deliveries are made to multiple stores or customers.

raw materials are used to manufacture products, which are then stored in DCs and, at a later stage, transported to sales locations (outbound logistics). Finally, the consumer phase involves the sales of products to customers.

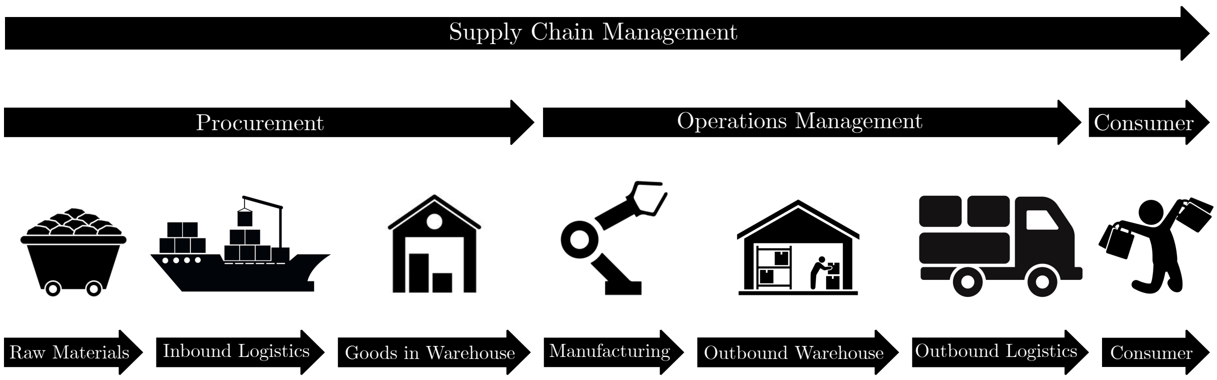


FIGURE 1.1: *General activities constituting a supply chain. Based on CIPS [20].*

A supply chain often involves using a fleet of dedicated delivery vehicles to transport thousands of physical units of goods from an appropriately located DC, often called a *depot* in the operations research literature, to multiple stores, often also referred to as *customers*, which are typically distributed geographically. Each vehicle is presented with a schedule specifying the sequence of customers to be visited, together with the number of product units of demand realised at these locations. Collectively, this forms part of an important activity performed during the outbound logistics stage of the operations management phase, called *vehicle routing*. The planning of routes for these delivery vehicles (and their subsequent refinement) can usually be aided significantly by modelling the scenario as a so-called *vehicle routing problem* (VRP). Computerised vehicle routing and scheduling decision support, which is grounded analytically in the solution of such models, has resulted in cost savings ranging from 5% to 20% in the context of managing a supply chain's distribution planning activities [134].

The VRP is one of the most studied and important combinatorial optimisation[2] problems in the operations research literature [134]. In 1959, Dantzig and Ramser [34] published the first paper on the VRP, titled *The truck dispatching problem*. The aim in this seminal paper was to find the best route set for a fleet of gasoline delivery vehicles departing from a central terminal and visiting multiple fuel stations to service their fuel demands. The quality of a VRP solution is often expressed in terms of the total transportation cost incurred or time expended by the delivery vehicles away from the depot, or the total distance travelled by these vehicles. Generally, a VRP instance is concerned with determining the best set of routes along which to distribute goods from a specified depot to a given set of customers [134]. The solution is a set of routing schedules, one for each delivery vehicle, which satisfies the demand of all the customers and adheres to a variety of operational and other constraints, whilst minimising the total transportation cost or distance travelled [134].

Many variations on the basic notion of a VRP can be found in the literature. The variant that is the simplest and most popular, is the so-called *capacitated VRP* (CVRP) [137]. In the formulation of a CVRP instance, each customer exhibits some demand to be satisfied by exactly one delivery vehicle, and a fleet of homogeneous delivery vehicles is available for servicing the demand of customers, without exceeding the capacity of any vehicle.

[2]A process of searching for an optimal solution when only a finite number or countably infinite number of feasible solutions exist [144].

If there are k identical vehicles in a CVRP instance with n customers, the size of its solution space is enumerated by the so-called *Stirling number of the first kind*, denoted by $S(n,k)$, if the capacities of the vehicles are neglected. In general, this number enumerates the distinct ways in which n distinguishable objects can be distributed among k indistinguishable containers, arranging the items in each container cyclically (*i.e.* along a cycle), so that no container is empty [94]. If the k vehicles are capacitated, however, then not all $S(n,k)$ solutions enumerated by the Stirling number of the first kind will, of course, be feasible. The solution space of a CVRP instance with n customers and k identical vehicles will, therefore, have a cardinality strictly smaller than $S(n,k)$, but typically still of the same order of magnitude as $S(n,k)$. There is no closed-form expression for $S(n,k)$, but its value can be computed recursively by means of the recursion relation

$$S(n,k) = S(n-1,k-1) + (n-1)S(n-1,k), \quad n > k > 1$$

in conjunction with the initial values $S(n,1) = n!$ and $S(n,n) = 1$ for any $n \in \mathbb{N}$. As a function of n and k, the Stirling number of the first kind is increasing in n and decreasing in k. For any fixed value of k, however, the value of $S(n,k)$ grows very quickly as n increases, as illustrated in Table 1.1.

Table 1.1: *Values of the Stirling number of the first kind, $S(n,k)$, for $n \in \{12, 15, 18, 21, 24\}$ and $k \in \{4, 8, 12\}$.*

	$k = 4$	$k = 8$	$k = 12$
$n = 12$	105 258 076	357 423	1
$n = 15$	310 989 260 400	2 681 453 775	143 325
$n = 18$	1 583 313 975 727 488	24 871 845 297 936	4 853 222 764
$n = 21$	12 870 931 245 150 988 800	311 333 643 161 390 640	135 585 182 899 530
$n = 24$	157 375 898 285 941 510 732 800	5 304 713 715 525 445 812 976	4 070 384 057 007 569 521

While exact CVRP solution methods typically traverse the solution space of a problem instance implicitly and much more intelligently than by brute force (*i.e.* they do not explicitly consider each feasible solution in turn), the order of magnitude of the Stirling number of the first kind, $S(n,k)$, nevertheless gives an indication of the intrinsic complexity of solving CVRP instances and motivates why vehicle dispatch managers typically require automated support in respect of their complicated vehicle routing decisions.

Extensions of the CVRP are illustrated graphically in Figure 1.2. A natural extension of the CVRP is the *vehicle routing problem with time-windows* (VRPTW) in which service at each customer can only start during a specified time interval, called a *time-window*. Time-windows can also be one-sided — *i.e.* allowing the service time at a customer to start as early as possible, but no later than some specified time [137]. This extension of the CVRP is especially prevalent in the retail environment where customers often specify preferred time intervals during which deliveries are to be made.

Another popular variation on the CVRP is the *periodic VRP* (PVRP) in which n customers have to be visited, once each by a single vehicle from a fleet of k identical, capacitated vehicles, over a period of T days. The customers to be visited on each of the T days have to be identified, after which a CVRP instance has to be solved for each day separately, involving only those customers actually assigned visitation on that day.

Since the well-known multinomial coefficient

$$\binom{n}{n_1, \ldots, n_T} = \frac{n!}{n_1! n_2! \cdots n_T!} \tag{1.1}$$

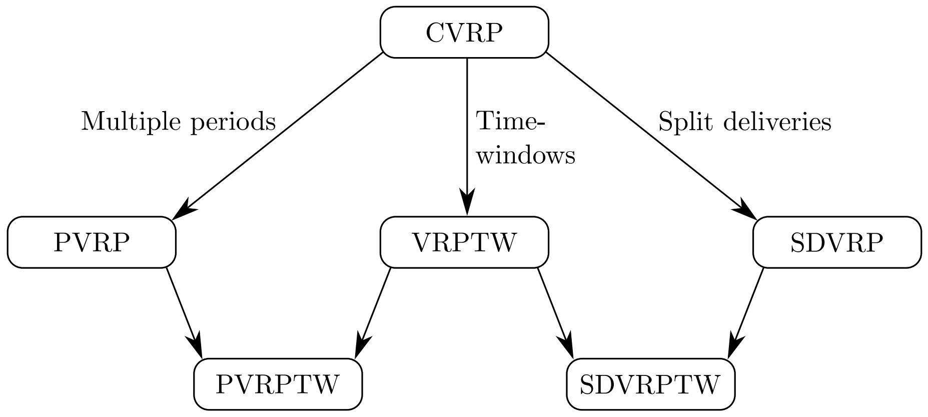


FIGURE 1.2: *The basic VRP variants and their relationships.*

in general represents the number of ways in which n distinguishable objects can be distributed amongst T distinguishable containers [118], it follows that the number of ways in which n customers can be assigned visitation days in a PVRP instance spanning T days, with n_i customers visited on day $i \in \{1, \ldots, T\}$, is given by the quantity in (1.1). If there are k identical vehicles in the PVRP instance, its solution space cardinality is, therefore, of order

$$\sum_{\substack{(n_1,\ldots,n_T):\\ n_1+\cdots+n_T=n\\ n_i \geq k \text{ for all } i}} \binom{n}{n_1,\ldots,n_T} \prod_{i=1}^{T} S(n_i,k) \gg k^n.$$

This cardinality grows very rapidly as n, k and/or T increases, generally disqualifying exact solution methodologies[3] for even moderately sized values of these parameters, despite sophisticated algorithmic parallelisation efforts and the impressive array of modern computing technology typically available in the retail sector today.

In the *split delivery VRP* (SDVRP), another variation on the CVRP, demand satisfaction at each customer may be partitioned into a number of visitations so that a customer can be serviced by more than one delivery vehicle. Each vehicle may therefore satisfy fractions of the demand of the customers that it services. Allowing for split deliveries may reduce the cost of solutions by up to 50% of the cost when not allowing split deliveries [6]. A further extension of the SDVRP was proposed by Gulczynski *et al.* [65] in which each customer specifies a minimum fraction of demand to be satisfied by any single delivery vehicle. This allows each customer to be serviced by no more than a preferred number of delivery vehicles. The SDVRP is a very complex combinatorial optimisation problem for which only problem instances involving at most thirty customers can currently be solved by means of *exact algorithms* [7].

When multiple VRP attributes (such as time-windows and split deliveries) are combined into a single problem, it is called a *multi-attribute VRP* (MAVRP). Interest in MAVRPs is typically motivated by specific applications and may include all three of the previously mentioned CVRP extensions, as well as others. The complexity of solving such an MAVRP instance naturally combines the underlying complexities of all of its constituent problems, resulting in an even more difficult problem to solve. This often renders infeasible the use of exact algorithms for solving even small MAVRP instances. Metaheuristic[4] solution approaches are instead required in order to find high-quality solutions to MAVRP instances within an acceptable time frame.

[3]Algorithms capable of solving an optimisation problem instances to optimality.

[4]A solution approach that combines local search procedures and higher-level search strategies in order to escape from local optima and perform a robust search of the solution space [58]. Such a solution approach is

1.2 The industry partner attached to this thesis

The industry partner attached to this thesis, who prefers to remain anonymous, is a large South African clothing retailer. It reportedly experiences practical outbound logistics challenges when attempting to implement recommendations stemming from solving VRP instances by means of standard, commercial software [126]. Externalities (typically unplanned), such as unanticipated traffic conditions, can result in increased vehicle travel times and subsequent degradations in supply chain operational efficiency. Moreover, drivers tend to get lost and/or often travel on roads that are not suitable for the delivery vehicles utilised when delivery routes differ significantly from one day to the next. As a result, the operational phase of implementing VRPTW solutions can be affected negatively (*e.g.* a driver might miss the time-window of the next scheduled delivery). This can easily result in a knock-on effect, rendering the rest of the driver's planned delivery schedule unusable.

A possible solution aimed at streamlining the practical implementation of planned delivery schedules, suggested by Stray [126], is to generate a set of standard delivery routes visiting each customer a specified number of times along different approaches, called *master routes*. Each master route should start and end at a given depot and visits a subset of the customers serviced by the depot in a specific sequence. The set of master routes may be considered valid for a single season (spring and summer combined, or autumn and winter combined) and should be computed for each season based on expected travel times between the depot and customers, as well as the average demand volumes of customers. During the season, when actual demand experienced by customers deviate from the averages on which the computation of the master routes were based, the master routes may then be used as blueprints for daily planning purposes when delivery routes are computed. Delivery vehicle drivers will thus be afforded the opportunity to become familiar with these routes, which is anticipated to increase the efficiency with which they perform deliveries if the actual delivery routes do not deviate too much from the master routes.

Decisions required for the creation of master routes may be modelled as an instance of a novel VRP, which includes many of the attributes described in the previous section, as illustrated graphically in Figure 1.3. Each customer (or retail store in the case of the industry partner) would have to be visited a specified number of times by different master routes. Redundancy should, however, be built into the master routes, in order to allow more freedom when computing actual delivery routes later during the season. The number of master routes visiting each customer may be specified *a priori* based on the average demand quantities of the customer during the relevant season.

The actual delivery routes computed during the season may be modelled as an instance of another novel VRP also including many standard attributes, as illustrated graphically in Figure 1.3. In this model, each customer would have to be assigned service on certain days during the planning horizon, although the demand of a customer may be split between delivery vehicles (and even over different days). Moreover, the capacities of delivery vehicles may not be exceeded, and service at each customer should start during its specified time-window. The solution should also adhere to several other operational constraints (such as enforcing a source to sink path for each vehicle and only assigning vehicles to stores that may, in fact, visit those stores), minimise the total transportation cost, and simultaneously attempt to maximise the extent to which daily delivery routes coincide with combinations of segments of the master routes. These delivery routes are to be computed over pre-specified planning horizons, each consisting of p decision

not guaranteed to reach an optimal solution, although a high-quality solution may typically be uncovered within a much shorter time-frame than that associated with an exact method, if the search strategy is appropriately tailored to the particular optimisation problem instance at hand.

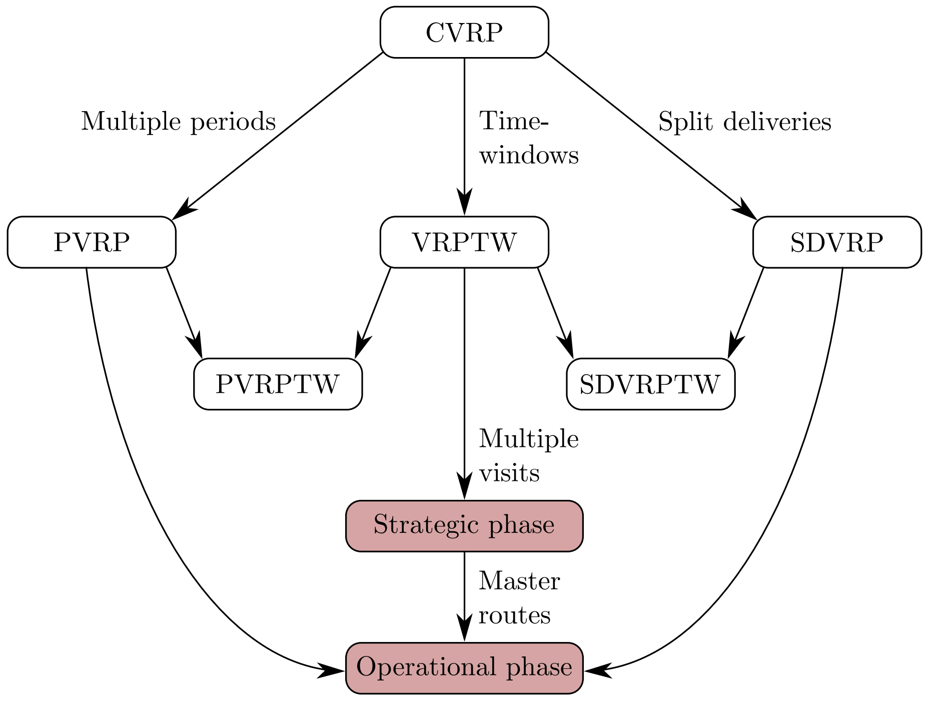


Figure 1.3: *Two novel MAVRPs introduced in this thesis and their relationships with other VRP variants.*

periods. The industry partner has expressed an interest in planning horizons representing a duration of a week and consisting of $p = 5$ decision periods, representing weekdays on which deliveries may take place.

An additional benefit of basing the computation of delivery schedules on the aforementioned two novel VRPs in conjunction with one another, aside from increasing driver-route familiarity, is that it may well increase the overall efficiency of the supply chain. First, instead of promising a specific day on which the demand of a customer will be fulfilled, the demand can be satisfied on any day during the relevant planning period. By including the multiple period attribute in the routing model that assigns certain customers to be serviced on each day of the planning period, the total travel distance required to fulfil the demand of all the customers during the planning period may be reduced, compared to when the day on which each customer has to be serviced is fixed.

Furthermore, by incorporating multiple days in a planning period and allowing split deliveries, the demand of customers during a planning period does not have to be delivered during a single visitation, but can be delivered by multiple delivery vehicles, spread over multiple days. A maximum volume of delivered stock that should not be exceeded on any single day can be specified by each customer. This would allow customers to unpack stock as it arrives, without the need for large storage spaces, while it may, in contrast, take a few days for customers to unpack their total delivered stock for the week. By allowing split deliveries across multiple periods, the number of delivery vehicles required may also be reduced, since the volume of demand delivered by a delivery vehicle on each day can be increased with appropriate stock so as to facilitate visitation of customers in an optimal sequence.

The quality of the actual delivery routes are, however, expected to depend markedly on the quality of the master routes. If master routes are devised that do not visit high-demand stores with sufficient multiplicity, or render it impossible to arrive at some store during its time-window, the quality of daily delivery schedules will be correspondingly poor, or else the actual delivery routes will be very dissimilar to the master routes, thus defeating the objective of achieving driver familiarity of delivery routes.

1.3 Informal problem description

The aim in this thesis is to propose a novel VRP, called the FVRP, for improved delivery routing which addresses the practical limitations derived from a lack of driver-route familiarity which is prevalent in standard VRP formulations, as highlighted by the industry partner. The FVRP consists of two phases, a strategic phase and an operational phase.

The first, strategic phase is concerned with determining an appropriate set of master routes for each depot. Each set of master routes should be valid for a season and is computed for each season based on expected travel times between a relevant depot and the customers it services, as well as the average demand volumes exhibited by these customers. Vehicle drivers may thus be afforded the opportunity to become familiar with these master routes, which is expected to increase the efficiency with which they are able to perform deliveries if the actual delivery routes do not deviate too much from these master routes. The strategic phase is to be modelled as an instance of a novel VRP, in which each customer has to be visited a specified number of times by different master routes, thereby building redundancy into these master routes so as to allow for more freedom when computing actual delivery routes in the future. The number of distinct master routes visiting each customer is to be specified based on the average demand quantities of the customer during the relevant season.

The second, operational phase is to be be modelled as an instance of another novel VRP, during which the master routes (computed during the strategic phase) are used as blueprints for daily planning purposes when actual delivery routes are computed. These actual delivery routes are computed for planning horizons consisting of multiple decision periods. In order to achieve high-quality delivery routes, the output of the operational phase should assign customers to be serviced during certain decision periods within the planning horizon and satisfy a set of operational constraints, while minimising the total transportation cost and maximising the lengths of segments travelled along the master routes.

Solution approaches for the models proposed for the two phases are to be implemented on an applicable software platform, serving the purpose of a concept demonstration of the FVRP proposed in this thesis. These implementations should be validated by applying them to a case study involving real-world data provided by the industry partner attached to this thesis.

1.4 Scope and objectives

The following eight objectives are pursued in this thesis:

I To *conduct* a thorough survey of the literature related to:

(a) The VRP and its variants in general,

(b) exact methods for solving VRPs, and

(c) (meta)heuristic methods for solving VRPs approximately within a realistic time-frame.

II To *derive*, based on the guidelines in the literature derived during the pursuit of Objective I(a), a VRP formulation for each phase of the FVRP. These formulations should include:

   (a) A model for the strategic phase of the FVRP, capable of determining an appropriate set of master routes for a given depot and the customers that it services. The master routes should be valid for a season and be computed for each season based on expected travel times between depots and customers, as well as on the average demand volumes of customers, and

   (b) a model for the operational phase of the FVRP, which uses the master routes computed by solving the model for the strategic phase of Objective II(a) as blueprints for daily planning purposes when actual delivery routes are computed. These actual delivery routes are to be computed for a planning horizon consisting of multiple decision periods. In order to achieve high-quality delivery routes, the output of the operational phase should assign customers to be serviced during certain decision periods of the planning horizon and satisfy a set of operational constraints, while minimising the total transportation cost and maximising the degree of intersection with segments taken from the master routes.

III To *verify* and *validate* the VRP formulations derived in pursuit of Objective II, according to generally accepted modelling guidelines.

IV To *propose*, based on the literature reviewed in fulfilment of Objectives I(b) and I(c), solution methods capable of solving instances of the VRP formulations derived in Objective II within an acceptable time-frame.

V To *implement* the solution approaches of Objective IV on an applicable software platform.

VI To *apply* the implementations of the solution approaches of Objective V to a real-world case study in order to demonstrate their practical application.

VII To *evaluate* the effectiveness of the FVRP of Objective II and its proposed solution approaches of Objective IV in terms of its capability to create driver-route familiarity and generate high-quality delivery schedules in the context of the case study of Objective VI.

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

The scope of the research carried out towards this thesis is limited by the following assumptions:

- The exact and metaheuristic methods researched in pursuit of Objectives I(c) and I(d), respectively, are limited to the best-known and prolific methods in the operations research literature with pertinent relevance to the mathematical models derived in this thesis.

- Problem instance-specific input parameters, such as the average travel times between depots and customers, delivery vehicle information, and time-windows, are provided as input to the FVRP models by the user or organisation implementing the solutions. No attempt is made to research realistic values of these parameters.

- Input parameters pertaining to the creation of master routes, such as the number of overlapping master route arcs allowed without penalty and the penalty coefficient for the number of overlaps above the maximally tolerated threshold, are specific to the preferences of the user or organisation implementing the solutions and are therefore provided as input.

- The number of visits stipulated for each customer during the strategic phase of master route generation is based on anticipated average demand quantities for the customer during the relevant season and is also specified as input to the model for the strategic phase by the user organisation. No attempt is made to forecast these demand quantities.
- The implementations of the solution approaches of Objective V are limited in scope to the design, implementation, verification, and validation of a concept demonstrator of the proposed FVRP. The development and integration of a full-scale DSS with the business processes and database of the industry partner attached to this thesis is not pursued.

1.5 Research methodology

The research reported in this thesis is executed in four phases. During the first phase, a thorough survey of the literature related to the work documented in this thesis is conducted in pursuit of Objective I. First, a fundamental understanding of the characteristics of relevant VRP variations is pursued in fulfilment of Objective I(a). Thereafter, exact and approximate solution approaches to VRPs are reviewed in pursuit of Objectives I(b) and I(c), respectively, and in support of Objective II. This research phase also encompasses the acquisition of a number of relevant technical skills by the author, such as developing a proficiency in the *IBM ILOG CPLEX Optimisation Studio 20.1.0* (CPLEX) software suite and in the programming language Python.

The second phase of the research is pursuant of Objectives II–V. First, two mathematical models are derived for the FVRP which may be used to determine a set of master routes and create actual delivery routes as solution, in pursuit of Objectives II(a) and II(b), respectively. Verification and validation of the derived models are performed in pursuit of Objective III within the context of randomly generated test instances. Exact solution approaches are also proposed for the two mathematical models in partial pursuit of Objective IV and are implemented in the CPLEX software suite, accessed *via* Python. Moreover, approximate (metaheuristic) solution approaches are proposed in final fulfilment of Objective IV, and implemented afresh in Python in fulfilment of Objective V. The solution approach implementations are designed to be modular and able to accommodate instance customisation in the form of a suite of user-specified parameter values, which are able to solve a wide variety of FVRP instances.

The third phase of the research comprises the application of the implementation of the proposed solution approaches to a real-world case study, in order to demonstrate and validate its practical application. The case study itself is carried out in two substages. During the first substage, a general background to, and appropriate data for, the case study are documented. The solution approach implementations are then used to evaluate the real-world data and deliver numerical results in the form of master routes as well as daily delivery routes for the case study instance in pursuit of Objective VI. During the second substage, the numerical results are evaluated and discussed in fulfilment of Objective VII, resulting in a pronouncement on the FVRP effectiveness in the context of the case study.

The final phase of the research relates to Objective VIII and entails summarising the contributions of this thesis and proposing prioritised suggestions for possible follow-up work which may be pursued in the future.

1.6 Thesis organisation

Apart from the current introductory chapter, this thesis comprises a further eight chapters (organised into four parts), and a bibliography. Part I is a literature review and comprises two chapters, Chapters 2 and 3. Chapter 2 is devoted to an overview of a taxonomy for VRP variants. The chapter opens with an introduction to the VRP after which the taxonomy is discussed. This is followed by a discussion on the characteristics of a number of VRP variants.

Chapter 3 is devoted to a discussion on three algorithmic solution methodologies for solving VRPs. The chapter opens with a prerequisite discussion on the simplex algorithm for solving linear programming problems, upon which all three exact solution approaches discussed thereafter are based. A variety of approximate VRP solution methodologies are also reviewed and the working of a state-of-the-art metaheuristic, called the *Hybrid Genetic Search with Advanced Diversity Control* (HGSADC) algorithm, is discussed. Finally, approximate VRP solution approaches are discussed in the context of multi-objective optimisation problems.

Part II of the thesis comprises a further two chapters, Chapters 4 and 5, and is focused on the proposed FVRP. Two novel mathematical models which form the working basis of the proposed FVRP are derived in these chapters. First, a mathematical model for creating master routes, related to the strategic phase of the FVRP, is derived in Chapter 4. Thereafter, a mathematical model for using these master routes during the generation of actual daily delivery routes over a user-specified planning horizon, related to the operational phase of the FVRP, is derived in Chapter 5. During the derivation of both models, the parameters, variables, constraints, and objective function of each model are discussed in detail. The approaches followed to implement, verify and empirically estimate the time complexity of each model is also documented.

Part III of the thesis also comprises two chapters, Chapters 6 and 7, which are collectively dedicated to a real-world validation case study of the FVRP in the operational context of the industry partner attached to this thesis. A general background on the case study is provided and the real-world data utilised during the case study are described in Chapter 6. The numerical results obtained during the case study are subsequently reported and discussed in Chapter 7.

The thesis closes in Part IV, which comprises a final two chapters, Chapters 8 and 9. Chapter 8 is devoted to a summary and self-appraisal of the contributions made in this thesis, while prioritised possible follow-up work, related to the work reported in this thesis and which may be pursued in the future, is proposed in Chapter 9.

Part I

Literature Review

CHAPTER 2

Vehicle Routing Problems

Contents

This chapter is devoted to a description of VRP variants based on a taxonomy proposed by Toth and Vigo [137]. An introduction to the VRP and the proposed taxonomy may be found in §2.1. The first class of variants, arising from changes in the network structure, are discussed in §2.2. Variants that arise from different types of transportation requests are described next in §2.3. Thereafter, variants resulting from the imposition of various intra-route and inter-route constraints are discussed in §2.4 and §2.5, respectively. Furthermore, variants arising from different fleet characteristics are discussed in §2.6. Finally, different objectives pursued in the VRP literature are described in §2.7. The chapter closes in §2.8 with a brief summary of the work presented in the chapter.

In order to limit the length of the discourse, mathematical formulations are not given for the VRP variants reviewed in this chapter. Reference are, however, included in which the various model formulations may be found.

2.1 Introduction

Dantzig and Ramser [34] were first to introduce the VRP in 1959. Thy were concerned with determining the best set of routes for a fleet of gasoline delivery vehicles, with the objective of minimising the total distance travelled by all of these vehicles. The VRP was formulated in this early paper as the *travelling salesman problem* (TSP) with the addition of a vehicle capacity constraint. An algorithmic solution approach was also suggested which is able to reach near optimal solutions for small instances of the problem. Since 1959, major developments have taken place in the research area of vehicle routing, including a large number of variations in the constraints and objectives of VRP formulations, as well as numerous exact and approximate solution methodologies for these problem variations, driven by a wide range of practical applications in industry. Today, exact algorithms exist that are able to solve TSP instances containing tens of thousands of vertices [2]. Although the VRP is a natural generalisation of the TSP, it is much harder to solve, with the currently best exact algorithms only being able to solve VRP instances containing of the order of one hundred vertices [9, 51].

Toth and Vigo [137] proposed a taxonomy for the numerous VRP variations available in the literature. They classified VRPs according to

- their network structure,
- the type of transportation requests involved,
- the constraints that affect each route individually (intra-route constraints),
- the fleet composition and the locations at which vehicles are stored,
- various inter-route constraints, and
- the optimisation objectives pursued.

The remainder of this chapter is devoted to a discussion on variations of the basic VRP of Dantzig and Ramser [34], based on the above-mentioned taxonomy of Toth and Vigo [137].

2.2 Network structure

A VRP may be classified according to whether tasks are represented by the vertices or arcs of a graph modelling the underlying transportation network. In the classical VRP, tasks are performed at specific points in space represented by the graph vertices. It is therefore classified as a so-called *vertex routing problem* [137]. The TSP is an example of a vertex routing problem in which each vertex of the underlying graph should be visited once, at minimal cost [104]. In an *arc routing problem* (ARP), on the other hand, tasks are not performed at vertices, but along graph arcs which may represent street segments [137]. The problem of traversing each arc of a graph at least once, at minimal cost, is known as the *Chinese postman problem* (CPP) and is an example of an ARP. Another example of an ARP is the *rural postman problem* (RPP) in which each arc in a subset of required arcs should be visited at least once, at minimal cost. When tasks are performed on both the vertices and arcs of the underlying graph, the problem is known as a *general routing problem* (GRP). In a GRP, each arc in a required subset of arcs and each vertex in a required subset of vertices should be visited at least once, at minimal cost.

A VRP may also be classified according to the data representing travel costs. If travel costs are symmetric (*i.e.* the same cost is incurred when traversing an arc in any direction), the problem is called symmetric and may be modelled by means of an undirected graph. If some arcs in the network may, however, only be traversed in a single direction, or the travel costs incurred when traversing some arcs in opposite directions differ, the problem is called asymmetric and is modelled by means of a directed graph [137].

A distinction between a VRP and an ARP/GRP may be made based on the granularity of the underlying data and graph [137]. Arcs in an ARP or a GRP usually represent individual street segments, whereas arcs in a VRP may represent entire shortest paths between vertices, which may themselves consist of multiple street segments. The distances and travel times in a VRP therefore have to be determined by the routing component of a *geographical information system* (GIS) and many optimal shortest paths may exist between a pair of vertices. Garaix *et al.* [52] addressed this problem by proposing to model VRPs on multi-graphs in which parallel arcs represent the optimal routes.

Finally, VRPs may be classified as dynamic or static, and deterministic or stochastic. In a *dynamic VRP*, some data are not known in advance, but only become available during operation, such as the travel time between vertices which may depend on the traffic conditions on the day of travel. If a VRP is not dynamic, it is called a *static VRP*. In a *stochastic VRP*, on the other hand, some data may be described by random variables associated with given probability distributions. If a VRP is not stochastic, it is called a *deterministic VRP*. The classic CVRP is neither stochastic nor dynamic, but rather deterministic and static since all the underlying data are fixed and known in advance.

2.3 Type of transportation request

According to the taxonomy proposed by Toth and Vigo [137], VRP variants in the literature are also classified based on the nature of the following transportation requests:

Delivery and collection. In the CVRP, commodities are collected from a single point (a depot) and delivered to multiple points (customers) in order to satisfy demand, resulting in a *one-to-many* VRP. The opposite situation involves the collection of commodities from customers (called pickups) and delivery thereof to a depot, and is known as a *many-to-one* VRP. Examples of many-to-one VRPs include the collection of empty glass bottles at consumption sites and the disposal of waste at homes to a waste site. New VRP variants result when both pickups and deliveries are allowed. The earliest and simplest variant in this class is the *VRP with backhauls* (VRPB) in which all deliveries are made from a depot to customers and all collections are made from customers and brought to the depot [135]. In this VRP variant, all customers at which deliveries have to be made are visited first, after which customers at which pickups occur are visited. Each delivery vehicle therefore arrives empty at the first customer at which a pickup is performed. This is done to avoid complications associated with rearranging items in delivery vehicles when pickups are performed between deliveries. If, however, it is possible to rearrange items in delivery vehicles (for example, loading and unloading from different sides), customers requiring pickups are allowed to be serviced before customers requiring deliveries, and the resulting variant is called the *mixed VRPB* (MVRPB) [142].

In the VRPB and MVRPB, each customer requires either a collection or a pickup, but both are not allowed. The variant in which customers are allowed to require both pickups and deliveries is called the *VRP with simultaneous pickup and delivery* (VRPSPD) [99]. Many real-world scenarios exist in which this variant is applicable, such as the simultaneous delivery of beverages and the pickup of empty bottles. The *VRP with devisable deliveries and pickups* (VRPDDP) is a relaxation of the VRPSPD which may result in cost savings [137]. Instead of performing the pickup from and delivery at a customer during the same visit, customers may be visited more than once and the delivery and pickup may be performed during different visits.

Simple visits. In some cases, neither collection nor pickup occurs at customers, but customers simply have to be visited [137]. An example of such a case is the repair or service of a product at customers that do not require commodities to be delivered or picked up.

Alternative and indirect services. In some cases, each customer may be serviced at alternative locations. In this case, the choice of location and a routing based on these locations have to be determined. An example of such a case is when a package may be delivered either to a customer's home, office, or post box. The *multi-vehicle covering tour problem* (MVCTP) involves servicing customers by visiting a location close enough to each customer (*i.e.* one of the alternative locations associated with the customer) [66].

Point-to-point transportation. When goods are not delivered from a depot to customers or *vice versa*, but from any point in the network to any other point, the problem is considered a *many-to-many VRP* and is often referred to as the *pickup-and-delivery problem* (PDP) [39] in the case of goods transportation or the *dial-a-ride problem* (DARP) [137] in the context of passenger transportation. A real-world application scenario corresponding to this type of problem is a taxi driver assigned scheduled trips or a courier transporting items between customers.

Repeated supply. In some cases, customers may require repeated visits throughout a planning horizon consisting of multiple periods, rather than a single visit on a specific date. This variant is referred to as the PVRP [28]. In the PVRP, a visiting pattern (the periods during which particular customers are to be visited) first have to be selected for each customer from a given set of admissible patterns. Thereafter, a VRP must be solved for each period, in which the subset of customers to be visited and the commodities to be delivered during each period depends on the visiting patterns selected [137]. A required number of visits may be specified for each customer. The commodities to be delivered to each customer during a period may then simply correspond to the total demand for the planning horizon divided by the number of required visits [50].

In the *inventory routing problem* (IRP), repeated supply is also required, but no customer orders take place [18]. Instead, the delivery company decides when to visit each customer and how many commodities to deliver so that a stock-out does not occur at any customer. This variant has enjoyed growing interest in the field of supply chain management due to the faster and more reliable information exchange that is today possible between customers and their suppliers, leading to shorter lead times, lower inventory levels, and higher service levels [137]. Furthermore, additional objectives, such as minimising inventory holding costs, are often included in the IRP. In both the IRP and the PVRP, the commodities delivered to customers during any period may also be limited according to the maximum storage capacity available at the customers.

Non-split and split services. Commodities may either be delivered at a customer in a single drop by a single delivery vehicle, or may be split into multiple drops delivered by different delivery vehicles. In the SDVRP, the demand of each customer may be split into arbitrarily many smaller batches delivered by different delivery vehicles [43]. Cost savings of up to 50% of the total routing distance are reportedly possible if split deliveries are allowed, compared to when not allowed, under the assumption of identical input data [137].

Combined shipment and multi-modal service. When the demand of a customer is not split, but the full demand is satisfied by more than one delivery vehicle using transfer points, the situation is referred to as *combined shipments* [137]. An example of combined shipments is when larger delivery vehicles transport commodities over long distances, such as from factories to depots, and smaller delivery vehicles are used for last-mile delivery from the depots to customers. Variants that arise from allowing combined shipments include *hub-and-spoke* delivery and *crossdocking*. The *2-echelon VRP* (2E-VRP) involves utilising intermediate depots, called satellites, as transfer points between depots and customers [85].

Routing with profits and service selection. It is sometimes impossible to service all customers with a limited delivery vehicle fleet. In this case, only a subset of the customers can be visited, or only some of the demand of customers can be satisfied. Cost savings may result from optimising routing and request selection simultaneously, instead of first performing request selection and then routing. This can be achieved either by including constraints on service levels and costs, by penalising unserviced requests, or by rewarding serviced ones. Many variants of this type of problem exist in which different approaches are adopted. In the *capacitated profitable tour problem* (CPTP), routing costs and profits are combined into one objective [4]. In the *team orienteering problem* (TOP), the route lengths of delivery vehicles are limited and the profit associated with servicing customers is maximised [5]. In the *prize-collecting VRP* (PCVRP), there is a minimum profit to be made and the routing cost is minimised [131].

The so-called *VRP with private fleet and common carrier* (VRPPC) has attracted interest in the recent literature. In the VRPPC, a subset of customers may be serviced with owned vehicles (a private fleet) while the remaining customers are subcontracted to a common carrier at a fixed price [25].

Dynamic and stochastic routing. In the *dynamic VRP*, some information only becomes available during operations, such as the locations of customers or the demand of customers [133]. Moreover, in a *stochastic VRP*, some information is uncertain, but follows a known distribution, such as the demand of customers, or the travel times between customers or between the depot and customers [13]. In a stochastic VRP, the impact of uncertainty on the routing cost and service levels is analysed [137].

2.4 Intra-route Constraints

According to the taxonomy proposed by Toth and Vigo [137], VRP variants in the literature are further classified based on the constraints that determine whether a route is feasible or infeasible, called intra-route constraints. Adherence to these constraints can be verified for each individual route, independently of other routes.

Loading. In the basic CVRP, the loading constraint of a route involves checking whether the commodities to be delivered to the customers visited along any route conform in number or quantity with the corresponding capacity of the delivery vehicle. This constraint typically only refers to a single dimension of capacity, such as weight or volume, although additional constraints may be added to account for different dimensions of capacity [137].

More complex loading constraints may also be included to account for two-dimensional or three-dimensional quantities. In these variants, multi-dimensional packing problems and VRPs are combined. In the *CVRP with three-dimensional loading constraints* (3L-CVRP), for example, additional loading constraints are included to ensure the stability of stacked boxes, the safety of fragile boxes, and the ease of unloading at customers [53]. Many such variants exist for specific cases such as two-dimensional quantities, pallet packing, and delivery vehicles with compartments.

Route length. The length or duration of a route refers to the resources consumed on the arcs traversed along the route. In the *distance-constrained CVRP* (DCVRP), a maximum route length is specified [24]. This constraint may also be adapted to limit the duration, fuel consumption, or other aspects associated with each route [8].

Multiple use of vehicles. In the basic CVRP, it is assumed that each delivery vehicle only performs a single trip. It is, however, possible for a vehicle to return to the depot, reload commodities, and complete additional trips, if the total duration of all trips is still within feasible bounds. In the *multi-trip VRP* (MTVRP), each vehicle may complete multiple trips [130]. This variant has become increasingly popular due to the growing prominence of electric delivery vehicles that often complete short trips before returning to the depot for recharging purposes before departing to complete additional trips [137].

Time-windows and scheduling aspects. Many practical VRP variants require some form of scheduling which may include the consideration of travel and service times, waiting times, and time-windows [137]. In the VRPTW, each customer specifies an earliest and latest possible time during which service is allowed to start at that customer [27]. Furthermore, travel times are associated with the arcs of the network, and the service start times of delivery vehicles at customers form part of the decision variables of the model. Generally, it is possible for a delivery vehicle to arrive at a customer before the start of its time-window by having the vehicle wait before starting service at the customer. The duration of such a waiting period is called *waiting time*. The opposite situation, where a delivery vehicle only arrives at a customer after the latest possible service start time of the customer, is not allowed. Some variants also include the service duration at each customer, but this may simply be included in the travel times to customers. Other variants allow for multiple non-contiguous time-windows for each customer, where the service start time at a customer must be within any of its specified time-windows. Furthermore, in the *VRP with soft time-windows* (VRPSTW), linear penalties are added for late or early service start times [129]. Some variants also exist that take into account driving regulations such as maximum weekly or daily driving times and required daily rest periods.

2.5 Inter-route Constraints

Constraints that render individual routes feasible or infeasible, and the problem variants that arise from these constraints, have already been discussed. Different variants may, however, arise from *inter-route constraints*, also called *global constraints*, where the feasibility of solutions not only depends on the feasibility of individual routes, but also on the combination of all routes [137]. Variants may, for instance, arise from the use of balancing constraints, in which the difference between the maximum and minimum route duration, load, length, or number of stops, may not exceed a certain threshold [14]. These constraints are often included to ensure that the workloads of delivery vehicle drivers are evenly distributed.

Other variants also arise when vehicles compete for globally limited resources [137]. Practical examples include a restriction on the number of vehicles that may be assigned to a specific depot, or a limited processing capacity at a destination to which multiple deliveries must be made.

Finally, variants arise from the need for synchronisation between the routes and schedules of different vehicles. Drexl [41] performed a systematic study of the *VRP with multiple synchronisation constraints* (VRPMS) and offered the following classification of synchronisation:

Task synchronisation. A decision must be made as to which vehicles jointly fulfil each task, such as in the case of the SDVRP or the PVRP where each customer may receive multiple visits from different delivery vehicles.

Operation synchronisation. Some tasks may require different vehicles to perform tasks at the same or different locations, where the tasks may have to be performed at the same time or else at different times, but respecting operational precedence specifications. An example of such a case occurs when two technicians are required to perform an installation in two distinct phases, and one phase has to be completed before the other one may start [59].

Movement synchronisation. Some tasks may require that more than one vehicle should travel along the same parts of a route. Examples of such a case include a trailer that has to be pulled by a delivery vehicle, or multiple snow plows required to remove snow along a street network at the same time [119].

Load synchronisation. It is required to ensure that the correct volumes or amounts of total commodities are delivered to or picked up at customers by all of the delivery vehicles combined.

Resource synchronisation. The total utilisation and consumption of resources available for the delivery vehicles should not be exceeded at any point in time.

2.6 Fleet Characteristics

In the basic CVRP, it is assumed that all available delivery vehicles are identical, that they are stationed at the same depot, and that they adhere to the same routing constraints. This is, however, seldom the case and VRP variants with different fleet characteristics therefore exist.

The multiple depot VRP. In the *multiple depot VRP* (MDVRP), the available delivery vehicles are identical, but the routes of each delivery vehicle may start and end at different depots [114]. Each depot may therefore have a limited or unlimited fleet stationed at it, called a *subfleet*. In another variant, depots can act as replenishment facilities along the routes of delivery vehicles, therefore allowing vehicles to reload and service additional customers [32]. This variant relates to the previously mentioned variants allowing for repeated use of vehicles.

The heterogeneous or mixed fleet VRP. A classification scheme for heterogeneous fleet VRPs was provided by Baldacci *et al.* [50]. In the *heterogeneous VRP* (HVRP), a limited fleet of different delivery vehicles is available to service customers, whereas in the *fleet size and mix VRP* (FSMVRP), an unlimited fleet of different delivery vehicles is available to service customers. In both the HVRP and FSMVRP, delivery vehicles may have different fixed costs, variable costs, capacities, travel times, and site-dependencies associated with using them. Site-dependencies refer to compatibility dependencies between customers and delivery vehicles, meaning that some types of delivery vehicles are only allowed to service some of the customers. This is often included in VRP formulations to account for vehicle size restrictions at customers. The FSMVRP may also be aimed at strategic decision making, rather than operational routing, since it is often related to the optimal acquisition of a fleet of delivery vehicles [137].

The routing of trucks and trailers. In the *truck-and-trailer routing problem* (TTRP), the fleet consists of at least two types of delivery vehicles — *single trucks* (STs) which are vehicles without trailers, and *truck-and-trailer combinations* (TTCs) [19]. Site dependencies often exist because of the manoeuvring space required by a TTC. Furthermore, it may be possible for a TTC to decouple its trailer at a customer and complete a subtour containing customers that are not compatible with TTCs after which it returns to pick up the trailer again.

In the *VRP with trailers and transshipments* (VRPTT), there are no fixed assignments between trailers and trucks [42]. A trailer may therefore be pulled by more than one delivery vehicle on parts of its itinerary.

2.7 Objectives

The objective in the basic CVRP is to minimise the total distance travelled by all delivery vehicles. As mentioned, some VRP variants include different or additional criteria in their objective functions. Furthermore, some VRP variants have multiple objective functions.

Single objective optimisation. In a *single-objective optimisation problem*, the task is to find a single solution that optimises the objective function [37]. In the context of a VRP, this means that a single solution exists which minimises the total routing cost. Additional terms may, however, be added to or removed from the objective function to take different aspects of VRP variants into account. In the *open VRP* (OVRP), for example, vehicles do not return to the depot after having serviced customers [86]. The cost of return trips to the depot is therefore not included in the objective function.

When, for instance, the type of transportation request is modelled as *routing with profits and service selection*, as discussed in §2.3, the objective function often includes a profit term for tasks being performed. When accommodating intra-route constraints, such as in routing with time-windows and scheduling components, as discussed in§2.4, additional terms may be added to the objective function to achieve certain operational objectives. The route duration or latest completion time of a delivery vehicle may, for example, be minimised. Customer satisfaction may also be incorporated by penalising a time discrepancy between the actual service start times and the desired service start times at customers. Furthermore, waiting times at customers may be penalised. In the HVRP and FSMVRP, discussed in §2.6, the fixed costs associated with using delivery vehicles may be included as an additional term in the objective function.

Many metaheuristics consider both feasible and infeasible solutions in order to retain diversity in a population of candidate solutions and later arrive at high-quality feasible solutions by applying neighbourhood search operators [140]. The infeasible aspects of solutions, such as arriving at a customer after the latest possible service start time or exceeding the capacity of a delivery vehicle, may be penalised using weights in the objective function.

A recently introduced family of VRP variants is called the *green vehicle routing problem* in which energy consumption and pollution is minimised [137].

Multiple hierarchical objectives. Minimising route length, duration, and the number of vehicles used are usually conflicting objectives. Since high fixed costs are often associated with using delivery vehicles, a hierarchical approach towards optimisation may be followed in which the number of delivery vehicles used is first minimised, upon which the number of vehicles is fixed and a second objective, such as distance travelled, is minimised [17].

Multi-criteria optimisation. Many real-world problems, including VRPs, exhibit multiple conflicting objectives [37]. In the *VRP with route balancing* (VRPRB), for instance, two objectives are minimised [72]. The total distance travelled by all delivery vehicles is minimised while also attempting to minimise the difference between the longest route length and the shortest route length. This approach is often adopted to ensure workload fairness among delivery vehicle drivers.

In a *multi-objective optimisation problem*, there is more than one objective function, and there is usually no single optimal solution, but rather a set of many optimal solutions, known as the *Pareto set* (PS) [37]. Many optimisation algorithms exploit the concept of domination to determine the PS. A solution $\boldsymbol{x}^{(1)}$ is said to dominate another solution $\boldsymbol{x}^{(2)}$ if both the following conditions are true:

1. Solution $\boldsymbol{x}^{(1)}$ is no worse than solution $\boldsymbol{x}^{(2)}$ in all objectives, and
2. solution $\boldsymbol{x}^{(1)}$ is strictly better than solution $\boldsymbol{x}^{(2)}$ in at least one objective.

If at least one of these conditions is not met, solution $\boldsymbol{x}^{(1)}$ does not dominate solution $\boldsymbol{x}^{(2)}$. Given a finite set $\mathcal{P}$ of solutions, pairwise comparisons can be performed between all solutions and a set of non-dominated solutions $\mathcal{P}'$ can be determined. This set of non-dominated solutions has the property of not being dominated by any other solution in the given finite set of solutions [37]. If the finite set of solutions represents the entire feasible decision space, then the set of non-dominated solutions is also the PS.

Consider, for example, a bi-objective VRP instance in which both objectives are minimised, and suppose that a finite solution set representing the entire feasible decision space for this VRP has been identified, as indicated in the objective space in Figure 2.1. Upon comparison of Solutions 1 and 2, it is clear that none of the objective function values of Solution 1 is worse than the objective function values of Solution 2. Instead, Solution 1 is strictly better in terms of both objective function values than Solution 2. Since both conditions for domination are met, it is concluded that Solution 1 dominates Solution 2. Upon comparison of Solutions 1 and 3, it is clear that the first condition of domination is not met, since the value of f_1 for Solution 1 is worse than that of Solution 3. Solution 1 therefore does not dominate Solution 3. Upon continuation of such comparisons between all pairwise solutions, it is found that only Solutions 1 and 3 are not dominated by any other solution — they therefore form both the non-dominated and the PS.

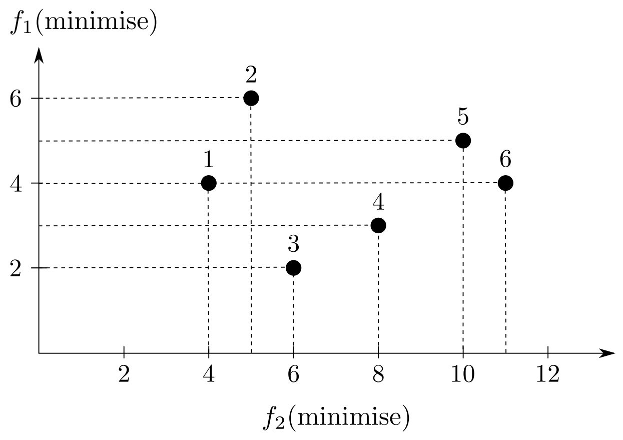


FIGURE 2.1: *A set of five solutions to a VRP instance with two objectives which are both to be minimised. The non-dominated solutions are Solutions 1 and 3.*

2.8 Chapter Summary

This chapter was devoted to a discussion on VRP variants based on a taxonomy proposed by Toth and Vigo [137]. A brief history of the VRP, as well as an overview of the proposed taxonomy, was given in §2.1. This was followed by a discussion on VRP variants arising from changes in the network structure in §2.2. In §2.3, different transportation requests and their resulting VRP variants were discussed. The variants that arise from different intra-route and inter-route constraints were next discussed in §2.4 and §2.5, respectively. Penultimately, VRP variants that are classified based in their fleet characteristics were discussed in 2.6. Finally, different types of objectives in VRP variants were described in §2.7.

CHAPTER 3

VRP solution methodologies

Contents

This chapter is devoted to a discussion on algorithmic solution methodologies for solving VRPs. Three exact solution approaches are discussed in §3.1. Thereafter, heuristic solution approaches are described in §3.2, and this is followed by a brief review of metaheuristic solution approaches in §3.3. A comparison of various metaheuristics is next presented within the context of vehicle routing in §3.4. The working of the HGSADC algorithm, a state-of-the-art solution methodology employed later in this thesis, is described in §3.5, and this is followed in §3.6 by a discussion on approaches followed when solving multi-objective optimisation problems approximately. The chapter is finally brought to a close with a brief summary of its contents in §3.7.

3.1 Exact solution approaches

Three exact VRP solution approaches are discussed in this section, namely the branch-and-bound method in §3.1.2, the cutting plane method in §3.1.3, and the branch-and-cut method[1] in §3.1.4. The section opens in §3.1.1, however, with a prerequisite discussion on the simplex algorithm for solving linear programming problems, upon which all three exact solution approaches considered in this section are based. The working of the various solution approaches discussed in this section are all illustrated by solving a two-variable integer programming problem instance in which the objective is to

$$\text{maximise} \quad z = 8x_1 + 5x_2 \tag{3.1}$$

subject to the constraints

$$\begin{aligned} x_1 + x_2 &\leq 6, & (3.2)\\ 9x_1 + 5x_2 &\leq 45, & (3.3)\\ x_1, x_2 &\in \mathbb{N}. & (3.4) \end{aligned}$$

3.1.1 The simplex algorithm for linear programming

The standard form of a *linear programming* (LP) problem is to

$$\text{maximise}^{2} \quad z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{3.5}$$

subject to constraints of the form

$$\begin{aligned} a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n &= b_i, \quad i = 1, \ldots, m, & (3.6)\\ x_j &\geq 0, \quad j = 1, \ldots, n, & (3.7) \end{aligned}$$

where n denotes the number of decision variables and m denotes the number of constraints, with $n > m$. The problem in (3.5)–(3.7) may be rewritten in matrix form by defining the matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},$$

as well as the vectors

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \text{ and } \quad \boldsymbol{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}.$$

[1]An apology related to the method of review of the exact optimisation methods in this section is in order. While it is acknowledged that there are considerable and beautiful theories underlying each of these methods, their workings are described here purely from a methodological point of view, without referring to the theories mentioned above and without proving the correctness of the methods. In this way, the length of the discourse is limited while still achieving a degree of self-containment of the material in this thesis

[2]The fact that the objective is to maximise the value of z is without loss of generality, since any function $f(\boldsymbol{x})$ can be minimised by maximising its negation $-f(\boldsymbol{x})$.

In matrix form, the objective is therefore to

$$\text{maximise } z = \boldsymbol{c}^T\boldsymbol{x}, \tag{3.8}$$

subject to the constraints

$$\begin{aligned} \boldsymbol{Ax} &= \boldsymbol{b}, && (3.9)\\ \boldsymbol{x} &\geq \boldsymbol{0}, && (3.10) \end{aligned}$$

where $\boldsymbol{0}$ is a column vector containing n zeros. In 1947, Dantzig [33] developed a method, which is efficient on expectation and is called the simplex algorithm, for solving LP problems in their standard form. Since $n > m$, a unique solution satisfying (3.9) cannot be found. Instead, $n - m$ variables are assigned the value zero, allowing for a unique solution to be found for the remaining m variables, called a *basic solution*, if the rank of $\boldsymbol{A}$ is m. The $n - m$ variables that are assigned the value zero are called *non-basic variables* and the remaining m variables with potentially non-zero values are called *basic variables*. A basic solution therefore depends on the choice of non-basic variables. Any basic solution in which all decision variables are nonnegative (*i.e.* $x_j \geq 0$ for all $j = 1, \ldots, n$) is called a *basic feasible solution* (BFS). A pseudo-code description of the simplex algorithm for solving an instance of (3.8)–(3.10) is given in Algorithm 3.1.

Algorithm 3.1: Simplex algorithm (maximisation)

```
Input   : An LP problem instance in standard form.
Output: An optimal solution to the LP problem instance (if it exists).
1  Construct the simplex tableau from the standard form of the LP problem instance.
2  Compute the $j$-th entries of the $g$-row and $z$-row of a new simplex tableau of the format
   shown in Table 3.1 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      The current solution is optimal — output this solution and stop.
5  if $a_{iq} \leq 0$ for all $i$ then
6      The objective function is unbounded — stop.
7  Identify $z_q$ as the most negative value in the $z$-row. The corresponding column $q$ is called
   the pivot column and determines the non-basic variable that enters the basis.
8  Compute the ratios in the $\theta$-column as $\theta_i = b_i/a_{iq}$ (only if $a_{iq} > 0$).
9  Identify $\theta_p$ as the smallest $\theta$ ratio. The corresponding row $p$ is called the pivot row and
   determines the basic variable that leaves the basis.
10 Perform a simplex iteration:
      (i)  Divide the values in the pivot row by $a_{pq}$.
      (ii) Calculate the values of new non-pivot rows entries as:
           new non-pivot row$_i$ = old non-pivot row$_i$ $- a_{iq} \times$ new pivot row.
11 Return to Step 2.
```

The simplex algorithm iteratively explores adjacent BFSs, each represented in a so-called simplex tableau format, as illustrated in Table 3.1, by ejecting a single basic variable per iteration from the basis (set of basic variables) and replacing it with a non-basic variable so as to obtain a different basic solution that again satisfies the constraint set in (3.9). The process of transitioning from one BFS to another involves performing Gauss-Jordan elimination to solve for the basic variables, and is summarised in the tableau form indicated in Table 3.1 during each algorithmic iteration. The objective function value of the BFSs thus generated during the execution of the simplex algorithm improves from each BFS generated to the next (*i.e.* from one tableau to the next), until the objective function value cannot improve any further and an optimal BFS has been reached. This latter BFS is returned by the algorithm as an optimal solution to (3.8)–(3.10).

Table 3.1: *The simplex tableau format assumed in this thesis.*

		Vector $\boldsymbol{c}$ Vector $\boldsymbol{x}$	RHS	θ
Coefficients	Basic variables	Matrix $\boldsymbol{A}$	Vector $\boldsymbol{b}$	Ratios
	g	g-row	**Objective function**	
	z	z-row		

The LP relaxation[3] of the problem instance in (3.1)–(3.4) may, for example, be written in the standard form presented in (3.5)–(3.7) by introducing non-negative slack variables s_1 and s_2 in order to replace the inequality constraints with equality constraints. The objective in this relaxation is therefore to

$$\text{maximise} \quad z = 8x_1 + 5x_2 \tag{3.11}$$

subject to the constraints

$$\begin{aligned} x_1 + x_2 + s_1 &= 6, & (3.12)\\ 9x_1 + 5x_2 + s_2 &= 45, & (3.13)\\ x_1, x_2, s_1, s_2 &\geq 0. & (3.14) \end{aligned}$$

The feasible region[4] of the problem instance in (3.11)–(3.14) is illustrated graphically in Figure 3.1.

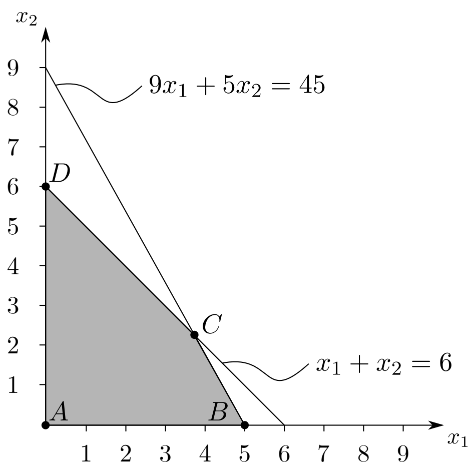


Figure 3.1: *The feasible region of the LP problem instance in (3.11)–(3.14).*

[3]The LP relaxation of an integer programming problem instance is obtained by relaxing all of the integer variable constraints in the original integer programming problem to constraints involving non-negative, continuous variables.

[4]The feasible region of an LP problem is the set of all points that satisfy all of the constraints and sign restrictions in (3.9)–(3.10) [144].

In order to demonstrate the working of Algorithm 3.1, it is applied to the the problem instance (3.11)–(3.14). First, since $n = 4$ and $m = 2$, a total of $n - m = 2$ variables, s_1 and s_2, are chosen as basic variables. After having completed Steps 1–2, the following simplex tableau is generated:

		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	$\leftarrow$
	g	0	0	0	0	0.		
	z	-8	-5	0	0			
		$\uparrow$						

The BFS corresponding to this simplex tableau is $x_1 = 0$, $x_2 = 0$, $s_1 = 6$, and $s_2 = 45$ with an objective function value of $z = 0$. According to the condition in Steps 3–4, this solution is not optimal, since not all z_j-values are non-negative. The pivot column corresponds to the column of the non-basic variable x_1, since z_1 is the most negative value in the z-row (as per Step 7). Furthermore, the pivot row corresponds to the basic variable s_2, since θ_2 is the smallest ratio in the θ column (as per Step 9). The pivot row and the pivot column are indicated by arrows in the tableau above. According to the condition in Steps 5–6, the objective function is not unbounded and the basic variable s_2 is replaced with the non-basic variable x_1 in order to find the next BFS. A simplex iteration is performed (as per Step 10) and the following simplex tableau is generated:

		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	$\frac{8}{9}$	40.		
	z	0	$-\frac{5}{9}$	0	$\frac{8}{9}$			
			$\uparrow$					

The BFS corresponding to this simplex tableau is $x_1 = 5$, $x_2 = 0$, $s_1 = 1$, and $s_2 = 0$ with an objective function value of $z = 40$, which is strictly better than the objective function value of the previous BFS. Once again, this BFS is not optimal, since not all z_j-values are non-negative. The pivoting column corresponds to the column of the non-basic variable x_2 and the pivoting row corresponds to the row of the basic variable s_1. The basic variable s_1 is therefore replaced with the non-basic variable x_2. Upon performing another simplex iteration, the following simplex tableau is generated:

		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	$\frac{5}{4}$	$\frac{3}{4}$	$\frac{165}{4}$.
	z	0	0	$\frac{5}{4}$	$\frac{3}{4}$	

The BFS corresponding this simplex tableau is $x_1 = \frac{15}{4}$, $x_2 = \frac{9}{4}$, $s_1 = 0$, and $s_2 = 0$ with an objective function value of $z = \frac{165}{4}$, which is once again strictly better than the objective function value of the previous BFS. Since all z_j-values are non-negative, it is concluded that this BFS is an optimal BFS (as per Steps 3–4). The BFSs found in the first, second, and third simplex tableaus correspond to the extremal points A, B, and C, respectively, in Figure 3.1.

3.1.2 The branch-and-bound method

In an LP problem, all decision variables are non-negative continuous variables. An *integer programming* (IP) problem, on the other hand, is an LP problem in which some of the decision variables are required to assume non-negative integer values [144]. In a *pure IP* problem, all decision variables are required to assume integer values, whereas in a *mixed IP* problem, only some of the decision variables are required to assume integer values. The LP relaxation of an IP problem instance is a less constrained version of the IP problem instance, with the property that the

$$\text{optimal objective value of LP relaxation} \geq \text{optimal objective value of IP problem instance.}$$

Therefore, an optimal solution to an IP problem instance is at most as good as an optimal solution to its LP relaxation. If an optimal solution to the LP relaxation is known, an upper bound on the optimal objective function value of the IP problem instance is therefore available. Furthermore, if the decision variables in an optimal solution to the LP problem assume integer values, then the solution is also an optimal solution to the corresponding IP problem instance, and so the objective function value of the LP problem instance in this case represents a lower bound on the optimal objective function value of the corresponding IP problem instance [144].

The branch-and-bound method, proposed in 1960 by Land and Doig [84], is a popular method for solving IP problem instances. An early application of the branch-and-bound method for solving VRP instances was proposed by Christofides and Eilon [23] in 1969. When adopting this method, a so-called branch-and-bound tree is constructed to represent the progression of the method visually. A node in the tree represents a sub-problem of the original LP relaxation to which certain branching constraints have been added. Branching on a sub-problem leads to two different sub-problems. If an optimal solution to a sub-problem contains a decision variable x which is, in fact, required to assume an integer value, but assumes a non-integer value with an integer part a and a fractional part b, the additional constraints $x \leq a$ and $x \geq a + 1$ are imposed on the two sub-problems, respectively.

When further branching on a sub-problem cannot lead to any further improvement in objective function value, the branch is considered *fathomed*. A sub-problem is deemed fathomed if (1) the sub-problem is infeasible, (2) all the decision variables in an optimal solution to the sub-problem assume integer values, in which case a lower bound is obtained on the optimal objective function value of the original IP problem instance, or (3) the objective function value corresponding to the optimal solution of the sub-problem is smaller than the best-known lower bound (in which case it is certain that all solutions yielded by further branching on the sub-problem will not lead to an improvement upon the current best solution) [144].

The traversal progression through the sub-problems in the branch-and-bound tree is governed by a so-called *search protocol*. All search protocols return optimal solutions, although some are more efficient than others when utilised in certain contexts. In the *last-in-first-out* (LIFO) protocol, for instance, the last sub-problem generated is always selected to be branched upon next. According to the *jumptracking* protocol, on the other hand, all sub-problems are generated and the sub-problem corresponding to the best objective function value uncovered is selected to be branched upon next [144].

The working of the branch-and-bound method in conjunction with the LIFO search protocol is demonstrated in the remainder of this section for the example problem instance (3.1)–(3.4) and the resulting branch-and-bound tree is illustrated graphically in Figure 3.2.

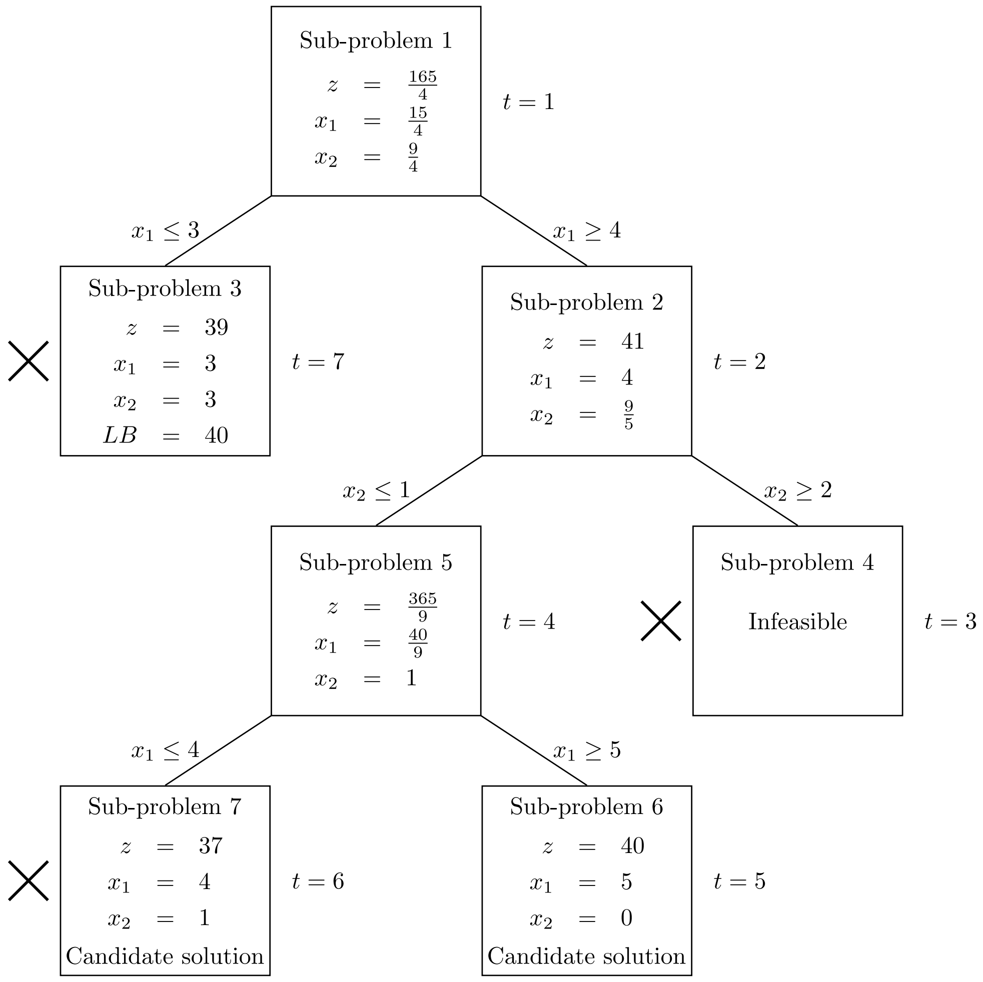


FIGURE 3.2: *An example of the branch-and-bound tree obtained when adopting the LIFO search protocol to solve the IP problem instance in (3.1)–(3.4). The order in which sub-problems are considered are indicated by the t-values next to each sub-problem [144].*

The branch-and-bound method begins by solving the LP relaxation of the IP problem instance, represented by the root node of the tree (Sub-problem 1 in Figure 3.2). The optimal solution to Sub-problem 1 is $x_1 = \frac{15}{4}$, $x_2 = \frac{9}{4}$, which yields an objective function value of $z = \frac{165}{4}$ (as determined in §3.1.1, using the simplex algorithm). Since the two decision variables, x_1 and x_2, do not assume integer values, x_1 is arbitrarily chosen to branch upon. Since the optimal objective function value of the LP relaxation is $z = \frac{165}{4}$, it is known that the optimal objective function value of the IP problem instance cannot exceed $z = \frac{165}{4}$, thereby establishing an upper bound on the optimal objective function value of the IP problem instance. Branching on the decision variable x_1 yields the following two sub-problems:

Sub-problem 2: Sub-problem 1 + Constraint $x_1 \geq 4$, and

Sub-problem 3: Sub-problem 1 + Constraint $x_1 \leq 3$.

Since neither of these sub-problems allows for the possibility that $x_1 = \frac{15}{4}$, the optimal solution to the sub-problem corresponding to the root node of the tree cannot reoccur. In the optimal solution to Sub-problem 2, the decision variable x_1 assumes an integer value, but not the decision variable x_2. The decision variable x_2 is therefore chosen to branch upon next, yielding the following two sub-problems:

Sub-problem 4: Sub-problem 2 + Constraint $x_2 \geq 2$, and

Sub-problem 5: Sub-problem 2 + Constraint $x_2 \leq 1$.

Sub-problem 4 is infeasible and therefore cannot yield an optimal solution to the IP, indicated by the $\times$ in Figure 3.2. In the optimal solution to Sub-problem 5, the decision variable x_1 does not assume an integer value. Branching on this decision variable yields the following two sub-problems:

Sub-problem 6: Sub-problem 5 + Constraint $x_1 \geq 5$, and

Sub-problem 7: Sub-problem 5 + Constraint $x_1 \leq 4$.

In the optimal solution to Sub-problem 6, both decision variables assume integer values, yielding an objective function value of $z = 40$. The solution to Sub-problem 6 is therefore a candidate solution and yields a lower bound on the optimal objective function value of the original IP problem instance. Following the LIFO search protocol, Sub-problem 7 is branched upon next, since Sub-problem 6 cannot further be branched upon and Sub-problem 7 is the sub-problem last generated. In the optimal solution to Sub-problem 7, all decision variables once again assume integer values, but correspond to an objective function value of $z = 37$ which is less than the lower bound value of 40, found in Sub-problem 6. The last unsolved sub-problem generated, Sub-problem 3, is considered next. In the optimal solution to Sub-problem 3, all decision variables assume integer values, but once again correspond to an objective function value smaller than the known lower bound value of 40, indicating that the branch is fathomed. Since there are no remaining unsolved sub-problems, it is concluded that the optimal solution to the original IP problem instance is the solution found to Sub-problem 6. That is, $x_1 = 5$ and $x_2 = 0$ with an optimal objective function value of $z = 40$.

When solving the example problem instance (3.1)–(3.4) using the branch-and-bound method in conjunction with the jumptracking search protocol, instead of the LIFO search protocol, the branch-and-bound tree illustrated graphically in Figure 3.3 is obtained. Only the root node and the first level of the branch-and-bound tree are illustrated, since the child nodes of Sub-problem 2 are the same as in Figure 3.2. Instead of arbitrarily branching on Sub-problem 2, as was done when adopting the LIFO search protocol, Sub-problems 2 and 3 are first solved, and Sub-problem 2 is selected to branch upon next, since it corresponds to the best objective function value. The optimal solution to the example problem instance (3.1)–(3.4) is again obtained as $x_1 = 5$ and $x_2 = 0$, with an optimal objective function value of $z = 40$.

3.1.3 The cutting plane method

The *cutting plane method*, proposed by Gomory [61] in 1958, is another method for solving IP problems. Many cutting plane methods have been proposed for solving TSP instances, such as the work of Miliotis [98] in 1978 and Fleischmann [48] in 1985, although cutting plane methods are usually employed in conjunction with other methods for solving VRP instances, as described later in §3.1.4. According to this method, cutting planes are iteratively inserted as

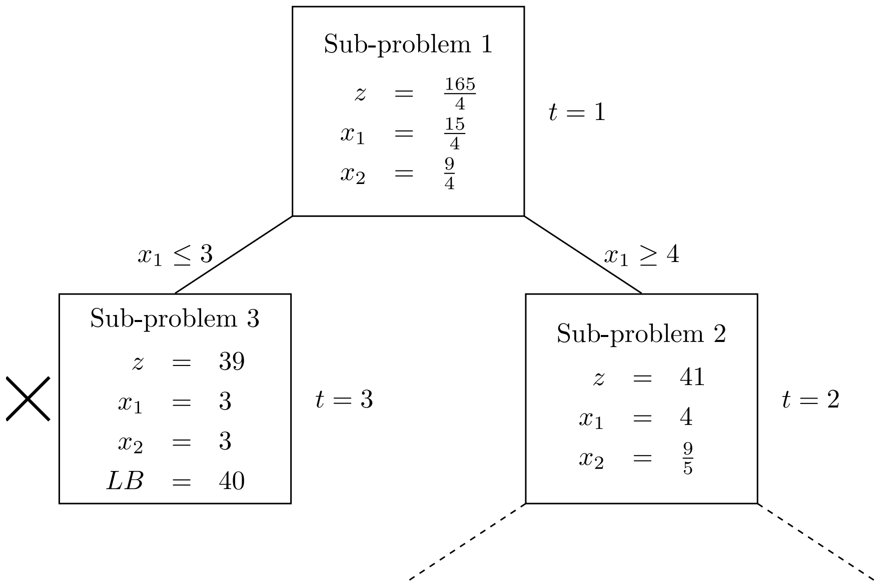


FIGURE 3.3: *A portion of the branch-and-bound tree obtained when adopting the jumptracking search protocol to solve the IP problem instance in (3.1)–(3.4). The order in which sub-problems are considered are indicated by the t-values next to each sub-problem [144].*

additional constraints to the LP relaxation of an IP problem instance, thereby removing portions of the feasible region containing the current optimal solutions and thus resulting in new optimal solutions. If all decision variables in one of these new optimal solutions assume integer values, an optimal solution to the IP problem instance has been found; otherwise the process of inserting new cutting planes continues. A cutting plane has the properties that (1) any feasible solution to the original IP problem instance satisfies the cut and (2) the current optimal solution to the LP relaxation does not satisfy the cut.

Let $[x]$ denote the largest integer not exceeding x. For example, if $x = 3.5$, then $[x] = 3$ while if $x = -1.25$, then $[x] = -2$. Any real value x may be written in the form $[x] + f$, where $0 \leq f < 1$. For example, $3.75 = 3 + 0.75$ while $-1.25 = -2 + 0.75$. Cutting planes are generated by representing the coefficients of decision variables in an LP relaxation constraint in the form $[x] + f$. When generating a cutting plane, the following steps are executed: (1) An equality constraint is arbitrarily selected from the simplex tableau representing an optimal solution to the current LP relaxation, (2) each variable coefficient is rewritten in the form $[x] + f$, as mentioned above, (3) the terms in the constraint are rearranged so that all integer-valued coefficients occur on the left-hand side while all non-integer valued coefficient occur on the right-hand side, and (4) a cutting plane is generated by setting the right-hand side of this constraint equal to zero. The cutting plane method is described in pseudo-code form in Algorithm 3.2.

While any constraint could, in fact, have been selected in Step 2 of Algorithm 3.2 to generate the cut, the selection of a constraint whose right-hand side has a fractional part closest to $\frac{1}{2}$ is expected to remove the largest portion of the LP relaxation feasible domain not containing an integer solution. This is, in turn, anticipated to accelerate the cutting plane method by reducing the number of cuts that have to be generated.

Each cut generated is inserted into the LP relaxation problem instance by adding it as an additional row to the simplex tableau. Since that part of the feasible region within which the previous optimal solution existed is removed when inserting the cut, the dual simplex algorithm is employed to regain a feasible solution. The working of the dual simplex method is similar to that of the (primal) simplex method described in §3.1.1, and involves adapting Algorithm 3.1

Algorithm 3.2: The cutting plane algorithm [144]

```
Input  : An integer programming problem instance for which the linear programming
         relaxation problem instance is in standard form.
Output: An optimal solution to the integer programming problem instance.
1 Find an optimal tableau for the LP relaxation of the IP problem instance by applying the
  simplex algorithm (Algorithm 3.1). If all variables in the optimal solution assume integer
  values, then an optimal solution to the IP problem instance has been found; otherwise,
  proceed to Step 2.
2 Pick a constraint to the linear programming relaxation optimal tableau whose right-hand
  side has a fractional part closest to $\frac{1}{2}$. This constraint is used to generate a cutting plane.
3 For the constraint identified in Step 2, write the coefficient of each variable in the form
  $[x] + f$, for some $0 \leq f < 1$.
4 Rewrite the constraint used to generate the cut as
      All terms with integer coefficients = all terms with fractional coefficients.
5 Then the cut is inserted by imposing the additional constraint
      All terms with fractional coefficients $\leq 0$.
6 Use the dual simplex algorithm to find an optimal solution to the linear program
  relaxation, with the cut inserted as an additional constraint.
```

very slightly. In the dual simplex method, the pivot row is identified before the pivot column (Steps 9 and 7 in Algorithm 3.1, respectively). The pivot row is determined by identifying the basic variable with the smallest value in the right-hand side column of the simplex tableau. Thereafter, the ratio

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

is calculated for each column and inserted as an additional row in the tableau. The pivot column is then selected as the column corresponding to the smallest absolute value in the θ'-row. A normal simplex iteration is then performed (Step 10 of Algorithm 3.1), after which the initial simplex algorithm is followed if an optimal solution has not yet been found.

The working of the cutting plane method is illustrated by means of the example problem instance (3.1)–(3.4). First, the constraint

$$x_1 - 1.25s_1 + 0.25s_2 = 3.75$$

is arbitrarily chosen from the simplex tableau representing the optimal solution to the LP relaxation. The constraint is then rewritten as

$$x_1 - 2s_1 + 0.75s_1 + 0s_2 + 0.25s_2 = 3 + 0.75$$

to represent each coefficient in the form $[x] + f$. The constraint is once again rewritten as

$$x_1 - 2s_1 + 0s_2 - 3 = 0.75 - 0.75s_1 - 0.25s_2$$

by placing all integer-valued coefficients on the left-hand side and all non-integer valued coefficients on the right-hand side. The cutting plane to be added therefore corresponds to

$0.75 - 0.75s_1 - 0.25s_2 = 0$ and is represented by the dashed line in Figure 3.4(a). The cut is imposed by enforcing the additional constraint

$$0.75 - 0.75s_1 - 0.25s_2 \leq 0, \tag{3.15}$$

which is equivalent to $3x_1 + 2x_2 \leq 15$. A slack variable s_3 may be introduced to rewrite the constraint in (3.15) as the equality constraint $-\frac{3}{4}s_1 - \frac{1}{4}s_2 + s_3 = -\frac{3}{4}$. The constraint is inserted into the previously optimal simplex tableau (as per Step 5 of Algorithm 3.2) and the following tableau is obtained:

		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	$-\frac{3}{4}$	$\leftarrow$
	g	8	5	$\frac{5}{4}$	$\frac{3}{4}$	0	$\frac{165}{4}$.	
	z	0	0	$\frac{5}{4}$	$\frac{3}{4}$	0		
	θ'	–	–	$\frac{5}{9}$	3	–		
				$\uparrow$				

The above simplex tableau corresponds to point C in Figure 3.4(b), which no longer forms part of the feasible region of the LP relaxation. Following the dual simplex method, the pivot row is identified as the row corresponding to variable s_3 and the pivot column is identified as the column corresponding to variable s_1, indicated by the arrows in the tableau above. After performing a normal simplex iteration (as per Step 10 of Algorithm 3.1), the following tableau is generated:

		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	$\frac{1}{3}$	$\frac{5}{3}$	40.
	z	0	0	0	$\frac{1}{3}$	$\frac{5}{4}$	

Since there are no negative values in the z-row and all of the basic variables assume non-negative values, an optimal solution has been found. The optimal solution to the original IP problem instance in (3.1)–(3.4) returned by the cutting plane method is therefore $x_1 = 5$, $x_2 = 0$, $s_1 = 1$, $s_2 = 0$ and $s_3 = 0$, which yields an objective function value of $z = 40$.

3.1.4 The branch-and-cut method

The well-known branch-and-cut method is a combination of the branch-and-bound method and the cutting plane method described in the previous sections. Padberg and Rinaldi [105] were the first to apply the concept of generating cutting planes intermittently at the nodes of branch-and-bound trees. The branch-and-cut method involves alternating between branching and inserting cutting planes in order to generate a smaller branch-and-cut tree than the corresponding

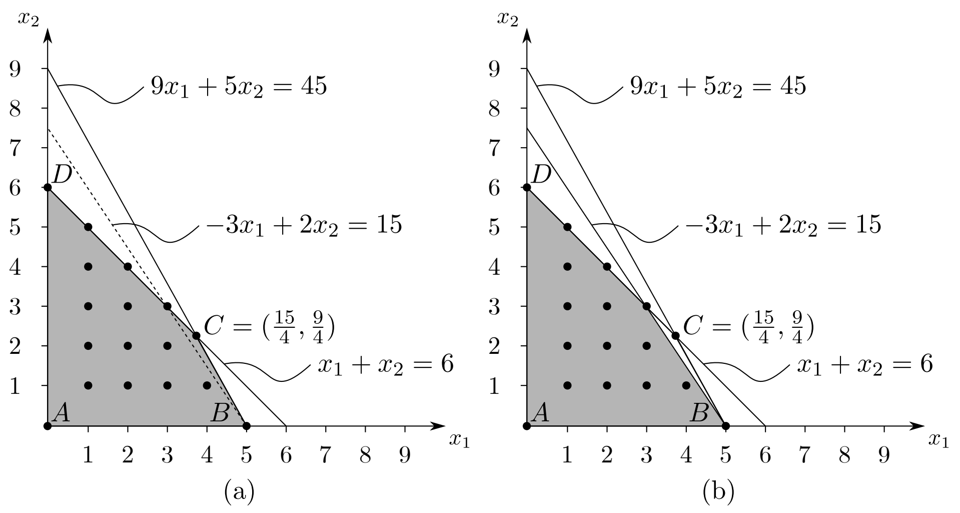


FIGURE 3.4: *An example of (a) including a cutting plane (dashed line) through the feasible region of the problem instance (3.1)–(3.4), thereby (b) reducing the size of the feasible region without eliminating any integer-valued solutions.*

branch-and-bound tree. Araque [3] reported on the solution of a VRP instance containing 48 customers which was solved exactly by means of a branch-and-cut method in 1989. The working of the branch-and-cut method, when employing the jumptracking search protocol, is again demonstrated in the context of (3.1)–(3.4) with the resulting branch-and-cut tree illustrated in Figure 3.5.

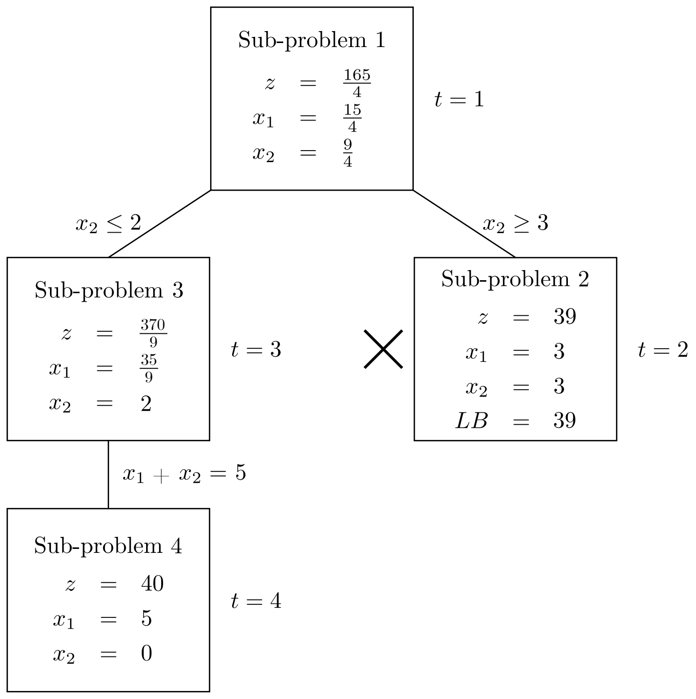


FIGURE 3.5: *An example of a branch-and-cut tree obtained when adopting the jumptracking search protocol to solve the IP problem instance in (3.1)–(3.4). The order in which sub-problems are considered are indicated by the t-values next to each sub-problem and the feasible region of each sub-problem is illustrated graphically in Figure 3.6.*

Similar to the branch-and-bound method, the first sub-problem in the branch-and-cut tree corresponds to the LP relaxation of the original IP problem instance. The feasible regions of the various sub-problems in the branch-and-cut tree are illustrated graphically in Figure 3.6. The solution to Sub-problem 1 corresponds to $x_1 = \frac{15}{4}$ and $x_2 = \frac{9}{4}$, yielding an objective function value of $z = \frac{165}{4}$. The variable x_2 is arbitrarily chosen to branch on, yielding Sub-problems 2 and 3 in Figure 3.5. In the optimal solution to Sub-problem 2, all decision variables assume integer values, yielding a feasible solution to the original IP problem instance with an objective function value of $z = 39$. Sub-problem 2 is therefore a candidate solution and produces a lower bound on the optimal objective function value of the original IP problem instance.

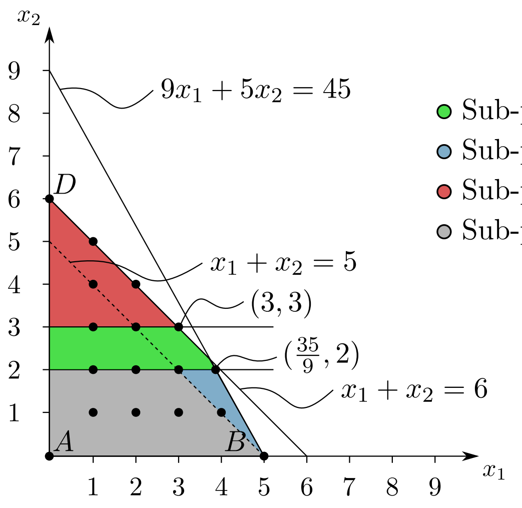


FIGURE 3.6: *The feasible regions of the different sub-problems considered when applying the branch-and-cut method to solve the IP problem instance in (3.1)–(3.4).*

In the branch-and-bound method, the search process would have continued by branching on the decision variable x_1 to expand Sub-problem 3. In the branch-and-cut method, however, the next step is to generate a cutting plane to be inserted as an additional constraint in Sub-problem 3. The simplex tableau corresponding to the optimal solution to Sub-problem 3 is:

		8	5	0	0	0	
		x_1	x_2	s_1	s_2	s_3	RHS
5	s_1	0	0	1	$-\frac{1}{9}$	$-\frac{4}{9}$	$\frac{1}{9}$
8	x_1	1	0	0	$\frac{1}{9}$	$-\frac{5}{9}$	$\frac{8}{9}$
0	x_2	0	1	0	0	1	2
	g	8	5	0	$\frac{8}{9}$	$\frac{5}{9}$	$\frac{370}{9}$.
	z	0	0	0	$\frac{8}{9}$	$\frac{5}{9}$	

Following the cutting plane method, the second constraint is arbitrarily selected and rearranged as $x_1 - s_3 - 3 = -\frac{1}{9}s_2 - \frac{4}{9}s_3 + \frac{8}{9}$, yielding the cutting plane $-\frac{1}{9}s_2 - \frac{4}{9}s_3 + \frac{8}{9} = 0$, which is equivalent to the hyperplane

$$x_1 + x_2 = 5.$$

The constraint $x_1 + x_2 \leq 5$ is therefore inserted as an additional constraint to Sub-problem 3, yielding Sub-problem 4. The dual simplex method is applied to regain feasibility in Sub-problem 4. This solution corresponds to $x_1 = 5$ and $x_2 = 0$, yielding an objective function value of $x = 40$. In the optimal solution to Sub-problem 4, all decision variables assume integer values, yielding a feasible solution to the original IP problem instance with an objective function value of $z = 40$. Since each branch in the branch-and-cut tree is now fathomed, an optimal solution to the IP problem instance in (3.1)–(3.4) has been found and the branch-and-cut process terminates (because the latter solution is better than that corresponding to the previous lower bound). The optimal solution to the IP problem instance in (3.1)–(3.4) returned by the branch-and-cut method is therefore $x_1 = 5$ and $x_2 = 0$ with an objective function value of $z = 40$. The advantage of combining the branch-and-bound method with the cutting plane method is evident when comparing the size of the branch-and-bound tree in Figure 3.2 with that of the corresponding branch-and-cut tree in Figure 3.5.

3.2 Heuristics

As mentioned in §3.1, exact solution approaches are employed to find optimal solutions to optimisation problem instances. Some instances are, however, too complicated and/or large to be solved to optimality within an acceptable time-frame, requiring the use of *heuristics*. A heuristic is a rule-based solution approach which typically returns solutions of a relatively good quality, but which are not necessarily optimal, within a modest time-frame [88, 134]. There are, however, disadvantages associated with using heuristics as a solution approach. First, heuristics are usually not designed to solve a variety of problem instances, but are rather tailored for problems of a specific type. It may therefore be difficult to adapt existing heuristics to solve new problem types. Secondly, heuristics tend to converge towards local optima and may therefore return poor solutions to problem instances having multiple local optima [88].

Heuristic solution approaches for solving VRP instances are often classified as one of three main classes of heuristics, namely *improvement heuristics*, *constructive heuristics*, or *two-phase heuristics*. A classification tree of heuristics for solving VRPs is illustrated graphically in Figure 3.7.

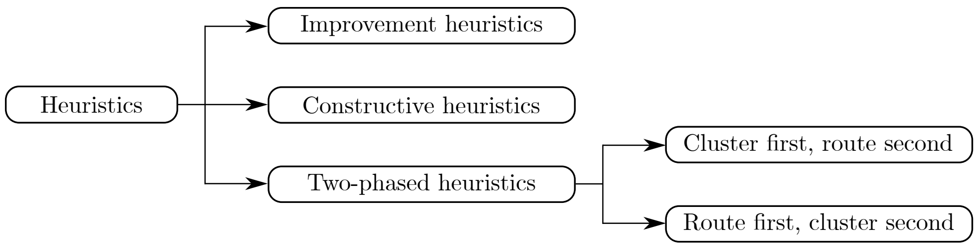


FIGURE 3.7: *A classification tree of heuristic methods for solving VRPs.*

In classical improvement heuristics tailored to solve routing problems (approximately), intra-route and inter-route moves are iteratively applied to form new feasible solutions [137]. The feasible region is iteratively explored to find a solution that is better than the current best solution, and the best feasible solution found is returned when the search terminates [88]. A popular intra-route move is the λ-opt move, in which any λ edges are removed and replaced by λ other edges. An example of a λ-opt move with $\lambda = 2$ is illustrated in Figure 3.8. Two popular inter-route moves are *relocate*, in which k consecutive customers are removed from their current route and inserted into a different existing route, and *swap*, in which k consecutive customers are swapped with ℓ consecutive customers from a different existing route.

Unlike improvement heuristics, constructive heuristics do not improve upon existing feasible solutions, but rather build up feasible solutions iteratively, guided by the objective function [134]. An

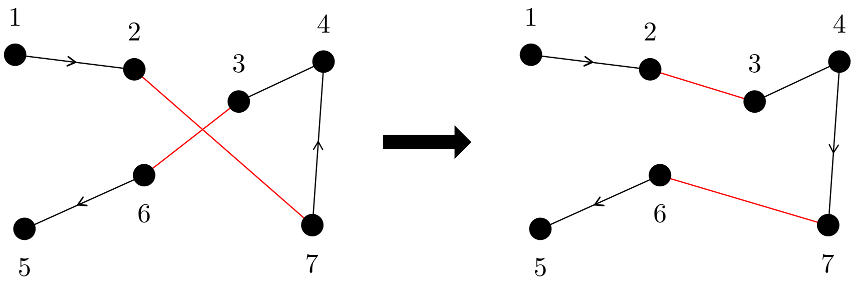


FIGURE 3.8: *A graphical illustration of the 2-opt move applied to a part of a route in a VRP instance. The two edges changed in the move are presented in red.*

example of a constructive heuristic is the savings algorithm, suggested by Clarke and Wright [26] in 1964, applied to a VRP instance in which a fixed number of vehicles are to be used. The savings algorithm starts by creating routes of the form $(0, i, 0)$ for $i = 1, \dots, n$, where 0 denotes the depot and n denotes the number of customers in the problem instance. When two routes $(0, \dots, i, 0)$ and $(0, j, \dots, 0)$ may be merged into a single feasible route $(0, \dots, i, j, \dots, 0)$, a cost saving of $s_{ij} = c_{i0} + c_{j0} - c_{ij}$ is obtained. In the savings algorithm, (1) the cost saving associated with merging each pair of routes is calculated, and (2) the merge associated with the largest cost saving and resulting in a feasible route, is performed. These two steps are iteratively performed until the required number of vehicles are used, after which the final solution is returned.

In a *two-phase* heuristic, the VRP is decomposed into two components, namely the *clustering* of vertices into groups and the actual *route construction* for each of these groups [137]. Two sub-classes of two-phase heuristics exist, namely *cluster-first, route-second* methods, and *route-first, cluster-second* methods. In the former sub-class, the clustering component is performed first and the route construction component is performed thereafter, whereas in the latter sub-class, the opposite order is followed. An example of a popular and simple route-first, cluster-second method is the sweep algorithm, proposed by Wren and Holliday [145] in 1972. In the sweep algorithm, (1) an unused delivery vehicle is chosen, (2) a ray centred at the depot is rotated in polar angle form from the last vertex added (which may be any vertex if the algorithm has not yet been initialised), adding vertices to a cluster as long as the total demand associated with the vertices forming part of the cluster do not exceed the capacity of the delivery vehicle under consideration, and (3) a route is constructed by solving a TSP instance for the vertices forming part of the cluster and assigning them to a delivery vehicle. This process is performed iteratively until each vertex has been included in a route.

In a route-first, cluster-second method, a TSP tour containing all of the customers in the VRP instance is constructed, after which segments of the tour are segmented into feasible routes. Beasly [12] was the first to propose segmentation of a TSP tour into routes using Dijkstra's shortest path-finding algorithm. Route-first, cluster-second approaches are, however, typically not competitive with other heuristics [134].

3.3 Metaheuristics

In order not to converge to a local optimum of a problem instance, a structured approach must be adopted to utilise the information gathered during the search for solutions, in order to guide the search towards a global optimum of the problem instance [88]. A *metaheuristic* is a general solution method that governs the interaction between local improvement procedures and higher-level exploration strategies, in order to perform a robust search of the feasible region of a problem instance [58]. This allows the search process to escape from local optima, thereby hopefully approximating a global optimum. Metaheuristics provide a general structure and strategy guidelines which may be applied to any type of problem, resulting in a more general solution approach than simple heuristics [88]. Furthermore, metaheuristics typically perform a much more thorough search of the solution space than heuristics by allowing for the consideration of non-improving and sometimes infeasible moves, as well as the recombination of current solutions to create new ones [29].

Braekers and Ramaekers [16] performed a taxonomic review of the VRP literature published between the beginning of 2009 and the middle of 2015, and reported that more than 70% of the articles reviewed employed metaheuristics as a proposed solution method. Elshaer and Awad [44] proposed a taxonomy for metaheuristics tailored to solve the VRP and its variants. An adapted classification tree of this taxonomy is illustrated in Figure 3.9. Metaheuristics may be classified as either *trajectory-based* or *population-based*. These two classes differ in the number of candidate solutions maintained during the search process. In trajectory-based methods, a single candidate solution is perturbed during iteration t from a current solution x_t to a new solution x_{t+1} which resides in the neighbourhood $N(x_t)$ of x_t, until a stopping criterion is met. The objective function value $f(x_{t+1})$ of the new solution x_{t+1} is not necessarily better than the objective function value $f(x_t)$ of the previous solution x_t. In trajectory-based methods, it is therefore required to employ techniques to avoid cycling [134]. Trajectory-based methods are exploitation-orientated, since they intensify the search for solutions in promising areas of the feasible region of a problem instance. In population-based methods, on the other hand, a set of candidate solutions is iteratively maintained and improved simultaneously until a stopping criterion is met. Population-based methods are exploration-orientated, since they explore different areas of the feasible region by considering multiple candidate solutions simultaneously.

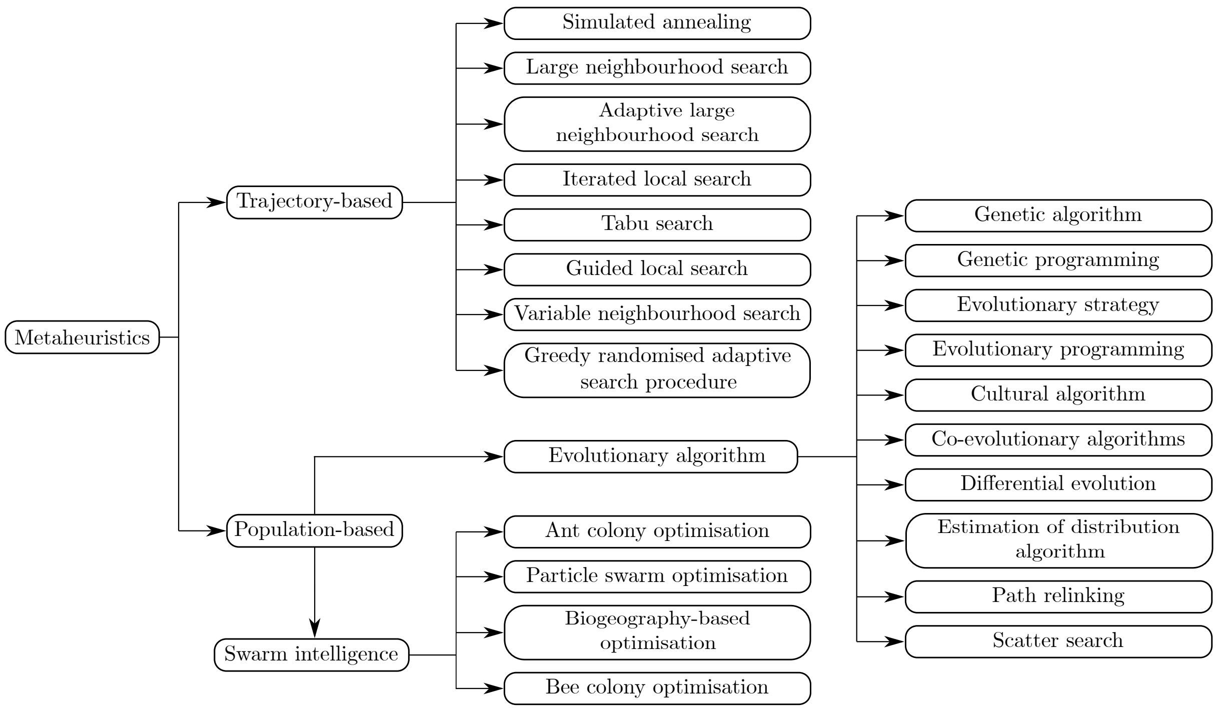


FIGURE 3.9: *A classification tree of metaheuristic methods for VRPs (adapted from [44]).*

3.3.1 Trajectory-based metaheuristics

The principle of *iterated local search* (ILS) was first introduced by Baxter [11] in 1981. According to this metaheuristic, escape occurs form local optima by perturbing a current solution and performing a local search procedure, starting from the perturbed solution. During iteration t, (1) the current solution x_t is perturbed to form an intermediate solution x'_t, (2) a local search procedure is applied, starting from x'_t until a local optimum is reached, and (3) x'_t is either accepted as the new solution used during the next iteration (*i.e.* $x_{t+1} = x'_t$) or rejected (*i.e.* $x_{t+1} = x_t$), according to an acceptance criterion [45]. The perturbation mechanism and acceptance criterion jointly control the trade-off between diversification and intensification during the search [15]. A successful ILS metaheuristic for solving CVRP instances was proposed by Chen *et al.* [22] in 2010. In this metaheuristic, the perturbation strategy is based on the exchange of a number of consecutively visited customers.

The method of *simulated annealing* (SA), proposed by Kirkpatrick *et al.* [78] in 1983, is based on a simulation of the statistical mechanics of annealing in solids to solve combinatorial optimisation problems. In condensed matter physics, annealing is a process during which a solid is melted by increasing its temperature, after which the temperature is gradually lowered in stages until the solid reaches a state of low energy, reducing the hardness and making it more workable [54]. An early implementation of simulated annealing in the context of solving VRP instances was proposed by Robusté *et al.* [116] in 1990. In the method of SA, cycling is avoided by randomly selecting a solution x in $N(x_t)$ during each iteration t. If the objective function value $f(x)$ is better than $f(x_t)$, then $x_{t+1} = x$, otherwise

$$x_{t+1} = \begin{cases} x & \text{with probability } p_t, \\ x_t & \text{with probability } 1 - p_t, \end{cases}$$

where p_t is usually a decreasing function of t and $f(x) - f(x_t)$, often defined as

$$p_t = \exp(-[f(x) - f(x_t)]/\theta_t),$$

where θ_t denotes a temperature parameter during iteration t [137]. During the SA procedure, the temperature parameter is progressively lowered according to a pre-defined cooling schedule. The method of SA exhibits the desired property of asymptotically converging towards the global optimum of a problem instance if allowed enough search time [54].

Tabu search was proposed by Glover [56] in 1986. According to this method, the best solution in the neighbourhood $N(x_t)$ of the current solution x_t is selected as the current solution for the following iteration x_{t+1}, regardless of whether $f(x_{t+1})$ is better than $f(x_t)$, therefore allowing the search to escape from local optima. Furthermore, cycling is avoided by maintaining a short-term memory, called a tabu list. Any solution in $N(x_t)$, which is also in the tabu-list, may not be selected as the new solution for the next iteration. Willard [143] was among the first to propose tabu search for solving VRP instances approximately in 1989.

Feo and Resende [47] proposed the *greedy randomised adaptive search procedure* (GRASP) in 1989 for solving combinatorial optimisation problems. Kontoravdis and Bard [81] proposed a GRASP implementation tailored for solving VRPTW instances in 1995. During each iteration of this procedure, (1) a feasible solution is constructed and (2) a local search procedure is applied to the solution constructed. When constructing a feasible solution, a greedy search heuristic is employed to build up a solution iteratively by randomly selecting the best elements from a set of candidate elements, called the *restricted candidate list*, which is ordered according to a greedy approach [15].

Variable neighbourhood search (VNS), proposed by Mladenovic [101] in 1995, is another trajectory-based metaheuristic in which a single solution x_t is improved by exploring dynamically changing neighbourhoods. VNS was originally proposed for solving combinatorial optimisation problems in general and was illustrated by solving TSP instances. Kytöjoki *et al.* [83] proposed a successful VNS for solving large VRP instances in 2007. This method requires a predefined set of neighbourhood structures, often ordered as a sequence $N_1, N_2, \ldots, N_n$ of neighbourhoods in increasing order of cardinality. Three steps are performed during each iteration, namely *shaking*, *local search*, and *moving*. Starting with a randomly generated initial solution and $n = 1$ during each iteration, (1) shaking is performed during which a solution x'_t is randomly selected from the n^{th} neighbourhood of solution x_t, (2) a local search procedure is performed, starting from the solution x'_t, and (3) if the solution x'_t is better than the current solution x_t, then $x_{t+1} = x'_t$, $n \leftarrow 1$, and the cycle is restarted; otherwise $x_{t+1} = x_t$, the algorithm moves to the next neighbourhood (*i.e.* $n \leftarrow n + 1$), and another iteration is performed. VNS performs well if the neighbourhood structures contain different local optima, allowing the search to escape from one local optimum to another [15].

Guided local search (GLS), proposed by Voudouris [141] in 1997, makes use of a memory structure to change the objective function dynamically. The result is called an *augmented objective function* and is based on local optima found previously during the application of a local search procedure. First, a set of features ft_n is defined for $n \in \{1, \ldots, n_{max}\}$, where n_{max} denotes the number of features. In the case of the VRP, a feature ft_i may represent the presence of an arc from one vertex to another in a candidate solution. Furthermore, a penalty value p_i is associated with each feature ft_i. The augmented objective function f' of a solution x is calculated as

$$f'(x) = f(x) + \lambda \sum_{i=1}^{n_{max}} p_i I_i(x),$$

where $f(x)$ denotes the normal objective function value of solution x, λ is an input parameter to the algorithm, and $I_i(x)$ is a Boolean variable assuming the value one if solution x exhibits feature ft_i, or zero otherwise. Initially, each penalty value is assigned the value zero. When a local optimum is reached upon applying a local search procedure, the penalty values are updated to penalise features incurring a high cost. During future iterations, solutions exhibiting other features will therefore become more attractive, allowing the search to escape from local optima. A large value of λ encourages diversification, whereas a small value of λ intensifies the search in the current area [15]. Kilby *et al.* [77] were among the first to propose a GLS implementation tailored for solving VRPTW instances in 1999.

Large neighbourhood search (LNS) was proposed by Shaw [121] in 1998. During the execution of this trajectory-based metaheuristic, a starting solution is partially destroyed according to a so-called *destructive heuristic* and then rebuilt according to a constructive heuristic, with multiple destructive and constructive heuristics available to choose from during each iteration. In classical LNS metaheuristics, the selection of the destructive heuristic and the constructive heuristic utilised during each iteration is probabilistic. In a variant of LNS, called *adaptive LNS* (ALNS), the selection of destructive and constructive heuristics is, however, determined stochastically, based on the previous performances of the available heuristics during the search. Heuristics that previously yielded good results will have a higher probability of being selected during future iterations [45]. A successful ALNS metaheuristic in the context of solving VRP instances was proposed by Pisinger and Ropke [107] in 2007.

3.3.2 Population-based metaheuristics

A population-based metaheuristic may further be classified as either an *evolutionary algorithm* (EA) or as exhibiting *swarm intelligence* (SI). EAs are based on the concept of species evolving and adapting to their environments over time, whereas SI methods are based on the behaviour of elements in decentralised and self-organised systems, such as ant colonies, flocks of birds, or schools of fish [10].

Evolutionary algorithms

The notion of an *evolutionary strategy* (ES), proposed by Reschenberg in 1965 [112], was inspired by Darwin's theory of natural selection [35]. The principle of natural selection is imitated by means of a parent selection and mutation process. The first ES, called a *two-membered ES*, was proposed as an approach towards parameter optimisation [15]. In a two-membered ES, the population consists of a single parent producing a single offspring with the better solution between the two forming the population during the next iteration. Further development of the notion of a two-membered ES resulted in *multi-membered ESs*, consisting of two versions proposed by Schwefel [120] in 1981, the $(\mu+\lambda)$-ES and the (μ, λ)-ES, incorporating both mutation and a recombination operators. In a $(\mu+\lambda)$-ES, parents are selected from a population consisting of μ individuals to generate $\lambda \geq 1$ offspring by means of recombination and mutation, after which the λ worst solutions are discarded in order to maintain a population size of μ individuals during each iteration. In a (μ, λ)-ES, where $\lambda > \mu$, λ offspring are generated, after which the μ best offspring form the population during the next iteration. A solution x is encoded by the decision variables of the optimisation problem instance under consideration as $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, where n denotes the number of decision variables in the problem instance. According to the mutation operator, a mutation value $z_i = N(0, \sigma)$ is drawn from a normal distribution with zero mean and standard deviation σ for $i = 1, \ldots, n$, with σ called the mutation step size. For each decision variable i in a solution encoding, the mutation value z_i is added to x_i to form an offspring y. The mutation step size σ may be constant over time or it may be adjusted dynamically, based on the number of iterations or feedback received during the search process [15]. A popular method for adjusting the mutation step size is the 1/5 success rule. This rule states that if more than one out of the last five offspring generated resulted in a success, the mutation step size is increased to diversify the search, whereas if none of the last five created offspring resulted in a success, the mutation step size is decreased in order to intensify the search; otherwise no change is made [15]. Homberger and Gehring [70] proposed two successful ESs tailored for solving VRPTW instances in 1999.

A class of metaheuristics similar to ESs is *Evolutionary programming* (EP), proposed by Fogel *et al.* [49] in 1966. EP does not, however, include a recombination operator when generating offspring. In EP, parent selection is probabilistic and mutation is the only operator used to generate offspring. Solutions are also encoded by their decision variable values and the same mutation operator is applied as in ESs. Due to its similarity to ES, EP is not employed as often [15]. In 1997, Porto and Fogel [108] implemented EP for optimising the behaviour of a fleet of military vehicles, which included the routing of these vehicles.

One of the most popular methods in the class of EAs is the *genetic algorithm* (GA), also inspired by Darwin's theory of natural selection [35]. Since Holland [69] proposed the first GA in 1975, many variations tailored for specific problem types have been proposed. The work of Prins [110] in 2004 lead to a breakthrough in the success of using GAs for solving VRP instances [137]. A pure GA is a generic solution paradigm that allows for relatively easy adaptation to solve a wide variety of problem types [54]. According to this paradigm, a population of individuals, representing candidate solutions to an optimisation problem encoded as strings, evolves from one generation to the next as a result of operations simulating the process of natural selection. The process starts with an initial population of individuals which is usually generated randomly. During each iteration, (1) two parents are selected stochastically according to a bias based on the objective function of the problem instance, (2) offspring are generated by invoking a crossover procedure during which the genetic features of the parents are combined to form the genetic features of the offspring, (3) mutations occasionally occur within the offspring, and (4) the offspring and a subset of the current population are retained to form the new population considered during the next iteration. These steps are iterated until a stopping criterion is met and the best feasible solution found throughout the search process is returned [88]. Due to the considerable success of GAs when solving single objective optimisation problems, they have been adapted for solving multi-objective optimisation problems as well. The notion of *genetic programming* (GP), proposed by Koza [82] in 1992, is an EA similar to the GA, although solutions are not encoded as strings but rather as computer programs which may be executed [15]. These programs are represented as syntax trees which undergo the same operations as those utilised in GAs.

Scatter search (SS) and *path relinking* (PR) were proposed by Glover [55] in 1977. An early application of SS and RR for solving VRP instances in particular was proposed by Rochat and Taillard [117] in 1995. During both SS and PR, a subset of the population of individuals is selected to form a set of reference individuals. During each iteration, (1) new individuals are created from combinations of the individuals in the reference set, (2) these solutions are modified according to a heuristic procedure, and (3) the reference set is updated by including some of these modified solutions. The reference set is typically relatively small compared to the size of populations in classical EAs. Furthermore, the selection criteria for individuals to be included in the reference set not only depends on the objective function values of solutions, but also on the diversity contribution of each solution to the reference set. SS and PR differ in the way that new individuals are generated. In SS, new solutions are generated as linear combinations of subsets of the reference solutions in Euclidean space, and the heuristic applied to each individual often serves the purpose of yielding integer values for integer-constrained variables in the solution. In PR, on the other hand, new solutions are generated as combinations of subsets of the reference solutions in the neighbourhood space. Generating solutions in the neighbourhood space leads to new solutions on the path from the initiating solution and the guiding solution. Generating solutions in Euclidean space leads to solutions within and beyond solutions in the neighbourhood space [57].

Co-evolutionary algorithms (CoEAs), proposed by Hillis [68] in 1990, are inspired by the phenomenon in nature where the evolutions of two species influence each other, a notion called co-evolution. Examples include predators and prey or insects and the flowers that they pollinate [15]. In contrast to other EAs, the fitness of an individual is not determined independently of those of other individuals in the population, but rather as a function of its interactions with other individuals. Two categories of CoEAs exist, namely *cooperative co-evolution* and *competitive co-evolution*. Cooperative co-evaluation involves simulating the mutually beneficial relationships between species, such as the relationship between insects and the flowers that they pollinate. In these types of CoEAs, the problem is often decomposed into a collection of smaller, easier sub-problems, each assigned to a population. Individuals therefore represent potential components of a larger solution and the evolution of their populations occur simultaneously, but in an isolated fashion, with interaction only taking place to determine fitness values. The way in which a problem is decomposed may either be static (*i.e.* remain fixed for the entire duration of the algorithm) or dynamic (*i.e.* change over time). Competitive co-evolution, on the other hand, involves simulating opposing relationships between species, such as between predators and prey. In such a CoEA, the fitness of an individual is evaluated in terms of its competition with other individuals. Changes in one individual may result in changes in another individual competing for a better fitness value, therefore resulting in higher-quality individuals emerging over time. In 2002, Machado *et al.* [93] implemented a cooperative CoEA specifically for solving VRP instances.

Cultural algorithms (CAs) are a class of evolutionary algorithms based on the cultural evolution process observed in nature [15]. Reynolds [115] proposed the first CA in 1994 by developing an optimisation approach according to which cultural evolution is interpreted as both a micro-evolutionary process (transmission of genetic material between individuals in a population) and a macro-evolutionary process (knowledge gained from individual experiences). Ma *et al.* [91] applied a CA in the context of vehicle routing in 2008. A CA comprises three components, which are (1) the population of individuals that represent the micro-evolutionary process, consisting of operations often included in GAs such as generating offspring, mutation, and survivor selection, (2) a belief space, which represents the macro-evolutionary process, and stores the knowledge gained by all individuals in the population, and (3) the communication protocol that governs the communication between the population and the belief space. During each iteration of a CA, all individuals in the population are evaluated, after which some are allowed to contribute their knowledge to the belief space. Offspring are then generated by invoking various operators with a view to achieve desirable behaviours stored in the belief space. Finally, individuals are selected to form the generation during the next iteration.

Estimation of distribution algorithms (EDAs), proposed by Mühlenbein and Paaß [102] in 1996, are based on probabilistic models. The crossover and mutation operators of traditional EAs are replaced by (1) the estimation of the probability distribution of selected individuals, and (2) the generation of a new population by sampling from this probability distribution. New solutions are then included in the population for use during the next generation, leading the search to promising areas of the feasible region. The probabilistic models employed in EDAs depend on the problem being solved [15]. Pérez-Rodríguez and Hernández-Aguirre [106] proposed a successful hybrid EDA for solving VRPTW instances in 2019.

In *differential evolution* (DE), originally proposed by Storn and Price [125] in 1997 for solving continuous optimisation problem instances, a *mutant individual* is created from a *base individual* in the population. A set of *target individuals* are then utilised, each creating an offspring with the mutant individual by invoking a crossover procedure to generate a set of *trial individuals*. Each trial vector is then compared with its corresponding target individual and the best individual is retained in the population. Variants of this paradigm of EAs are conventionally named $DE/x/y/z$, where x denotes the way in which the base individual is selected (for example, the best individual or randomly), y denotes the number of target individuals selected from the population, and z denotes the crossover procedure to be used [109]. An advantage of DE algorithms is that only three input parameters control the search process. Similar to GAs, DE algorithms have achieved considerable success in solving single-objective optimisation problems and have therefore also been adapted for solving multi-objective optimisation problems as well [97]. Mingyong and Erbao [100] were the first to propose a DE implementation specifically for solving VRP instances in 2010.

Swarm intelligence

A class of swarm intelligence metaheuristics inspired by the communication and cooperation mechanisms observed between ants, is *ant colony optimisation* (ACO), proposed by Dorigo *et* *al.* [40] in 1991. ACO was originally proposed for solving TSP instances, after which Kawamura *et al.* [75] proposed an ACO algorithm for solving VRP instances in 1998. When searching for food, an ant releases a chemical compound, called a *pheromone*, along its path [54]. The amount of pheromone along a path depends on the length of the path and the quality of the food source. Ants are attracted to paths along which more pheromone has been released. They then, in turn, release more pheromone along the same paths, resulting in an efficient reinforcement procedure for ant colonies to procure food [134]. In the context of routing problems, new solutions are generated by means of a savings-based procedure and local search. Instead of utilising a savings value of $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, as was done in the savings algorithm of Clarke and Wright [26], an attractiveness value $\chi_{ij} = \tau_{ij}^{\alpha} - s_{ij}^{\beta}$ is used. The pheromone value τ_{ij}^{α} is a representation of the success associated with combining vertices i and j during previous iterations, and α and β are input parameter values to the algorithm. The combination of vertices i and j takes place with probability $p_{ij} = \chi_{ij}/(\sum_{(h,l)\in\Omega_k} \chi_{hl})$, where Ω_k is the set of feasible (i, j) combinations yielding the k best savings. The probability of combining i and j therefore increases as the pheromone value τ_{ij}^{α} increases, just as the probability of an ant following a specific path increases when more pheromone is released along that path.

Particle swarm optimisation (PSO), proposed by Kennedy and Eberhart [76] in 1995, involves simulating the flocking behaviour of birds to solve optimisation problems. Chen *et al.* [21] proposed a discrete PSO algorithm for solving CVRP instances in 2006. In the PSO algorithm, many particles, each representing a candidate solution to an optimisation problem instance under consideration, are randomly generated, and each particle i is associated with a location $\overrightarrow{X_i}$ in the search space as well as a velocity $\overrightarrow{V_i}$. Furthermore, each particle i is connected to a subset of the total population of particles, called its neighbourhood. Moreover, each particle i is equipped with a memory in which the best location uncovered by the particular particle throughout its search $\overrightarrow{P_i}$, as well as the best location found by any particle in its neighbourhood $\overrightarrow{P_g}$, is stored. During iteration t, the location $\overrightarrow{X_i}$ and the velocity $\overrightarrow{V_i}$ of each particle i is perturbed in each dimension d of the search space, and the memory of each particle is updated. The velocity of a particle i determines the distance and direction with which it should be perturbed, calculated as

$$V_{id}(t + 1) = V_{id}(t) + C_1\phi_1(P_{id}(t) - X_{id}(t)) + C_2\phi_2(P_{igd}(t) - X_{id}(t)),$$

where ϕ_1 and ϕ_2 are uniformly generated numbers in the range [0,1], and C_1 and C_2 are constants, called *acceleration coefficients*. The updated location of a particle i is then calculated as

$$X_{id}(t + 1) = X_{id}(t) + V_{id}(t + 1).$$

The basic PSO algorithm, however, often converges to a local optimum of the problem instance under consideration [15].

Bee colony optimisation (BCO), first proposed by Lučić and Teodorović [89] in 2001, is a class of optimisation algorithms based on imitations of various behaviours of honey bee colonies, such as *waggle dancing*, *food foraging*, the role of the *queen bee*, *task selection*, *mating*, and many more [74]. Lučić and Teodorović [90] implemented BCO for solving stochastic VRP instances in 2003. Some of the most widely used BCO algorithms include the *marriage in honey bees optimisation algorithm* [1] proposed in 2001 in which the mating and marriage of the queen bee in beehives are simulated, the *discrete bee dance algorithm* [62] proposed in 2003 in which the waggle dance of bees for communicating food locations is simulated, and the *queen bee evolutionary algorithm* [73] also proposed 2003 in which the structure of a bee hive is simulated. A survey of algorithms simulating bee swarm intelligence was performed by Karaboga and Akay [74].

Biogeography-based optimisation (BBO), proposed by Simon [122] in 2008, is an optimisation approach inspired by MacArthur and Wilson's [92] equilibrium theory of island biogeography. According to the equilibrium theory of island biogeography, the number of species inhabiting an island depends on the balance between new species immigrating to the island and established species emigrating from the island. According to the basic BBO algorithm, a set of candidate solutions, called islands, share information with each other. Each island is associated with a *habitat suitability index*, representing the fitness of a candidate solution to the optimisation problem instance under consideration, as well as an *immigration rate* λ and an *emigration rate* μ. The immigration rate λ_i and the emigration rate μ_i of island i may be calculated as

$$\lambda_i = I\left(1 - \frac{S_i}{S_{max}}\right) \text{ and } \mu_i = E\left(\frac{S_i}{S_{max}}\right),$$

where I denotes the maximum immigration rate (when no species is established on an island), E denotes the maximum emigration rate (when all possible species, denoted by S_{max}, are established on an island), and S_i denotes the number of species currently established on island i. Islands hosting many established species (good solutions) therefore often share their features with islands hosting fewer established species (poor solutions), in order to improve candidate solutions and thus approximate a global optimum. In 2011, Ergezer and Simon [46] proposed a framework for applying BBO in the specific context of solving discrete combinatorial optimisation problems, such as the VRP.

3.4 A comparison of metaheuristics

Many metaheuristics have been proposed for solving VRP instances approximately, with most of the powerful ones often being hybridisations of stand-alone techniques proposed earlier [137]. Toth and Vigo [137] compared a set of successful metaheuristics in terms of their solution quality and computational time. The metaheursitics considered in their comparison, along with appropriate references and the algorithmic approaches adopted, are summarised in Table 3.2. The comparison was based on the reported results of these metaheuristics when solving two widely used benchmark data sets for the VRP. The first data set, proposed by Christofides *et al.* [24], is referred to as the CMT data set and contains fourteen instances ranging in size from 50 to 200 customers. The second data set, proposed by Golden *et al.* [60], is referred to as the GWKC data set and contains twenty instances involving 240 to 483 customers. The locations of customers in the GWKC instances are distributed spatially according to symmetric patterns, whereas the CMT instances have more realistic locations of customers. When evaluating the computational time and solution quality of the different metaheuristics in the context of solving the CMT instances, the computational time varied significantly, although relatively good solution quality was achieved by all metaheuristics. When solving the GWKC instances, on the other hand, a larger spread in solution quality was observed and even new best solutions were found. The GWKC instances may therefore be used to compare the computational time and solution quality of the metaheuristics.

The run times reported by the authors of the metaheursitics were normalised to match the corresponding run time on an Intel Core i7 CPU operating at 2.93 GHz. The run time required and average gap associated with the best-known solutions to the GWKC benchmark instances are plotted in Figure 3.10 for each of the metaheuristics listed in Table 3.2. Metaheuristics for which more than one configuration have been published are indicated by an “A”, “B” or “C” after the identifier of the metaheuristic, with “A” denoting the most powerful configuration. The two configurations marked in red in Figure 3.10, represent the HGSADC algorithm, proposed by

TABLE 3.2: *The set of metaheuristics included in the comparison carried out by Toth and Vigo [137].*

Heuristic	Reference	Approach
CLM01	Cordeau *et al.* [30]	Tabu Search
TV03	Toth and Vigo [136]	Granular Tabu Search
RDH04	Reiman *et al.* [113]	Ant Colony Optimisation
T05	Tarantilis [132]	Adaptive Memory + Tabu Search
MB07	Mester and Bräysy [96]	Evolutionary Algorithm + Evolutionary Local Search
PR07	Pisinger and Ropke [107]	Adaptive Large Neighbourhood Search
NB09	Nagata and Bräysy [103]	Hybrid Genetic Algorithm
P09	Prins [111]	Greedy Randomise Adaptive Search Procedure + Evolutionary Local Search
GGW10	Groër *et al.* [63]	Route-to-Route + Evolutionary Computation
ZK10	Zachariadus and Kiranoudis [146]	Guided Local Search + Tabu Search
GGW11	Groër *et al.* [64]	Parallel Route-to-Route
CM12	Cordeau and Maischberger [31]	Parallel Iterated Tabu Search
JCL12	Jin *et al.* [71]	Parallel Cooperative Tabu Search
VCGLR12	Vidal *et al.* [139]	Hybrid Genetic Algorithm
SUO13	Subramanian *et al.* [127]	Set Partitioning + Iterated Local Search

Vidal *et al.* [140] in 2012, and an improved version of this algorithm published in 2013, denoted by VCGLR12-B and VCGLR12-A, respectively. Both of these configurations are non-dominated in terms of their solution quality and run time compared with other metaheuristics. Moreover, these algorithmic configurations appear on the elbow of the Pareto front in Figure 3.10 which represents an effective trade-off between relatively small optimality gaps achieved upon having expended relatively limited computational resources. The 2013 algorithmic version achieved the best solution quality of all metaheuristics that have a normalised run time of less than ten thousand seconds. Furthermore, the HGSADC algorithm is known to be able to solve a wide variety of VRP variants such as the PVRP, the MDVRP, the VRPTW, or any combination of these variants, and it can easily be adapted to solve other VRP variants. For this reason, the HGSADC algorithm was selected as solution methodology for the VRP models proposed in this thesis.

3.5 The HGSADC algorithm

In 2013, Vidal *et al.* [140] proposed the HGSADC algorithm for solving a large class of VRPs with time-windows approximately. This algorithm is similar to the one proposed in 2012 by Vidal *et al.* [139], but includes an additional decomposition phase aimed at effectively addressing large problem instances. The later algorithm also includes new move evaluation techniques in the neighbourhood search phase. Moreover, the algorithm reportedly outperforms all current state-of-the-art algorithms on benchmark instances representing any combination of periodic, multi-depot, site-dependent, and duration-constrained VRPs with time-windows. It combines the exploration breadth of genetic algorithms with the improvement capabilities of local search metaheuristics, and employs powerful population-diversity management schemes.

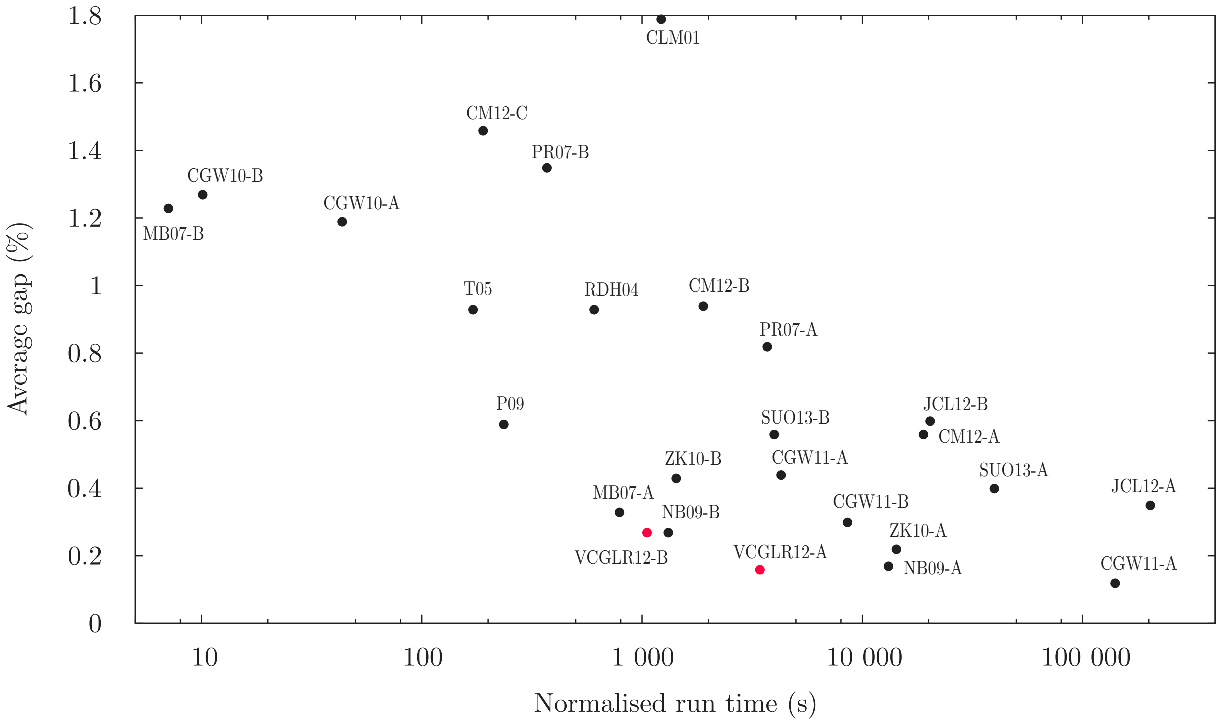


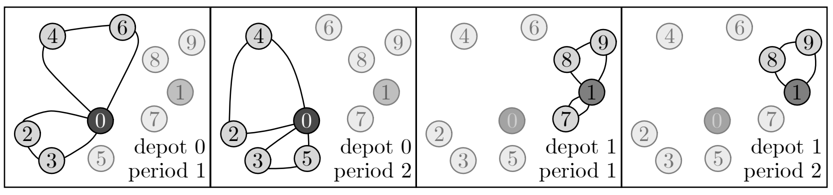
FIGURE 3.10: *The normalised run time and average gap to the best-known solutions for the GWKC benchmark instances for each of the metaheuristics listed in Table 3.2. Metaheuristics for which more than one algorithmic configuration have been proposed are indicated by an "A", "B" or "C" after the identifier of the metaheuristic, with "A" denoting the most powerful configuration [137].*

3.5.1 Solution representation

Each individual P in the population of candidate solutions maintained by the HGSADC algorithm is presented as a set of three chromosomes. They are the *pattern chromosome*, the *depot chromosome*, and the *giant tour chromosome*, as illustrated in Figure 3.11. The giant tour chromosome is the most important, since the other two chromosomes can be deduced from this chromosome. The pattern chromosomes registers, for each customer i, its so-called *pattern* $\pi_i(P)$, which is a set containing the periods of the planning horizon during which customer i is visited. The depot chromosome registers the depot assignment $\delta_i(P)$ of customer i (each customer may only be assigned to a single depot). The final chromosome, the giant tour chromosome, contains, for each depot o and period ℓ combination, a sequence of customers $V_{o\ell}(P)$ which is the concatenation of all routes from depot o during period ℓ in an arbitrary manner, with the removal of visits to the depot. In Figure 3.11, for example, the two routes $0, 4, 6, 0$ and $0, 3, 2, 0$ are visible, departing from depot 0 during period 1, and these routes are presented as a giant tour chromosome $V_{01}(P) = 4, 6, 3, 2$ which is the concatenation of the two routes in an arbitrary order, excluding visits to the depot. The giant tour chromosome allows for simple and efficient crossovers between individuals.

In order to extract routes from a giant tour chromosome, a so-called *split* algorithm is invoked to find optimal route delimiters within the giant tour (such that the load associated with each route is less than double the capacity of the delivery vehicle). Prins [110] reported the first successful utilisation of a giant tour representation within a genetic algorithm by proposing a split algorithm for extracting optimal routes from giant tours. His split algorithm was based on Bellman's shortest path finding algorithm which can be adapted to the current VRP setting by incorporating penalisation costs and a limited fleet size. The split algorithm of Prins [110] can be

MDVRP solution:



An individual in the genetic algorithm:

(P)

(a) Pattern chromosome

Cust i	2	3	4	5	6	7	8	9
Pat $\pi_i(P)$	{1,2}	{1,2}	{1,2}	{2}	{1}	{1}	{1,2}	{1,2}

(b) Depot chromosome

Cust i	2	3	4	5	6	7	8	9
Dep $\delta_i(P)$	0	0	0	0	0	1	1	1

(c) Giant tour chromosome

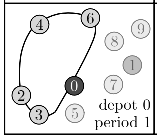
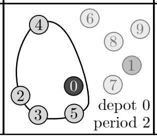
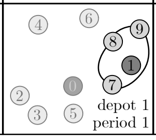
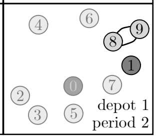
$V_{01}(P)$	$V_{02}(P)$	$V_{11}(P)$	$V_{12}(P)$
4 6 3 2	3 2 4 5	9 7 8	8 9
			

FIGURE 3.11: *Three chromosomes representing a candidate solution to an MDPVRP instance produced by the HGSADC algorithm [140].*

implemented in $O(n^2)$ time for unlimited fleets, where n denotes the number of customers, but an updated version of the algorithm, proposed by Vidal [138], runs in $O(n)$ time for unlimited fleets and in $O(nm)$ time for limited fleets containing m delivery vehicles. The Linear split algorithm for a limited fleet is described in pseudo-form in Algorithm 3.3.

Algorithm 3.3: Linear split

```
1  $p[0] \leftarrow 0$
2  $\Lambda \leftarrow (0)$
3  for $t = 1$ to $n$ do
4      $p[t] \leftarrow p[front] + f(front, t)$
5      $pred[t] \leftarrow front$
6      if $t < n$ then
7          if not $dominates(back, t)$ then
8              while $|\Lambda| > 0$ and $dominates(t, back)$ do
9                  $popBack()$
10             $pushBack()$
11         while $Q[t+1] > Q + Q[front]$ do
12             $popFront()$
```

Let $D[i]$ denote the cumulative distance and let $Q[i]$ denote the cumulative load associated with customers 1 to $i \in \{1, \dots, n\}$, computed as

$$D[i] = \sum_{k=1}^{i-1} d_{k,k+1}$$

and

$$Q[i] = \sum_{k=1}^{i} q_k,$$

respectively, where $d_{i,i+1}$ denotes the cost associated with travelling from vertex i to the next vertex in the giant tour, and $q[i]$ denotes the demand volume associated with vertex i. Furthermore, for $i < j$, let $c(i, j)$ denote the cost associated with departing from the depot, visiting customers $i + 1, \ldots, j$, and returning to the depot, computed as

$$c(i, j) = d_{0,i+1} + D[j] - D[i + 1] + d_{j,0},$$

where the arc (i, j) is only present if the demand volume associated with the route visiting customers $i, \ldots, j$ does not exceed the capacity $\mathcal{Q}$ of a delivery vehicle (*i.e.* if $Q[j] - Q[i] \leq \mathcal{Q}$). Let $\Lambda = (\lambda_1, \ldots, \lambda_{|\Lambda|})$ denote a double-ended queue to which the following operations may be applied in $O(n)$ time:

front	–	access the oldest element in the queue,
*front*2	–	access the second oldest element in the queue,
back	–	access the most recent element in the queue,
push_back	–	add an element to the queue,
pop_front	–	remove the oldest element from the queue,
pop_back	–	remove the most recent element from the queue.

A function $f(i, x)$ denotes the cost incurred when extending the route from a predecessor node i to a node $x \in \{i+1, \ldots, n\}$. This function returns an infinite value if the arc $(i, j) \notin \mathcal{A}$ because the cumulative volume of demand delivered to customers $\{i, \ldots, x\}$ exceeds the capacity of a delivery vehicle. The function may be defined as

$$f(i, x) = \begin{cases} p[i] + c(i, x), & Q[x] - Q[i] \leq \mathcal{Q}, \\ \infty, & \text{otherwise.} \end{cases}$$

Furthermore, a boolean function $dominates(back, t)$ returns $True$ if the node i dominates node j as a predecessor, or the value $False$ otherwise. The function is defined as

$$dominates(i, j) \equiv \begin{cases} p[i] + d_{0,i+1} - D[i+1] \leq p[j] + d_{0,j+1} - D[j+1] \text{ and } Q[i] = Q[j] & \text{if } i \leq j, \\ p[i] + d_{0,i+1} - D[i+1] \leq p[j] + d_{0,j+1} - D[j+1] & \text{if } i > j. \end{cases}$$

Algorithm 3.3 does not iterate over all arcs to compute minimum-cost paths, but rather maintains a set of non-dominated predecessors in the queue Λ. For each node $t \in \{1, \ldots, n\}$, this structure facilitates the identification of a best predecessor for t, stored in $pred[t]$, along with the cost of a shortest path from vertex 0 to vertex t, stored in $p[t]$. The final routes are then determined by working backwards through the predecessor matrix $pred[i]$.

3.5.2 Solution evaluation

In the HGSADC algorithm, the population consists of both feasible and infeasible individuals, stored in two separate subpopulations. A route r which forms part of a solution P is characterised by its load $q(r)$, its distance $c(r)$, its time-warp $tw(r)$, and its duration $\tau(r)$. Time-warp is the opposite of waiting time, where a driver arrives at a customer after the end of its specified time-window, as illustrated in Figure 3.12. Drivers are allowed to wait at a customer until the start of its time-window, but time-warp renders a route infeasible. When a driver arrives late at a customer, time-warp is penalised during evaluation of the route, and the driver is assumed to leave the customer at the end of its time-window. Routes are, in fact, allowed to be infeasible in terms of their load, duration, and time-warp, by penalising these infeasibilities with appropriate weights when calculating the costs associated with these routes. The penalised cost of a route r is calculated as

$$\phi(r) = c(r) + \omega^D \max\{0, \tau(r) - D\} + \omega^Q \max\{0, q(r) - Q\} + \omega^{TW} \times tw(r),$$

where D denotes the maximum duration of a route, Q denotes the maximum capacity of a delivery vehicle, and ω^D, ω^Q, and ω^{TW} denote the weights with which infeasible loads, durations, and time-warps are penalised in routes, respectively. The penalised cost $\phi(P)$ of a solution is then calculated as the sum of the penalised costs of all its routes.

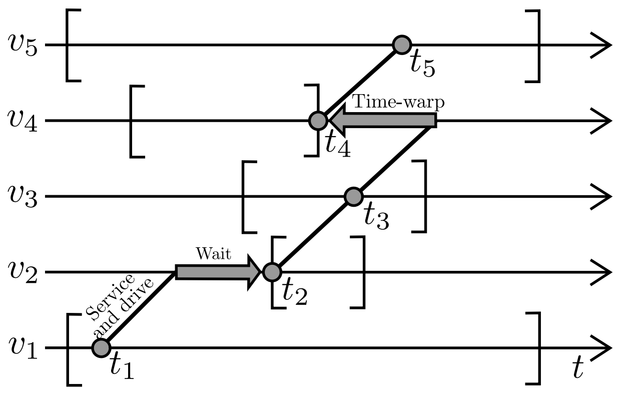


FIGURE 3.12: *A graphical illustration of waiting time and time-warp for the time-windows (denoted by brackets) of five customers* $v_1, \ldots, v_5$.

Any individual P in the population is characterised by its penalised cost $\phi(P)$ and its diversity contribution $\Delta(P)$. The diversity contribution of an individual is defined as the average Hamming distance δ from the individual to its closest neighbours N_{close} in the subpopulation. That is,

$$\Delta(P) = \frac{1}{N_{close}} \sum_{P_2 \in N_{close}} \delta(P, P_2).$$

Any individual P in the population is also evaluated based on its *biased fitness*

$$BF(P) = fit(P) + \left(1 - \frac{N_{elite}}{N_{indiv}}\right) dc(P),$$

where $fit(P)$ denotes the rank of an individual P in the subpopulation in terms of its penalised cost cost $\phi(P)$, and where $dc(P)$ denotes the rank of the individual in the subpopulation with respect to its diversity contribution $\Delta(P)$. This ensures that high-quality individuals are maintained in the population, but also that the population is diverse, and therefore sufficiently explores the solution space.

3.5.3 Pseudo-code description

The HGSADC algorithm is described in pseudo-code form in Algorithm 3.4. First, a population is initialised by creating 4μ individuals. An individual is created by randomly assigning each customer to a pattern and randomly inserting the customer into routes according to its selected pattern. Each initialised individual undergoes *eduction* and then *repair*, after which it is inserted into the appropriate subpopulation. The *education* and *repair* phases are essential for fast progression to high-quality solutions, but incur 90–95% of the computational effort of the algorithm.

The *education* phase is performed by invoking two local search procedures, namely *route improvement* (RI) and *pattern improvement* (PI), which are called in the order RI, PI, and then RI again. PI is performed by evaluating, for each customer in random order, the best combination of re-insertions within periods. Any improving re-insertion is immediately performed until all possible insertions have been evaluated. RI is performed by exploring a neighbourhood based on relocations and exchanges of customer visits, for each period. The following neighbourhoods are evaluated:

- N_1 (Swap and relocate): Swap two disjoint visitation sequences containing between zero and two visits. Combine this with the reversal of one or both sequences.
- N_2 (2-opt*): Swap two visitation sequences involving the extremities of two distinct routes.
- N_3 (2-opt): Reverse a visitation sequence.

Neighbourhoods N_1 and N_2 may involve one empty sequence (*i.e.* one of the two sequences may contain no customer visits). Each neighbourhood is explored in random order and the best improving move is implemented as soon as 5% of the neighbourhood has been explored since having accepted the last improving move. The neighbourhoods are pruned based on customer correlation measures with a view to improve the computational efficiency of the education phase. The neighbourhood of a customer v_i is defined as the set $v_j \in \Gamma(v_i)$ consisting of the $|\Gamma|$ closest customers v_j determined according to the correlation measure

$$\gamma(v_i, v_j) = c_{ij} + \gamma^{WT} \max\{e_j - \tau_i - t_{ij} - \ell_i, 0\} + \gamma^{TW} \max\{e_i + \tau_i + t_{ij} - \ell_j, 0\},$$

where c_{ij} is the cost associated with travelling from customer i to customer j, e_i is the earliest service start time at customer i, ℓ_i is the latest service start time at customer i, τ_i is the service duration at customer i, t_{ij} is the travel time from customer i to customer j, and γ^{WT} and γ^{TW} are coefficients that form part of the parameter set of the algorithm. The set of correlated customers $\Gamma(v_i)$ can be viewed as the best options for a direct visit from customer v_i, therefore restricting the neighbourhood search to a subset of "promising arcs," and thus lowering computational effort required during the *education* phase.

The algorithm iterates for a maximum time limit T_{max} or over at most It_{NI} iterations since last having improved the best feasible solution. During each iteration, parents are selected according to a binary tournament selection scheme from the union of both the feasible and infeasible populations. An offspring is then created by invoking the *periodic crossover with insertions* (PIX) operator [139] when solving a PVRPTW instance, or the Ordered crossover operator [36] when solving a VRPTW instance. Both these crossover operators are known to facilitate important structural changes, yet also allow for small solution refinement [140]. The PIX crossover is dedicated specifically to periodic routing problems and is designed to combine high-quality sequences of visits, while allowing for pattern, depot, and route recombinations. The PIX crossover is described in pseudo-code form in Algorithm 3.5. First, an inheritance rule is defined. Let Λ_1, Λ_2, and Λ_{mix} denote the sets of (depot, period) (o, ℓ) couples for which the offspring solution C inherits material from the first parent P_1, the second parent P_2, or both parents, respectively. Data are inherited from P_1. For $(o, \ell) \in \Lambda_1$, the sequence of customer visits in $V_{(o,\ell)}(P_1)$ is copied to $V_{(o,\ell)}(C)$. For $(o, \ell) \in \Lambda_{mix}$, on the other hand, two chromosome cutting points α_{kl} and β_{kl} determine the substring of customer visits in $V_{(o,\ell)}(P_1)$ to be copied to $V_{(o,\ell)}(C)$. Thereafter, data are inherited from P_2. For $(o, \ell) \in \Lambda_2 \cup \Lambda_{mix}$ in random order, visitation of customer i in $V_{(o,\ell)}(P_2)$ is copied to the end of $V_{(o,\ell)}(C)$ if customer i is already assigned to depot o or if no depot assignment have been made yet, and the addition of period ℓ to the current visitation pattern $\pi_i(C)$ of customer i in solution C is feasible (included in the list of

Algorithm 3.4: HGSADC [140]

```
1  Initialise population
2  while number of iterations without imporvement $\leq It_{NI}$, and time $\leq T_{max}$ do
3      Select parent solutions $P_1$ and $P_2$
4      Create offspring $C$ from $P_1$ and $P_2$ (crossover)
5      Educate $C$ (local search procedure)
6      if $C$ infeasible then
7          Insert $C$ into infeasible subpopulation,
8          Repair with probability $P_{rep}$
9      if $C$ feasible then
10         Insert $C$ into feasible subpopulation
11     if maximum subpopulation size reached then
12         Select survivors
13     if best solution not improved for $It_{div}$ iterations then
14         Diversify population
15     Adjust penalty parameters for infeasibility
16     if number of iterations $= k \times It_{dec}$ where $k \in N$ then
17         Decompose the master problem
18         Use HGSADC on each subproblem
19         Reconstitute three solutions, and insert them in the population
20 Return best feasible solution
```

allowable visitation patterns, denoted by L_i). Finally, customer services are completed so as to ensure that each customer is assigned a feasible visiting pattern. The split algorithm is invoked to extract routes from the current giant tour chromosome of C, and the visitation frequency of each customer is determined. While customers with unsatisfied visitation frequencies exist, these customers are selected in random order and an insertion corresponding to the minimum increase in penalised cost is performed. When the crossover procedure has been completed, the offspring undergoes *education*, after which it is placed in the appropriate subpopulation (feasible or infeasible). If it is infeasible, it is repaired with probability P_{rep} and then placed in the feasible subpopulation if feasible.

Both subpopulations are managed to contain between μ and $\mu + \lambda$ individuals (except when the population is initialised). If any subpopulation size reaches $\mu + \lambda$ individuals, *survivor selection* takes place by removing λ individuals from the subpopulation in order once again to contain the minimum number of μ individuals in the subpopulation. A total of λ individuals are removed by first removing duplicate individuals after which the worst individuals are removed in terms of biased fitness — ensuring that good solutions are maintained in the subpopulation and that diversity is also maintained.

Another phase that contributes to the diversity of the population, is *diversification.* This phase occurs whenever It_{div} iterations have passed since last having improved the best feasible solution. The best $\mu/3$ individuals are retained in each subpopulation and 4μ individuals are introduced into the population in the same manner as when the population was initialised.

The penalty parameters ω^D, ω^Q, and ω^{TW} are adjusted every one hundred iterations. The parameter ζ^{REF} represents the target proportion of feasible individuals. If the feasible portion of the last one hundred newly generated individuals is between $\zeta^{REF} - 5\%$ and $\zeta^{REF} + 5\%$,

Algorithm 3.5: The PIX operator [139]

```
1  STEP 0: INHERITANCE RULE
2  Pick two random numbers between zero and the total number of (depot,period) $(o, \ell)$
   couples according to a uniform distribution. Let $n_1$ and $n_2$ be the smallest and the
   largest of these numbers, respectively
3  Randomly select $n_1$ (depot, period) couples to form the set $\Lambda_1$
4  Randomly select $n_2 - n_1$ remaining couples to form the set $\Lambda_2$
5  The remaining $td - n_2$ couples make up the set $\Lambda_{mix}$
6  STEP 1: INHERIT DATA FROM $P_1$
7  for each (depot,period) (o, ℓ) belonging to the set $\Lambda_1$ or $\Lambda_{mix}$ do
8      $\Lambda_1$: Copy the sequence of customer visits from $V_{o,\ell}(P_1)$ to $V_{o,\ell}(C)$
9      $\Lambda_{mix}$: Randomly select two chromosome-cutting points $\alpha_{kl}$ and $\beta_{kl}$ according to a
       uniform distribution and copy the $\alpha_{kl}$ to $\beta_{kl}$ substring of $V_{o,\ell}(P_1)$ to $V_{o,\ell}(C)$
10 STEP 2: INHERIT DATA FROM $P_2$
11 for each (depot,period) $(o, \ell) \in \Lambda_2 \cup \Lambda_{mix}$ selected in random order do
12     Consider each customer visit $i$ in$V_{o,\ell}(P_2)$ and copy it to the end of $V_{o,\ell}(C)$ when
13     (1) The depot choice $\delta_i(C)$ is equal to $o$ or undefined (no visit to $i$ has been copied to
       $C$ yet), and
14     (2) At least one visit pattern of customer $i$ contains the set $\pi_i(C) \cup \ell$ of visit periods.
15 STEP 3: COMPLETE CUSTOMER SERVICES
16 Invoke the split algorithm to extract routes for each (depot, period) pair
17 if the service frequency requirements are satisfied for all customers then
18     Stop
19 else
20     while customers with unsatisfied frequency requirements exist do
21         Randomly select a customer $i$ for which service frequency requirements are not
           satisfied
22         Let $\mathcal{F}$ be the set of admissible (depot,period) $(o, \ell)$ couples with respect to its
           pattern list $L_i$ and the visits already included in $C$. Let $\psi(i, o, \ell)$ be the minimum
           penalised cost for the insertion of customer $i$ into a route from depot $o$ during
           period $\ell$. Insert $i$ into $(o^*, \ell^*) = \arg\min_{(o,\ell)\in\mathcal{F}} \psi(i, o, \ell)$
```

the penalty parameters remain unchanged. If, however, the portion of feasible newly generated individuals are below or above this range, the penalty parameters are increased or decreased, respectively, in order to achieve the target proportion of feasible individuals.

The *decompistion* phase occurs every It_{dec} iterations and enables the algorithm to solve large problem instances effectively. During this phase, the problem is decomposed into smaller subproblem instances and the HGSADC algorithm is applied to each of them for $It_{dec}/2$ iterations. An elite individual is uniformly chosen from the 25% best feasible individuals and used to decompose the problem. In the VRPTW case, subproblem instances are created by sweeping the routes of the elite solution circularly around the depot by polar angle and adding routes to the subproblem instance until a certain number of customers have been included or all the routes have been swept. In the PVRPTW case, the period assignments of the elite individual can be fixed to decompose the problem instance naturally and treat each period as a subproblem instance. Once the HGSADC algorithm has been applied to each subproblem instance, an elite solution is attained by joining the best feasible solution to each subproblem instance and inserting it into the population. Two additional elite solutions are created in the same manner and also inserted into the population.

3.5.4 Parameter calibration

A *metacalibration* approach [95] was adopted in order to identify good parameter values for the HGSADC algorithm. This approach was adopted due to its ability to perform particularly well for genetic algorithm calibration [123]. Metacalibration involves invoking metaheuristics to solve the problem of parameter optimisation [139]. Parameter tuning for the HGSADC algorithm was performed by means of the *evolutionary strategy with covariance matrix adaptation* (CMA-ES) proposed by Hansen and Ostermeier [67]. A set of training instances for various VRP classes was employed during the calibration process and an appropriate set of parameters was determined for each VRP class, as indicated in Table 3.3. It was found that this set of parameters appears to be independent of the VRP class, with the exception of the generation size λ. For VRP classes with time-windows, a separate calibration procedure was performed in order to obtain good parameter values for γ^{TW} and γ^{WT}, which balances the role of geometrical and temporal aspects during neighbourhood pruning in VRP classes with time-windows. It was found that the performance of the HGSADC algorithm appears to increase with parameter values close to $(\gamma^{TW}, \gamma^{WT}) = (1, 0.2)$. Furthermore, in order to balance the run time of the algorithm with those incurred by other authors, the parameter values $It_{NI} = 5\,000$ and $It_{dec} = 2\,000$ were recommended.

3.6 Approximate multi-objective optimisation

The problem of solving multi-objective optimisation problems approximately has resulted in a large number of methods being proposed for the task. Crucially, a method is required for assigning fitness values to solutions in such a manner that all objectives are considered during selection or evaluation procedures of approximate solution approaches. Many classical methods are available according to which a vector of objective function values can be scalarised into a single fitness value. In the *objective weighting* method, M objectives $f_1, \ldots, f_M$ (which are all to be minimised) are combined into one overall value

$$Z = \sum_{i=1}^{M} w_i f_i(\boldsymbol{x}), \qquad \boldsymbol{x} \in \boldsymbol{X},$$

TABLE 3.3: *The results of the parameter calibration performed for the HGSADC algorithm for different VRP classes [139].*

	Parameter	Range	PVRP	MDVRP	MDPVRP	Final parameters
μ	Population size	[5,200]	18	24	30	**25**
λ	Number of offspring in a generation	[1,200]	33	87	146	**40\70\100**
el	Proportion of elite individuals, such that $nbElit = el \times \mu$	[0,1]	0.38	45	0.36	**0.4**
nc	Proportion of close individuals considered for distance evaluation, such that $N_{close} = nc \times \mu$	[0,0.25]	0.24	0.18	0.15	**0.2**
P_{rep}	Repair rate	[0,1]	0.57	0.61	0.33	**0.5**
h	Granularity threshold in RI	[0,1]	0.53	0.36	0.35	**0.4**
ξ^{REF}	Reference proportion of feasible individuals	[0,1]	0.1	0.3	0.2	**0.2**

where $\boldsymbol{X}$ represents the feasible region, and w_i denotes the weight associated with objective function f_i, such that $\sum_{i=1}^{M} w_i = 1$. The solution returned when employing this method, of course, depends on the weight vector $\boldsymbol{w} = [w_1, \ldots, w_M]$. An advantage of using this method is that subjective preference of one objective over another can easily be controlled and that a Pareto-optimal solution is associated with a minimal value of Z in the case of a convex optimisation problem. A major disadvantage of the objective weighting method, however, is that Pareto-optimal solutions in non-convex regions of the decision space are masked (*i.e.* are not identifiable by large values of Z).

According to the *distance functions* method, scalarisation is performed using a so-called demand-level vector $\bar{\boldsymbol{y}}$, representing the individual optima of each objective function. Multiple objectives may be combined into a single value

$$Z = \left[\sum_{i=1}^{M} |f_i(\boldsymbol{x}) - \bar{y}_i|^r \right]^{1/r}, \qquad r \geq 1, \quad \boldsymbol{x} \in \boldsymbol{X},$$

where r is often assigned the value 2. An inappropriate demand-level vector $\bar{\boldsymbol{y}}$ will, however, lead to a non-Pareto-optimal solution being returned [124].

According to the *min-max* method, the relative deviations of the objective function values from their demand-level values are minimised. For a problem in which all objectives are to be minimised, the corresponding min-max problem is

$$\text{minimise } \mathcal{F}(\boldsymbol{x}) = \max[Z_i(\boldsymbol{x})], \qquad i = 1, \ldots, M \quad \boldsymbol{x} \in \boldsymbol{X},$$

where $Z_i(\boldsymbol{x})$ is calculated for non-negative target optimal values $\bar{y}_i$ of each objective function value $f_i(\boldsymbol{x})$ as

$$Z_i(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x}) - \bar{y}_i}{\bar{y}_i}, \qquad i = 1, \ldots, M.$$

This method often returns good solutions when all objectives are of equal importance. Classical methods are, however, very sensitive to the weights or demand-levels used to scalarise vectors and therefore require prior knowledge of the problem being solved [124]. Furthermore, a single solution is returned in such approaches, whereas it may be preferred to return a set of non-dominated solutions in order to have many possible solutions to select from, based on the multiple objectives of the problem.

Srinivas and Deb [124] proposed a *non-dominated sorting procedure* in 1995 for assigning fitness values to solutions based on their degree of dominance relative to each other. Instead of only differentiating between dominated and non-dominated solutions, each solution is assigned a rank based on the non-dominated front in which it resides. This rank, denoted by F, is assigned a value by iteratively identifying non-dominated fronts, removing the solutions residing in these fronts, and repeating the process until no unassigned solutions remain. For example, the first set of non-dominated solutions may be identified and assigned a rank of $F = 1$. The subset of non-dominated solutions with rank $F = 1$ are then removed, and the new subset of non-dominated solutions among the remaining solutions are assigned a rank of $F = 2$. The process is continued until all solutions have been assigned a rank value. This concept is illustrated graphically in Figure 3.13 by indicating the non-dominated front (or rank) in which each solution resides. Solutions 1, 3, and 7 form the first set of non-dominated solutions and are therefore assigned a rank value of $F = 1$. If these solutions are removed from the set of solutions, Solutions 2 and 4 form the second set of non-dominated solutions and are therefore assigned a rank value of $F = 2$. Finally, if all of the aforementioned solutions are removed from the set of solutions, Solutions 5 and 6 form the third set of non-dominated solutions and are therefore assigned a rank value of $F = 3$. Since each solution has been assigned a rank value, the non-dominated sorting procedure terminates. Furthermore, in order to differentiate between solutions residing within the same non-dominated front, a dummy fitness value is assigned to each solution based on the number of solutions surrounding it in close proximity. The dummy fitness values of solutions within a non-dominated front, however, remain larger (in the case of minimisation) than the dummy fitness values in the previous front, thus ensuring that preference is afforded to solutions in better ranked non-dominated fronts.

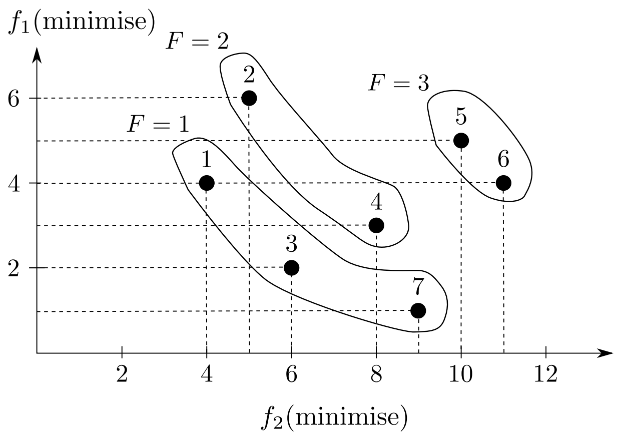


FIGURE 3.13: *The rank F assigned to individuals residing in each non-dominated front during the non-dominated sorting procedure of Srinivas and Deb [124].*

Deb *et al.* [38] applied this non-dominated sorting procedure, with the addition of a *crowding distance density* (CDD) measure for each solution in each non-dominated front, in order to further differentiate between solutions within the same non-dominated front. The CDD associated with a solution is a measure of its proximity to other solutions in the same non-dominated front. A smaller CDD value indicates that a solution occurs within a dense area of its non-dominated front, whereas a larger CDD value indicates that a solution is more isolated. A pseudo-code description of the assignment of the CDD measure to solutions in a non-dominated front is provided in Algorithm 3.6.

Algorithm 3.6: Crowding distance density assignment [38]

```
Input  : A non-dominated set $\boldsymbol{C}$ of solutions to a multi-objective optimisation problem
         instance, containing the objective function values for each solution in a vector
         denoted by $\boldsymbol{z}$.
Output: The crowding distance $\boldsymbol{C}[i]_{dist}$ for each solution in $\boldsymbol{C}$.
1  $\ell \leftarrow |\boldsymbol{C}|$
2  for $i \in \boldsymbol{C}$ do
3      $\boldsymbol{C}[i]_{dist} \leftarrow 0$
4  for $m = 1, \dots, M$ do
5      $\boldsymbol{C} \leftarrow$ sort$(\boldsymbol{C}, m)$
6      $\boldsymbol{C}[1]_{dist}|m \leftarrow \infty$
7      $\boldsymbol{C}[\ell]_{dist}|m \leftarrow \infty$
8      for $i = 2, \dots, (\ell - 1)$ do
9          $\boldsymbol{C}[i]_{dist}|m \leftarrow (\boldsymbol{C}[i+1]|m - \boldsymbol{C}[i-1]|m)/(\boldsymbol{z}_m^{\max} - \boldsymbol{z}_m^{\min})$
10 for $i = 1, \dots, \ell$ do
11     $\boldsymbol{C}[i]_{dist} = \sum_{m=1}^{M} \boldsymbol{C}[i]_{dist}|m$
12 Return $\boldsymbol{C}[i]_{dist}$ for each solution $i \in \boldsymbol{C}$
```

The CDD value is assigned to each non-dominated front $\boldsymbol{C}$ by successively considering as input the solutions in the non-dominated front and a vector $\boldsymbol{z}$ containing their corresponding objective function values for each of the M objectives. Let ℓ denote the number of solutions in the non-dominated front $\boldsymbol{C}$. First, the CDD of each solution $i \in \boldsymbol{C}$, denoted by $\boldsymbol{C}[i]_{dist}$, is set to a value zero. Thereafter, for each objective $m \in \{1, \dots, M\}$, the solutions are sorted in ascending order in terms of that objective by invoking the sort function. The m^{th} objective function value of the i^{th} solution in the sorted list is denoted by $\boldsymbol{C}[i]|m$. Furthermore, the CDD of the i^{th} solution for the m^{th} objective is denoted by $\boldsymbol{C}[i]_{dist}|m$. For each objective m, the solutions corresponding to the minimum and maximum objective function values, represented by the first and the last elements in the sorted list $\boldsymbol{C}$, are assigned an infinite CDD measure (*i.e.* $\boldsymbol{C}[1]_{dist}|m = \infty$ and $\boldsymbol{C}[\ell]_{dist}|m = \infty$). For the intermediate solutions $i \in \{2, \dots, \ell - 1\}$ in the ordered list $\boldsymbol{C}$, each solution i is assigned a CDD value by calculating the normalised distance between the solutions below Solution $i - 1$ and above Solution $i + 1$ it in the ordered list $\boldsymbol{C}$. This is calculated as $(\boldsymbol{C}[i+1]|m - \boldsymbol{C}[i-1]|m)/(\boldsymbol{z}_m^{\max} - \boldsymbol{z}_m^{\min})$, where $\boldsymbol{z}_m^{\max}$ and $\boldsymbol{z}_m^{\min}$ denote the maximum and minimum values for the m^{th} objective in the non-dominated front $\boldsymbol{C}$, respectively. Finally, the CDD value associated with each solution $i \in \boldsymbol{C}$ is calculated as the sum of all CDD values calculated for each objective for the corresponding solution. A graphical illustration of the CDD measure for a bi-objective optimisation problem is shown in Figure 3.14, where solid vertices represent solutions that form part of the set of non-dominated solutions under consideration.

Since a GA is a population-based metaheuristic operating on many solutions at once, it is often employed for solving multi-objective optimisation problems in order to capture a number of approximately Pareto-optimal solutions [124]. In order to direct the search for solutions towards the approximate Pareto optimal solutions already uncovered, measures such as the non-dominating sorting procedure have been employed in conjunction with the CDD measure in a variety of GAs [38]. During selection procedures, such as parent selection or survivor selection, the rank of the non-dominated front in which solutions reside may be used as selection criterion.

In the case of two or more solutions residing in the same non-dominated front, the solutions associated with the largest CDD value may be selected in order to direct the search towards sparser areas of the approximate Pareto front.

In order to compare the relative performances of different multi-objective optimisation methods, a procedure is required to compare the sets of non-dominated solutions returned by these methods for multi-objective optimisation problem instances. In the context of single-objective optimisation problems, the solutions returned by two different methods may easily be compared by simply comparing the objective function values associated with the solutions. In the context of multi-objective optimisation problems, however, a set of many solutions may be returned for a single problem instance, therefore not allowing for easy comparison of the sets of solutions returned by different optimisation methods. Sets of solutions may, of course, be compared by visually inspecting them in objective space, although this approach does not quantify the quality of solutions and becomes difficult to employ when more than three objectives are to be optimised [87].

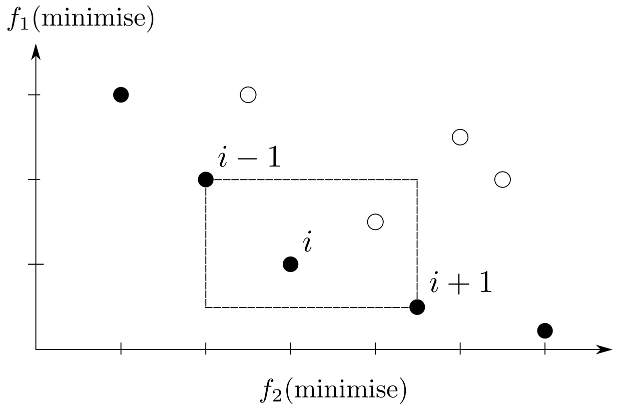


FIGURE 3.14: *A graphical illustration of the CDD measure calculation for solution i, indicating the difference between the objective function values of the lower and higher ranked solutions within the non-dominated front of solution i. Solid vertices represent solutions residing within the non-dominated front under consideration [38].*

A popular measure employed to quantify the quality of a set of solutions is the *hypervolume* (HV) quality indicator proposed by Zitzler and Thiele [147] in 1998. This measure is often employed, due to its ability to compare two sets of solutions, without requiring the Pareto set. The HV of a set of non-dominated solutions is the volume enclosed in objective space between the set of solutions and a reference point, and is illustrated graphically in Figure 3.15. When comparing sets of solutions, the set corresponding to the largest HV measure is considered to be better [87].

When computing the HV measure, the choice of reference point is an important decision, because different reference points may lead to results being inconsistent, yet no consensus has been reached in the literature on how to choose a reference point for a given problem instance [80]. It is common practice to take the *nadir point* or 1.1 times the nadir point as reference point. The nadir point corresponds to the worst objective function values for solutions in the Pareto set [79]. The nadir point therefore corresponds to lower and upper bounds for the Pareto set, in the case of an objective being maximised or minimised, respectively. If, however, the nadir point is estimated incorrectly and a solution outside the objective space bounds of the nadir point is present, a disproportional HV measure will be returned for the corresponding set of solutions [79].

Furthermore, the HV measure is sensitive to the range of a problem instance's objectives. It is therefore recommended to normalise the objectives of the problem to prevent objectives with

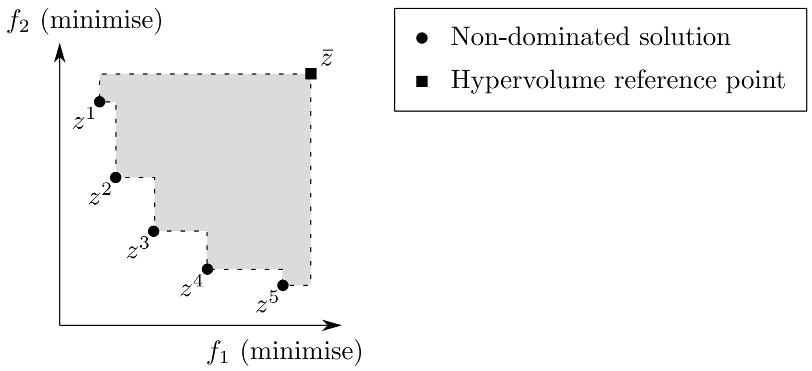


FIGURE 3.15: *A graphical illustration of the HV measure for a bi-objective problem instance in which both objectives are to be minimised.*

larger ranges from having a greater effect on the HV measure [87]. In order for each objective to have an equal effect on the HV measure of a set of solutions, objectives are converted to lie within the same range, such as the unit interval $[0, 1]$, for instance. It is common practice to transform the objectives of solutions as $a'_m = (a_m - \min_m)/(\max_m - \min_m)$, where a'_m and a_m denote the transformed and original values of objective function m of solution a, respectively, and $\min_m$ and $\max_m$ denote the minimum and maximum values of objective function m in the Pareto set, respectively. If the Pareto set is not known, then $\min_m$ and $\max_m$ denote the minimum and maximum values for objective function m with respect to the combined set of non-dominated solutions of the collection of all solution sets being considered [87].

3.7 Chapter summary

This chapter was devoted to a discussion on algorithmic solution techniques for solving VRPs. Two main classes of solution methodologies were discussed, namely exact solutions methodologies and approximate solution methodologies. Three exact solution approaches applicable to the methodologies employed later in this thesis were discussed in §3.1. Thereafter, a classification of heuristic solution methods used to solve VRPs was discussed in §3.2, along with an example of a heuristic in each of the discussed classes. Similarly, a classification of metaheuristic solutions was discussed in §3.3. A comparison of various metaheuristics in the context of VRPs was discussed in §3.4. The working of the HGSADC algorithm, a state-of-the-art solution methodology employed later in this thesis, was described in §3.5, and this was followed in §3.6 by a discussion on approaches that have been adopted in the literature for solving multi-objective optimisation problems approximately.

Part II

Mathematical Models

CHAPTER 4

Model for the strategic phase

Contents

In this chapter, a mathematical model is proposed for a portion of the FVRP. This model assumes the form of a mixed binary programming problem. The model is applicable during the strategic phase of the FVRP for which decision support is proffered by the framework of the previous chapter. First, the model is derived in §4.1, after which an exact solution approach for the model, involving the branch-and-cut method, is described in §4.2. The model is implemented in the CPLEX optimisation environment in order to invoke this solution methodology. The methodology followed to verify the model implementation is discussed in §4.3. The time complexity of the model is next studied empirically in §4.4, and thus is followed by a discussion on the approximate solution methodology adopted for realistically sized instances of model in §4.5. The chapter closes in §4.7 with a summary of its contents.

4.1 Model derivation

This section is devoted to a derivation of the aforementioned model for the strategic phase of the FVRP. First, the required input data are elucidated in §4.1.1 in terms of the model parameters employed to configure an instance of the model. The decision variables of the model are next declared in §4.1.2, after which the constraints of the model are derived and explained in §4.1.3. The objective function of the model is then derived and motivated in §4.1.4, upon which an additional constraint set, aimed at improving the performance of the exact solution methodology, is derived in §4.1.5.

4.1.1 Model parameters

In the model for the strategic phase of the FVRP proposed in this chapter, a set of master routes is computed as the routing solution for a fleet of available delivery vehicles. In the remainder of this section, the routes travelled by these delivery vehicles represent the aforementioned master routes. Let $\mathcal{V} = \{0, 1, \ldots, n, n+1\}$ index the set of FVRP vertices, with 0 and $n+1$ representing virtual copies of the depot — the former representing the depot at vehicle departure and the latter representing the depot at vehicle return. The set $\mathcal{V}' = \mathcal{V} \setminus \{0, n+1\}$ furthermore represents the set of customers to be serviced. Let $\mathcal{A}$ denote the set of directed road links, called *arcs*, that delivery vehicles are allowed to traverse. The directed travel graph on which the FVRP is defined, is denoted by $G(\mathcal{V}, \mathcal{A})$, which has $\mathcal{V}$ as vertex set and $\mathcal{A}$ as arc set. Let $\delta^+(i)$ denote the set of vertices to which there are arcs from vertex $i \in \mathcal{V}$ in the travel graph $G(\mathcal{V}, \mathcal{A})$ and let $\delta^-(i)$ denote the set of vertices from which there are arcs to vertex $i \in \mathcal{V}$.

Let $\mathcal{L} = \{1, 2, \ldots, |\mathcal{L}|\}$ index the set of vehicle types to which each delivery vehicle may belong. Moreover, let $\mathcal{K} = \{1, 2, \ldots, |\mathcal{K}|\}$ index the set of (heterogeneous) delivery vehicles available at the depot to service customers, ordered in such a way that delivery vehicles of the same type $\ell \in \mathcal{L}$ are indexed successively. Let K_ℓ denote the number of identical delivery vehicles of vehicle type $\ell \in \mathcal{L}$, so that $\sum_{\ell \in \mathcal{L}} K_\ell = |\mathcal{K}|$. Let Q_k denote the cargo volume capacity of delivery vehicle $k \in \mathcal{K}$ and let C_k denote the fixed cost associated with using delivery vehicle $k \in \mathcal{K}$. Also, let β_k denote a cost coefficient associated with lengthening the trip duration of vehicle $k \in \mathcal{K}$ by one minute. Moreover, define the binary parameters

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

in order to allow for potential size restrictions preventing certain vehicles from visiting certain customers. Since different vehicle types may have different travel times and travel costs associated with them, let c_{ijk} denote the cost of vehicle $k \in \mathcal{K}$ travelling from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$ and let t_{ijk} denote the expected time (in minutes) spent by vehicle $k \in \mathcal{K}$ travelling from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$.

In order to incorporate the multiple visits attribute into the model, it is necessary to specify the number of different vehicles that are required to visit customer $i \in \mathcal{V}'$, denoted by f_i. Moreover, in order to specify time-windows for each customer, let a_i denote the earliest possible vehicle service start time at customer $i \in \mathcal{V}'$ and let b_i denote the latest possible vehicle service start time at customer $i \in \mathcal{V}'$ (both measured in minutes after some fixed reference time). Additionally, let q_i denote the average demand volume of customer $i \in \mathcal{V}'$ per planning period. Furthermore, let s_i denote the average service time duration at customer $i \in \mathcal{V}'$ (measured in minutes), calculated as a function based on the average demand volume q_i per planning period associated with each customer, to which a fixed service setup time is added.

An *arc overlap* amongst the master routes is defined as the occurrence of a vehicle traversing an arc $(i, j) \in \mathcal{A}$ while another vehicle already traverses the same arc (in any direction). For instance, if no vehicle, or only one vehicle, traverses an arc $(i, j) \in \mathcal{A}$, then no overlap is associated with that arc amongst the master routes. If, however, two vehicles traverse the same arc $(i, j) \in \mathcal{A}$ (in any direction), a single overlap is associated with that arc amongst the master routes. Moreover, if three vehicles traverse the same arc $(i, j) \in \mathcal{A}$ (in any direction), two overlaps are associated with that arc amongst the master routes, and so forth. While such arc overlaps amongst the master routes are undesirable (because they represent a reduction in the variety of approach directions to customers available for vehicle routing during the operational phase of the FVRP), a certain number of overlaps is tolerable. Let α_1 denote the total number of arc overlaps amongst the master routes that are allowed without penalty, and let ρ be a cost coefficient associated with penalising the number of arc overlaps amongst the master routes over and above the maximally tolerated penalised number α_1.

4.1.2 Model variables

The binary decision variables

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

capture all vehicle flows and are stored in row i and column j of slice k in a three-dimensional $(n+2) \times (n+2) \times |\mathcal{K}|$ flow matrix $\boldsymbol{X}$. The binary auxiliary decision variables

$$y_k = \begin{cases} 1 & \text{if vehicle } k \in \mathcal{K} \text{ is utilised} \\ 0 & \text{otherwise} \end{cases}$$

are also defined and stored in a vector $\boldsymbol{Y}$containing $|\mathcal{K}|$ entries.

The service start time at node $i \in \mathcal{V}$, when serviced by vehicle $k \in \mathcal{K}$, is captured in a real-valued variable T_{ik} and stored in an $(n+2) \times |\mathcal{K}|$ matrix $\boldsymbol{T}$. The service start time of delivery vehicle k (if utilised) at vertex 0, is denoted by T_{0k} and represents the time at which the delivery vehicle departs from the depot. In addition, the service start time at vertex $n+1$ of the same delivery vehicle (if utilised) is denoted by $T_{n+1,k}$ and represents the time at which the delivery vehicle returns to the depot, after having departed from the depot and having served all customers assigned to it. The trip duration of vehicle $k \in \mathcal{K}$ (if utilised) is therefore $T_{n+1,k} - T_{0k}$.

The number of overlaps amongst the master routes involving an arc $(i, j) \in \mathcal{A}$ is captured in a real-valued variable r_{ij} and stored in an $(n+2) \times (n+2)$ matrix $\boldsymbol{R}$. Finally, the total number of penalised arc overlaps amongst the master routes is captured in a variable α_2. If there are, for example, $\sum_{(i,j)\in\mathcal{A}} r_{ij} = 5$ arc overlaps in total amongst all of the master routes and $\alpha_1 = 2$ arc overlaps are allowed without penalty, then the number of penalised arc overlaps is $\alpha_2 = \sum_{(i,j)\in\mathcal{A}} r_{ij} - \alpha_1 = 5 - 2 = 3$.

4.1.3 Model constraints

A number of constraints are imposed in the model to ensure the practical feasibility of solutions. In order to ensure that each customer $i \in \mathcal{V}'$ is visited exactly once by f_i distinct vehicles, the constraint set

$$\sum_{k\in\mathcal{K}} \sum_{j\in\delta^+(i)} x_{ijk} = f_i, \qquad i \in \mathcal{V}' \tag{4.1}$$

is imposed. Moreover, in order to enforce a source (depot) to sink (depot) path for each delivery vehicle utilised, the three constraint sets

$$\begin{aligned} \sum_{j\in\delta^+(0)} x_{0jk} &= y_k, \quad k\in\mathcal{K}, && (4.2)\\ \sum_{i\in\delta^-(j)} x_{ijk} - \sum_{i\in\delta^+(j)} x_{jik} &= 0, \quad j\in\mathcal{V}',\ k\in\mathcal{K}, && (4.3)\\ \sum_{i\in\delta^-(n+1)} x_{i,n+1,k} &= y_k, \quad k\in\mathcal{K} && (4.4)\end{aligned}$$

are imposed. Imposition of the constraint sets

$$\begin{aligned} T_{ik}+s_i+t_{ijk}-T_{jk} &\leq M_{ijk}(1-x_{ijk}), && k\in\mathcal{K},\ (i,j)\in\mathcal{A}, && (4.5)\\ a_i\sum_{j\in\delta_i^+} x_{ijk} \leq T_{ik} &\leq b_i\sum_{j\in\delta_i^+} x_{ijk}, && i\in\mathcal{V}\setminus\{n+1\},\ k\in\mathcal{K}, && (4.6)\\ a_{n+1}\sum_{i\in\delta_{n+1}^-} x_{i,n+1,k} \leq T_{n+1,k} &\leq b_{n+1}\sum_{j\in\delta_{n+1}^-} x_{j,n+1,k}, && k\in\mathcal{K} && (4.7)\end{aligned}$$

furthermore enforce adherence to customer delivery time-windows, where constraint set (4.5) also implicitly avoids subtour formation along vehicle routes. The parameter M_{ijk} in constraint set (4.5) represents a large constant, which can be set to $\max\{b_i+s_i+t_{ijk}-a_j,0\}$, with the maximum taken over all $i,j\in\mathcal{V}'$ and $k\in\mathcal{K}$. In order to ensure that each delivery vehicle utilised is able to service all customers along its assigned route without exceeding its capacity, the constraint set

$$\sum_{i\in\mathcal{V}'}\sum_{j\in\delta_i^+}\frac{q_i}{f_i}\times x_{ijk}\leq C_k y_k, \quad k\in\mathcal{K}, \tag{4.8}$$

is imposed. In order to ensure that a vehicle which is, in fact, able to visit any particular customer $i\in\mathcal{V}'$ is assigned to service that customer, the constraint set

$$\sum_{j\in\delta^+(i)} x_{ijk}\leq g_{ik}, \quad i\in\mathcal{V}',\ k\in\mathcal{K}, \tag{4.9}$$

is imposed, while the constraint set

$$\sum_{(i,j)\in\mathcal{A}} x_{ijk}\leq(n+1)y_k, \quad k\in\mathcal{K}, \tag{4.10}$$

ensures that a vehicle is considered utilised if it services at least one customer. Moreover, the constraint sets

$$\begin{aligned} \sum_{k\in\mathcal{K}}(x_{ijk}+x_{jik})-1 &\leq r_{ij}, \quad (i,j)\in\mathcal{A}, && (4.11)\\ r_{ij} &\geq 0, \quad (i,j)\in\mathcal{A}, && (4.12)\end{aligned}$$

are imposed in order to count the number of master route link overlaps that occur at each arc $(i,j)\in\mathcal{A}$, denoted by r_{ij}. Constraint sets (4.11) and (4.12) together ensure that the value of r_{ij} is $\max\{\sum_{k\in\mathcal{K}}(x_{ijk}+x_{jik})-1,0\}$. The additional constraint

$$\sum_{(i,j)\in\mathcal{A}}\frac{r_{ij}}{2}\leq\alpha_1+\alpha_2, \tag{4.13}$$

is imposed in order to count the total number of edge overlaps amongst the master routes. In constraint (4.13), the sum of the entries of the matrix $\boldsymbol{R}$ is divided by two, since the number of overlaps r_{ij} involving the arc $(i,j) \in \mathcal{A}$ is represented by both r_{ij} and r_{ji}, and is therefore counted twice. As a mere convention for a valid model formulation, the constraint

$$s_0 = 0 \tag{4.14}$$

is also imposed. The domain constraints

$$\begin{aligned} x_{ijk} &\in \{0,1\}, \quad && k \in \mathcal{K},\ (i,j) \in \mathcal{A}, && (4.15)\\ y_k &\in \{0,1\}, && k \in \mathcal{K}, && (4.16)\\ T_{ik} &\geq 0, && i \in \mathcal{V},\ k \in \mathcal{K}, && (4.17) \end{aligned}$$

are finally imposed to enforce the variable nature (binary and real-valued, respectively).

4.1.4 Objective function

During the generation of master routes, the model objective is to

$$\text{minimise} \quad \sum_{k\in\mathcal{K}} C_k y_k + \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ijk} x_{ijk} + \sum_{k\in\mathcal{K}} \beta_k (T_{n+1,k} - T_{0,k}) + \rho\alpha_2. \tag{4.18}$$

This objective function takes into account the fixed costs associated with utilising vehicles (the first term above), the variable costs associated with actual vehicle trips (the second term), the total time duration of the vehicle trips (the third term), and the total number of penalised arc overlaps among the master routes (the last term).

In the first term in (4.18), the fixed cost C_k associated with each vehicle is added to the objective function value if the vehicle is utilised (*i.e.* if $y_k = 1$), but no contribution is made if the vehicle is not utilised (*i.e.* if $y_k = 0$). This ensures that the number of vehicles utilised is minimised. By summing across all elements in the matrix $\boldsymbol{X}$, the second term in the function in (4.18) captures the variable cost of every vehicle's route. If vehicle k travels from customer i to customer j (*i.e.* if $x_{ijk} = 1$), the associated variable cost, denoted by c_{ijk}, contributes to the objective function value; otherwise no contribution is made (*i.e.* if $x_{ijk} = 0$). In the third term of the function in (4.18), the trip duration of vehicle k is calculated as $T_{n+1,k} - T_{0,k}$ and multiplied by the cost coefficient β_k. This ensures that the trip duration of each vehicle is minimised by scheduling the start of service at each customer as early as possible and ensuring that a vehicle k which does not visit customer i will have a service start time of $T_{ik} = 0$ at that customer. The final term in (4.18) ensures that the number of penalised arc overlaps among the master routes are minimised.

4.1.5 Symmetry-breaking constraints

Symmetry breaking constraints of the form

$$y_k \quad \geq \quad y_{k+1}, \quad k = \sum_{\ell\in\{1,\dots,j-1\}} K_\ell + 1, \dots, \sum_{\ell\in\{1,\dots,j\}} K_\ell - 1, \quad j \in \{0,\dots,|\mathcal{L}|-1\} \tag{4.19}$$

may additionally be imposed so as to speed up any exact solution duration of the model.

4.2 Exact model solution approach

The model derived in §4.1 was implemented in CPLEX for verification purposes. CPLEX is an optimisation software environment designed to solve mixed IP problems to optimality by invoking a parallel algorithm based on the branch-and-cut method described in §3.1.4. The model of §4.1 was implemented in CPLEX *via* its Python programming language interface, allowing for easy generation of input data and visualisation of solutions, thereby facilitating verification of the model.

The code for declaring the decision variables, implementing the objective function, and enforcing the constraints of the model in the Python interface of CPLEX is shown in Figure 4.1. The first two sets of decision variables, captured in the matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, are defined as binary variables and are therefore already constrained to the set $\{0, 1\}$ as a result enforcing constraint sets (4.15) and (4.16), respectively. The following two sets of decision variables, captured in the matrices $\boldsymbol{T}$ and $\boldsymbol{R}$, are defined as continuous variables with a lower bound of zero already defined for both sets of decision variables, thereby implicitly imposing constraint sets (4.17) and (4.12), respectively. The final decision variable, α_2, is also defined as a continuous variable. Furthermore, the variable name *vehicles* represents the set $\mathcal{K}$, whereas the variable name $K[\ell]$ represents the parameter K_ℓ.

```python
# Objective function in (6.18):
mdl.minimize(mdl.sum(y[k]*C[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(beta[k]*(T[n+1,k]-T[0,k]) for k in vehicles) + rho*alpha2)

# Constraint set in (6.1):
mdl.add_constraints(mdl.sum(x[i,j,k] for j in delta_plus[i] for k in vehicles) == f[i] for i in V_)
# Constraint set in (6.2):
mdl.add_constraints(mdl.sum(x[0,j,k] for j in delta_plus[0]) == y[k] for k in vehicles)
# Constraint set in (6.3):
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 V_ for k in vehicles)
# Constraint set in (6.4):
mdl.add_constraints(mdl.sum(x[i,n+1,k] for i in delta_minus[n+1]) == y[k] for k in vehicles)
# Constraint set in (6.5):
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 k in vehicles for (i,j) in A)
# Constraint set in (6.6):
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)
# Constraint set in (6.7):
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)
# Constraint set in (6.8):
mdl.add_constraints(mdl.sum(q[i]/f[i]*mdl.sum(x[i,j,k] for j in delta_plus[i]) for i in V_) <= Q[k]*y[k] for k in vehicles)
# Constraint set in (6.9):
mdl.add_constraints(mdl.sum(x[i,j,k] for j in delta_plus[i]) <= g[i,k] for i in V_ for k in vehicles)
# Constraint set in (6.10):
mdl.add_constraints(mdl.sum(x[i,j,k] for (i,j) in A) <= (n+1)*y[k] for k in vehicles)
# Constraint set in (6.11):
mdl.add_constraints(mdl.sum(x[i,j,k] + x[j,i,k] for k in vehicles) - 1 <= r[i,j] for (i,j) in A)
# Constraint set in (6.13):
mdl.add_constraint(mdl.sum(r[i,j]/2 for (i,j) in A) <= alpha1 + alpha2)
# Symmetry-breaking constraints in (6.18):
for j in range(0,len(L)):
    mdl.add_constraints(y[k]>=y[k+1] for k in range(sum(K[l] for l in range(0,j))+1, sum(K[l] for l in range(0,j+1))))
```

FIGURE 4.1: *The code for the implementation of the model in §4.1 in the Python interface of CPLEX.*

In the remainder of this section, a small example problem instance containing ten customers is considered in order to illustrate the input data and verify the output data of an instance of the model (4.1)–(4.18). The information associated with each vertex in this problem instance was randomly generated and is presented in Table 4.1, whereas information about the types of delivery vehicles are presented in Table 4.2.

In the pre-processing code to the model implementation, the travel distances between vertices were calculated as the Euclidean distances between the vertices, resulting in the symmetric

TABLE 4.1: *The input data related to vertices in an illustrative example problem instance. The horizontal axis and vertical axis coordinates of customers are denoted by ζ_i and η_i, respectively, and is measured in kilometres. The number of visits required by each customer is denoted by f_i. Furthermore, the demand volume in cubic metres associated with each customer is denoted by q_i, and the service duration in minutes at each customer is denoted by s_i. Finally, the time-window start time and end time associated with each customer is denoted by a_i and b_i, respectively, and is measured in minutes from the start of the day.*

Vertex, i	ζ_i	η_i	f_i	q_i	s_i	a	b
0	50	50	–	–	0	0	600
1	60.28	54.49	1	14.61	53.84	0	270
2	42.37	64.59	2	17.81	36.71	270	540
3	43.76	89.18	1	11.18	43.55	0	270
4	96.37	38.34	1	16.40	59.20	270	540
5	79.17	52.89	1	11.43	44.30	270	540
6	56.80	92.56	2	19.45	39.17	270	540
7	7.10	8.71	2	15.22	32.83	270	540
8	2.02	83.26	2	14.15	31.22	270	540
9	77.82	87.00	1	12.65	47.94	0	270
10	97.86	79.92	2	17.74	36.61	0	270

TABLE 4.2: *The input data related to vehicle types in an illustrative example problem instance. The fixed cost associated with each delivery vehicle in Rand is denoted by C_z, whereas the maximum cargo capacity in cubic metres of each delivery vehicle is denoted by Q_z. The variable cost in Rand per kilometre travelled of each delivery vehicle is denoted by h_z. A speed factor, denoted by v_z, is associated with each type of delivery vehicle. Finally, each type of delivery vehicle is associated with a cost coefficient, denoted by β_z Rand per minute, with which the duration of each route is penalised in the objective function.*

Vehicle type, z	Size	C_z	Q_z	h_z	v_z	β_z
1	Small	3 000	30	5.85	0.9	0.1
2	Medium	3 250	50	6.175	1	0.1
3	Large	3 500	70	6.5	1.1	0.1

distance matrix

$$\boldsymbol{d} = \begin{bmatrix} - & 11.21 & 16.47 & 39.67 & 47.81 & 29.32 & 43.10 & 59.54 & 58.38 & 46.29 & 56.44 \\ 11.21 & - & 20.56 & 38.42 & 39.54 & 18.96 & 38.23 & 70.16 & 64.97 & 36.94 & 45.38 \\ 16.47 & 20.56 & - & 24.63 & 60.04 & 38.62 & 31.48 & 66.07 & 44.46 & 41.94 & 57.57 \\ 39.67 & 38.42 & 24.63 & - & 73.15 & 50.70 & 13.48 & 88.42 & 42.15 & 34.13 & 54.89 \\ 47.81 & 39.54 & 60.04 & 73.15 & - & 22.52 & 67.12 & 94.05 & 104.49 & 52.07 & 41.60 \\ 29.32 & 18.96 & 38.62 & 50.70 & 22.52 & - & 45.54 & 84.53 & 82.91 & 34.14 & 32.86 \\ 43.10 & 38.23 & 31.48 & 13.48 & 67.12 & 45.54 & - & 97.47 & 55.57 & 21.73 & 42.96 \\ 59.54 & 70.16 & 66.07 & 88.42 & 94.05 & 84.53 & 97.47 & - & 74.72 & 105.50 & 115.36 \\ 58.38 & 64.97 & 44.46 & 42.15 & 104.49 & 82.91 & 55.57 & 74.72 & - & 75.89 & 95.90 \\ 46.29 & 36.94 & 41.94 & 34.13 & 52.07 & 34.14 & 21.73 & 105.50 & 75.8 & - & 21.26 \\ 56.44 & 45.38 & 57.57 & 54.89 & 41.60 & 32.86 & 42.96 & 115.36 & 95.90 & 21.26 & - \end{bmatrix}.$$

The travel time matrix $\boldsymbol{t_k}$ for each delivery vehicle $k \in \mathcal{K}$ was then calculated by multiplying each value in the distance matrix by the speed parameter v_k of the corresponding delivery vehicle. Similarly, the cost matrix $\boldsymbol{c_k}$ for each delivery vehicle $k \in \mathcal{K}$ was calculated by multiplying each value in the distance matrix by the cost parameter h_k of the corresponding delivery vehicle.

In this small problem instance, two of each type of delivery vehicle is assumed to be available to service customers. The vehicle type $\ell \in \mathcal{L}$ and corresponding information associated with each delivery vehicle $k \in \mathcal{K}$ are indicated in Table 4.3. Furthermore, the number of arc overlaps allowed without penalty was assigned the value $\alpha_1 = 1$, and the cost coefficient associated with penalising an arc overlap was assigned the value $\rho = 200$ kilometres.

Table 4.3: *The input data related to the fleet of vehicles available to service customers in the example problem instance.*

Vehicle number, k	Type, z	Size	C_k	Q_k	h_k	v_k	β_k
1	1	Small	3 000	30	5.85	0.9	0.1
2	1	Small	3 000	30	5.85	0.9	0.1
3	2	Medium	3 250	50	6.175	1	0.1
4	2	Medium	3 250	50	6.175	1	0.1
5	3	Large	3 500	70	6.5	1.1	0.1
6	3	Large	3 500	70	6.5	1.1	0.1

In the post-processing code to the model implementation, the decision variable values are processed and routes are deduced for each delivery vehicle. These routes are stored in a text file, along with additional information about the solution, such as the run time required and the optimal objective function value. The routes are also plotted in a figure, using the *Matplotlib* package in the Python programming language, and is stored under an appropriate name specifying the particular problem instance.

The solution to the small, illustrative example problem instance described above returned by CPLEX is illustrated graphically in Figure 4.2. The depot is represented by the black square, whereas each customer is represented by either a yellow vertex or a green vertex. A yellow vertex represents a customer $i \in \mathcal{V}$ having a morning time-window (*i.e.* $a_i = 0$ minutes to $b_i = 270$ minutes), whereas a green vertex represents a customer $i \in \mathcal{V}$ having an afternoon time-window (*i.e.* $a_i = 270$ minutes to $b_i = 540$ minutes). Furthermore, the index $i \in \mathcal{V}$ associated with each customer is indicated, along with the required number of visits f_i associated with the customer, indicated in brackets. This solution corresponds to an objective function value of R 14 876.35, comprising a fixed cost of R 10 000, a variable cost of R 4 733.91, a trip duration penalty cost of R 142.44, and an overlap penalty cost of zero Rand. In this solution, three delivery vehicles (two medium delivery vehicles (type $\ell = 2$) and one large delivery vehicles (type $\ell = 3$)) are utilised. Moreover, the number of penalised overlaps, denoted by α_2, is zero.

The route of each delivery vehicle, the volume of commodities delivered to each customer, the total volume of commodities delivered by each delivery vehicle, and the service start times at customers are summarised in Table 4.4. By inspecting these output data, it may be confirmed that the solution retured by CPLEX satisfies all of the constraints in the model and is therefore feasible. Each delivery vehicle utilised departs from the depot and returns to the depot after having serviced the customers along its route. Each customer is also visited the specified number of times and the service at each customer starts during its specified time-window. Furthermore, each customer's demand is satisfied and the capacity of each delivery vehicle utilised is not exceeded. The overall utilisation of the capacity of delivery vehicles used is 88.59%.

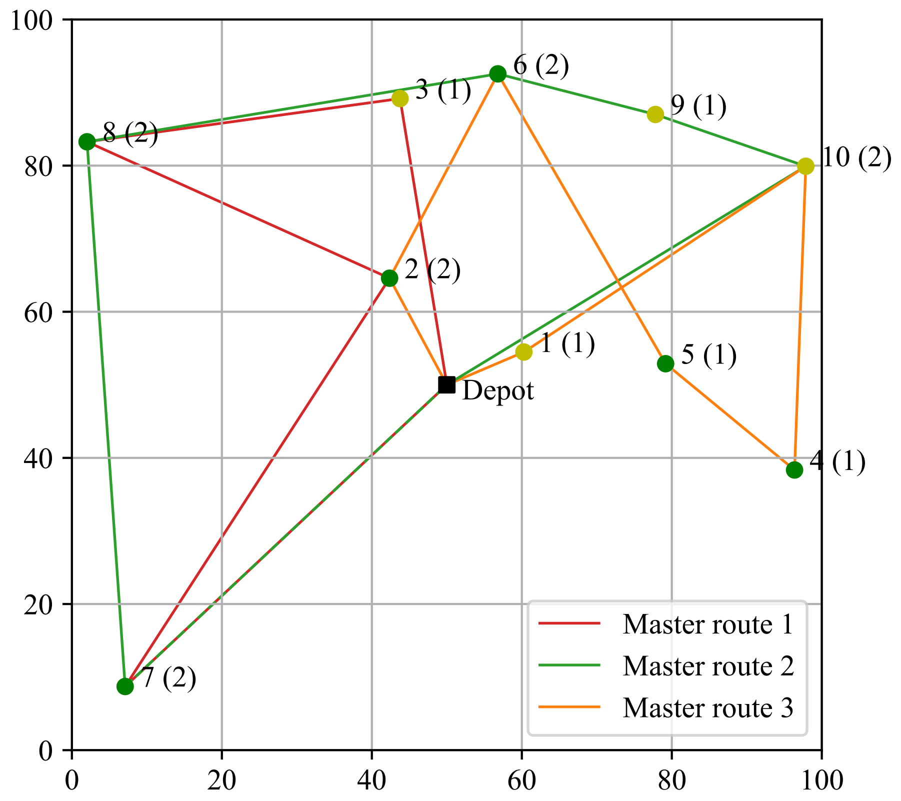


Figure 4.2: *Optimal solution to a small, illustrative example problem instance to the model* (4.1)–(4.18) *for the strategic phase of the FVRP.*

4.3 Systematic model verification

In order to ensure that the model derivation in §4.1 and its implementation in §4.2 are correct, further verification was performed. This involved solving the model for sixty randomly generated instances of the model (4.1)–(4.18) and ensuring that feasible solutions were returned by CPLEX, similar to the procedure followed for the small, illustrative example problem instance solved in §4.2.

In order to generate reproducible synthetic problem instances of the model (4.1)–(4.18) for the strategic phase of the FVRP, a problem instance generator was created in the Python programming language. This problem instance generator may be downloaded from the author's Gitthub repository[1]. The synthetic problem instance generator takes nine parameters as input. First, a width, denoted by w_c, is specified. The Cartesian coordinates of customers are uniformly distributed within a $w_c \times w_c$ square. Another width, denoted by w_d, specifies the location in which the depot may be placed. The location of the depot is uniformly placed in a $w_d \times w_d$ square which shares the same midpoint as the $w_c \times w_c$ square. When specifying $w_d = 0$, the

[1]Gitthub repository for producing problem instances of the model for the strategic phase of the FVRP: https://github.com/JacobusKing/Problem-Instance-Generator-for-Phase-1.git

TABLE 4.4: *Output data related to the optimal solution to a small illustrative example problem instance of the model for the strategic phase of the FVRP.*

Vehicle, k	From i	To j	q_j/f_j	T_{jk}
3	Depot	3	11.18	182.40
	3	8	7.07	270
	8	2	8.90	347.68
	2	7	7.61	453.43
	7	Depot	—	548.47
Total		34.76		
4	Depot	10	8.87	140.52
	10	9	12.65	199.35
	9	6	9.72	270
	6	8	7.07	367.24
	8	7	7.61	476.54
	7	Depot	—	571.59
Total		45.92		
5	Depot	1	14.61	75.17
	1	10	8.87	183.47
	10	4	16.40	270
	4	5	11.43	356.22
	5	6	9.72	455.17
	6	2	8.90	532.12
	2	Depot	—	588.58
Total		69.93		

depot is therefore placed in the centre of the $w_c \times w_c$ unit square. The travel times and travel costs associated with each arc are calculated based on the Euclidean distance between vertices.

A minimum and maximum demand volume, denoted by q_{min} and q_{max}, respectively, is also specified. The demand q_i of each customer $i \in \mathcal{V}$ is randomly generated according to a uniform distribution so that $q_{min} \leq q_i \leq q_{max}$. The service duration s_i associated with each customer $i \in \mathcal{V}$ is then calculated as $s_i = q_i\phi + \psi$, where ϕ denotes the variable service rate and ψ denotes the fixed service duration. A storage capacity e_i, assigned to each customer $i \in \mathcal{V}$, is uniformly distributed between a minimum storage capacity, denoted by e_{min}, and a maximum storage capacity, denoted by e_{max}. The number of visits f_i associated with each customer $i \in \mathcal{V}$ is then determined as q_i/e_i, rounded up to the nearest integer, since this represents the minimum number of times that a customer must be serviced in order to satisfy the demand associated with that customer without delivering more commodity volumes during a single visit than the storage capacity of the customer. A maximum time-window end time, denoted by b_{max}, is specified and represents the time at which service may start at the customer with the latest ending time-window. Customers are randomly assigned either a morning time-window (*i.e.* between 0 and $b_{max}/2$ minutes) or an afternoon time-window (*i.e.* between $b_{max}/2$ and b_{max} minutes) and the time-window of the depot is specified as between 0 and $b_{max} + 60$. The number of arc overlaps allowed without penalty, denoted by α_1, as well as the cost coefficient associated with penalising the number of arc overlaps, denoted by ρ, must also be specified. The feasible vehicle-to-customer visitation assignments, denoted by g_i, associated with each customer $i \in \mathcal{V}$ is simply assigned in such a way that each customer may be serviced by all vehicle types.

The output of the problem instance generator is a *Problem_Instance* class consisting of multiple Python dictionaries[2], one for each input parameter to the model (4.1)–(4.18) for the strategic phase of the FVRP, specifying the value associated with each customer for the corresponding parameter. Finally, an arbitrary number of problem instances may be created by specifying the number of customers in the problem instance, denoted by n and the seed number, denoted by $seed_1$, which represents the problem instance number. The standard input parameters of the problem instance generator (used if no value is specified for a parameter) are $w_c = 100$, $w_d = 0$, $q_{min} = 10$, $q_{max} = 20$, $\phi = 3$, $\psi = 10$, $b_{max} = 540$, $\alpha_1 = 1$, and $\rho = 200$. The only compulsory input parameters required for the generation of a problem instance is therefore the number of customers in each problem instance n and a desired seed number $seed_1$. The small problem instance solved in §4.2 can be generated by providing the parameters $n = 10$ and $seed_1 = 1$ to the problem instance generator.

The sixty problem instances solved in pursuit of further verifying the model (4.1)–(4.18) for the strategic phase of the FVRP in §4.1 and its implementation in §4.2 were generated by providing all combinations of the input parameters $n = 8, \ldots, 13$ and $seed_1 = 1, \ldots, 10$ to the problem instance generator. The information associated with the vehicle types to which each of the available delivery vehicles may belong is the same as the information presented in Table 4.2, and five delivery vehicles of each type were made available to service customers. The synthetic problem instances were solved on a computer with an Intel Core i7 CPU operating at 2.90GHz with 16GB of memory. Furthermore, a run time limit of eight hours was specified for each problem instance. The objective function value, required run time, and remaining optimality gap obtained when solving each of these sixty instances using CPLEX are summarised in Table 4.5. The memory of the computer was exceeded when solving some of the instances before the eight hours run time limit was reached. In these cases, the best solution found and the remaining optimality gap at that point in time were recorded, resulting in a run time shorter than the run time limit (of 28 800 seconds), yet with a positive optimality gap remaining.

For each of these synthetic problem instances, output data pertaining to the final solution returned by CPLEX was inspected and compared with the input parameters of the problem instance in order to verify feasibility of the solution. The following aspects of the output data were inspected to ensure feasibility:

- Each delivery vehicle utilised must be assigned a route that departs from the depot and returns to the depot after having serviced the customers along its route.
- The sum of all delivery quantities assigned to each delivery vehicle utilised may not exceed the capacity associated with the delivery vehicle.
- The sum of all delivery quantities received by each customer must equal the total volume of demand associated with the customer.
- Customers may only be assigned to delivery vehicles that are considered utilised and of a type compatible with visiting the customer.
- The service start time at each customer must be within its specified time-window.
- The service start time at each customer must be later than the service start time at the previous customer along the route, or the departure time at the depot if it is the first customer along the route.

[2]A Python dictionary is an unordered and changeable container that stores a value for each unique key in the dictionary. The f-values of a problem instance containing five customers, for example, are stored in a dictionary such that the keys are the customer indices, and the values of these keys their corresponding f-values in the form $f = \{1 : 1,\ 2 : 2,\ 3 : 1,\ 4 : 2,\ 5 : 1\}$.

TABLE 4.5: *The instance number, number of customers (n), seed number ($seed_1$), run time, and remaining optimality gap for each instance of the model (4.1)–(4.18) for the strategic phase of the FVRP solved according to the exact solution approach.*

Inst	n	$seed_1$	z	*Time*	*Gap*	Inst	n	$seed_1$	z	*Time*	*Gap*
1	8	1	13 794.81	2.55	0.00	31	11	1	17 325.52	3 356.41	0.00
2	8	2	12 624.84	2.70	0.00	32	11	2	17 123.57	227.72	0.00
3	8	3	11 583.99	9.27	0.00	33	11	3	12 940.34	4.11	0.00
4	8	4	12 421.91	1.39	0.00	34	11	4	17 516.01	954.27	0.00
5	8	5	13 587.02	0.78	0.00	35	11	5	18 242.14	112.61	0.00
6	8	6	9 943.26	0.42	0.00	36	11	6	13 520.00	27.28	0.00
7	8	7	14 559.62	4.09	0.00	37	11	7	18 003.02	31.25	0.00
8	8	8	13 365.80	1.19	0.00	38	11	8	13 729.71	4.39	0.00
9	8	9	8 750.91	0.13	0.00	39	11	9	13 852.75	1.44	0.00
10	8	10	17 276.51	10.19	0.00	40	11	10	18 464.30	310.33	0.00
11	9	1	14 571.51	3.45	0.00	41	12	1	17 901.95	4 727.39	0.00
12	9	2	12 939.48	4.95	0.00	42	12	2	17 099.55	14 265.33	0.00
13	9	3	9 241.64	0.16	0.00	43	12	3	13 716.29	149.70	0.00
14	9	4	13 248.37	12.03	0.00	44	12	4	20 116.82	28 800.00	0.15
15	9	5	17 102.64	8.83	0.00	45	12	5	20 595.02	1 198.16	0.00
16	9	6	12 918.15	4.19	0.00	46	12	6	13 865.97	1 597.08	0.00
17	9	7	16 068.26	110.44	0.00	47	12	7	17 373.35	3 467.70	0.00
18	9	8	17 501.68	67.08	0.00	48	12	8	17 483.07	355.70	0.00
19	9	9	12 989.34	6.78	0.00	49	12	9	16 829.56	211.16	0.00
20	9	10	17 837.26	289.27	0.00	50	12	10	22 489.83	28 800.00	0.12
21	10	1	14 876.35	11.41	0.00	51	13	1	18 883.85	28 800.00	0.01
22	10	2	13 066.65	11.39	0.00	52	13	2	21 310.63	28 800.00	0.12
23	10	3	9 384.24	0.30	0.00	53	13	3	17 732.17	28 800.00	0.01
24	10	4	16 389.40	2118.56	0.00	54	13	4	20 968.86	13 013.00	0.14
25	10	5	17 928.33	15.17	0.00	55	13	5	22 963.72	1 463.55	0.00
26	10	6	12 563.86	10.83	0.00	56	13	6	17 017.14	726.84	0.00
27	10	7	17 511.59	155.27	0.00	57	13	7	18 370.59	1 421.38	0.00
28	10	8	17 150.79	143.59	0.00	58	13	8	17 174.01	317.91	0.00
29	10	9	12 886.50	2.05	0.00	59	13	9	17 084.55	116.83	0.00
30	10	10	18 106.12	14.91	0.00	60	13	10	22 489.83	28 800.00	0.12

- The number of penalised arc overlaps α_2 must equal the difference between the total number of arc overlaps in the solution and α_1 or it must assume a value of zero if the aforementioned value is less than zero.

The final solution returned by CPLEX for each of the sixty problem instances was inspected and the above requirements were indeed satisfied in each case. The model derived in §4.1 and its implementation in §4.2 has therefore been verified empirically.

4.4 Time complexity of exact model solution approach

In order to determine the relationship between the number of customers in a randomly generated instance of the model (4.1)–(4.18) and the run time required by CPLEX to find an optimal solution to the corresponding problem instance, the information summarised in Table 4.5 was analysed. The run times of the exact solution approach for the model (4.1)–(4.18) of the strategic FVRP phase are plotted on a logarithmic scale for different numbers of customers in Figure 4.3. The remaining optimality gaps associated with solving the instances are plotted in Figure 4.4.

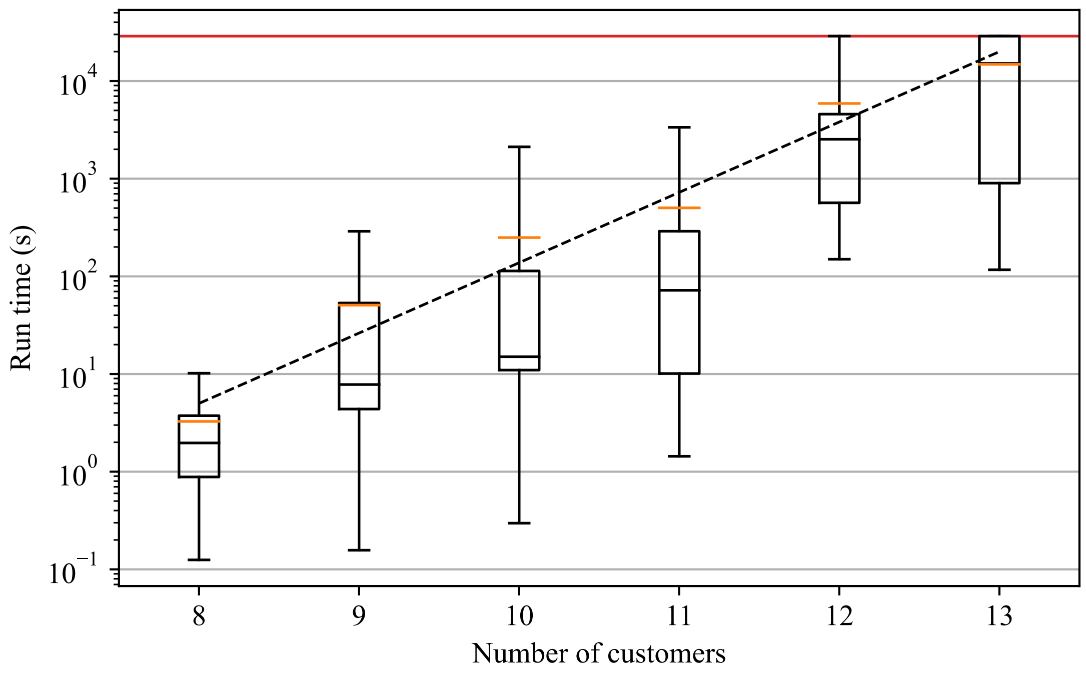


FIGURE 4.3: *The run times recorded when solving synthetic instances of the model (4.1)–(4.18) exactly, involving different numbers of customers, during the strategic phase of the FVRP. The red horizontal line represents the eight hour run time limit specified when solving each problem instance. The mean trend of the run times is indicated by means of a dotted line.*

Despite being plotted on a logarithmic scale, the aforementioned run times seem to increase with a strong linear trend as the number of customers in the problem instance increases. This clearly demonstrates that the computational complexity of the model increases exponentially as the number of customers in the problem instances increase for a fixed number of available delivery vehicles. Furthermore, seven of the problem instances (two instances containing twelve customers and five instances containing thirteen customers) could not be solved to optimality due to the computer either running out of memory or CPLEX not being able to solve the instances to optimality within the eight hour time limit, resulting in the optimality gaps observed in Figure 4.4. The exact solution approach for the model of the strategic phase of the FVRP would therefore not seem to be scalable to the size of real-world problem instances in which it may be required to solve problem instances, typically containing hundreds of customers. This motivates the need for an approximate solution approach in which high-quality, but not necessarily optimal, solutions may be achieved for large problem instances within an acceptable time-frame.

4.5 Approximate solution approach

An approximate solution approach is proposed in this section for the model (4.1)–(4.18) of the strategic phase of the FVRP. This solution approach is based on the second version of the

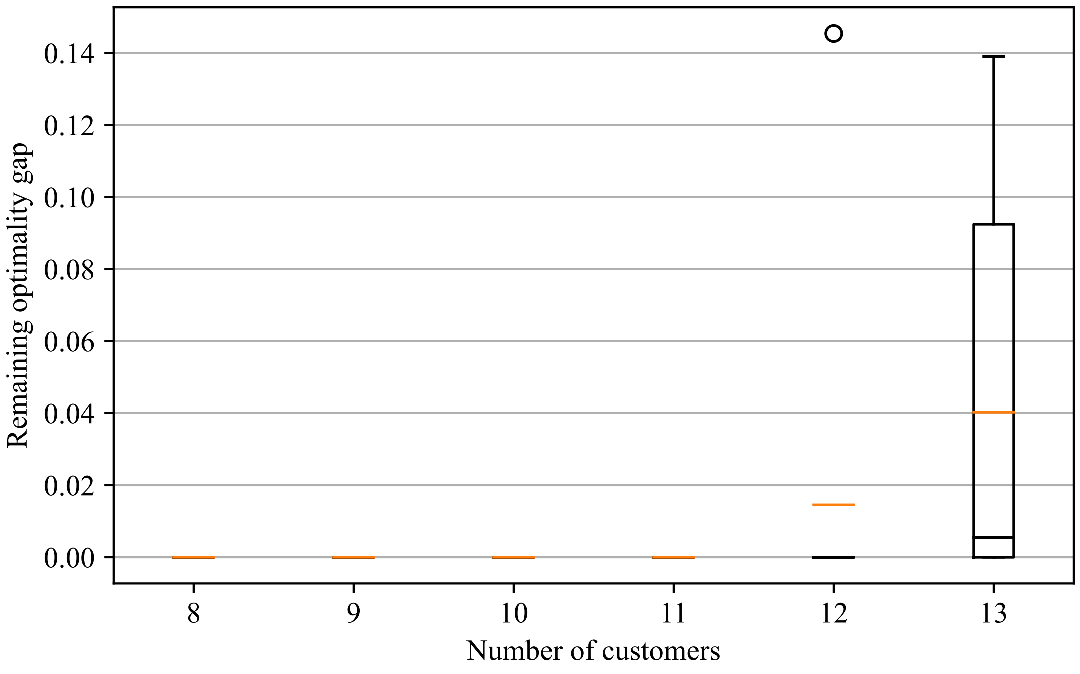


Figure 4.4: *The optimality gaps recorded when solving synthetic instances of the model (4.1)–(4.18) involving different numbers of customers during the strategic phase of the FVRP.*

HGSADC algorithm proposed by Vidal *et al.* [140], described in §3.5, with some adaptions made to the structure and the working of the algorithm in order to be able to accommodate instances of the model in (4.1)–(4.18).

A pseudo-code description of the proposed solution approach is presented in §4.5.1. This is followed by a description of the adaptions made to the original HGSADC algorithm. Thereafter, an exact solution approach component included in the proposed approximate solution approach is discussed in §4.5.2. Furthermore, additions of a heterogeneous fleet attribute and a multiple visits attribute are described in §4.5.3 and §4.5.4, respectively. The approach adopted towards penalising arc overlaps in the master routes is elucidated in §4.5.5, and this is followed by a discussion in §4.5.6 on the manner in which the service start times of delivery vehicles at customers were determined. The parameter values adopted during the implementation of the approximate solution approach are reviewed in §4.5.7, whereas the classes and functions created during the implementation of the approximate solution approach in the programming language Python are discussed in §4.5.8 and §4.5.9, respectively.

4.5.1 Pseudo-code description

The proposed approximate solution approach is described in pseudo-code form in Algorithm 4.1. The pseudo-code description is similar to that of the HGSADC algorithm proffered in Algorithm 3.4, but with the addition of lines 20–25 in which an exact solution component is included. The proposed solution approach begins by initialising a population consisting of a feasible subpopulation and an infeasible subpopulation, by randomly generating individuals representing "candidate" solutions. The main loop of the algorithm is then executed for It_{NI} non-improving iterations or for a maximum duration of T_{max} minutes, after which the best solution found is returned. During each iteration, two parent solutions, $P1$ and $P2$, are selected according to a

Algorithm 4.1: Approximate solution approach for the model (4.1)–(4.18)

```
1  Initialise population
2  while number of iterations without imporvement $\leq It_{NI}$, and time $\leq T_{max}$ do
3      Select parent solutions $P_1$ and $P_2$
4      Create offspring $C$ from $P_1$ and $P_2$ (crossover)
5      Educate $C$ (local search procedure)
6      if $C$ infeasible then
7          Insert $C$ into infeasible subpopulation,
8          Repair with probability $P_{rep}$
9      if $C$ feasible then
10         Insert $C$ into feasible subpopulation
11     if maximum subpopulation size reached then
12         Select survivors
13     if best solution not improved for $It_{div}$ iterations then
14         Diversify population
15     Adjust penalty parameters for infeasibility
16     if number of iterations $= k \times It_{dec}$ where $k \in \mathbb{N}$ then
17         Decompose the master problem
18         Use HGSADC on each subproblem
19         Reconstitute three solutions, and insert them in the population
20     if number of iterations $= \ell \times It_{exact}$ where $\ell \in \mathbb{N}$ then
21         Select the best individual $I$ in the feasible subpopulation
22         Provide $I$ as a warm start to an exact solution approach for $T_{exact}$ seconds
23         Insert $I$ in the population
24         if $I$ is optimal then
25             Terminate the search
26 Return best feasible solution
```

binary tournament selection scheme from the union of the feasible and infeasible subpopulations, and an offspring solution C is created by invoking the PIX crossover operator. Each offspring solution created undergoes education during which two local search procedures are performed. If the offspring solution is not feasible, it is placed in the infeasible subpopulation and undergoes repair with probability P_{rep}, during which education is performed with increased penalty weights for infeasibility. If, however, the offspring solution created is feasible, it placed in the feasible subpopulation.

Both subpopulations are managed to contain between μ and $\mu + \lambda$ individuals. If any subpopulation contains more than $\mu + \lambda$ individuals, survivor selection takes place during which λ individuals are removed from the corresponding subpopulation. In order to explore a larger area of the search space, diversification takes place every It_{div} non-improving iterations, during which $\mu/3$ individuals are retained in each subpopulation and 4μ new individuals are generated randomly and introduced into the population in the same manner as when the population was initialised. Furthermore, in order to solve large problem instances effectively, decomposition occurs every It_{dec} iterations. During decomposition, the problem instance is decomposed into smaller subproblem instances and each subproblem instance is solved approximately after which they are reconstituted once again to obtain a complete solution which is inserted into the appropriate subpopulation. An exact solution approach is invoked every T_{exact} iterations. The best solution in the feasible subpopulation is provided as input to the exact solution approach, a practice referred to as providing a "warm" start to the exact model solution methodology, after which it is improved for T_{exact} seconds and the improved solution is re-introduced back into the feasible subpopulation. If a proven optimal solution is found by the exact solution approach, the execution of the algorithm is is terminated and this optimal solution is returned. Finally, upon termination of the algorithm, the best feasible solution found during the search is returned

4.5.2 Exact solution approach component

In order to combine the advantages of hybrid metaheuristics and exact solution approaches, such as the branch-and-cut method, an exact solution approach component is included in the approximate solution approach for the model of the strategic phase. After each It_{exact} iterations, the best solution in the feasible subpopulation uncovered by the adapted HGSADC algorithm is transformed into an appropriate format and provided as a warm start to the exact solution approach described in §4.2. This initial solution is then further improved for a duration of T_{exact} seconds *via* the Python interface in CPLEX after which the final solution returned by CPLEX is transformed into the format of an individual and inserted as an additional individual in the subpopulation of feasible individuals, and then the execution of the algorithm resumes.

This exact solution component is particularly efficient in quickly solving small problem instances to optimality, or proving that an optimal solution has already been found by the metaheuristic. For larger problem instances, the exact solution approach is often able to find a diverse improving solution quickly which, in turn, accelerates the search for improving feasible solutions by the metaheuristic. The parameters associated with this component are to be selected based on the exact solution approach adopted.

4.5.3 Heterogeneous fleet attribute

Whereas the original HGSADC algorithm is capable of solving VRP instances containing multiple depots, master routes are computed for a single depot and its assigned customers in the model (4.1)–(4.18) for the strategic phase of the FVRP. Assigning customers to depots is therefore not required in the FVRP. The model (4.1)–(4.18) for the strategic phase of the FVRP, however, makes provision for having a heterogeneous fleet of delivery vehicles stationed at the depot — an attribute not included in the original HGSADC algorithm. This difference in capability required by the model (4.1)–(4.18) may be accommodated by replacing the depot chromosome according to which individuals are represented, illustrated in Figure 3.11, with a *vehicle type* chromosome. Instead of assigning each customer to a depot, customers are assigned to multiple vehicle types, and a giant tour is generated for each decision period and vehicle type combination (p, ℓ). This giant tour is then partitioned into multiple routes, each performed by a delivery vehicle of type ℓ during decision period p. When evaluating a route, the fixed cost associated with the delivery vehicle to which the route is assigned, is added to the penalised cost. Moreover, the distance measure incorporated into the penalised cost of a route is adapted to take into account the different costs associated with travelling between vertices by the different vehicle types.

When initialising the population, individuals are created by randomly assigning each customer to a feasible visiting pattern, and then randomly adding the customer to the giant tour of any type of delivery vehicle within each of the decision periods associated with the assigned pattern. Furthermore, during any stage of the algorithm, a customer may only be assigned to a giant tour of a vehicle type and period combination (p, ℓ) if the customer is, in fact, allowed to be visited by the vehicle type ℓ, thereby ensuring feasible vehicle-customer assignments.

4.5.4 Multiple visits attribute

The original HGSADC algorithm does not allow for a customer to be visited more than once during a single decision period, although the proposed model for the strategic phase of the FVRP requires that some customers receive multiple visits by different master routes. Furthermore, the original HGSADC algorithm is capable of solving problem instances containing multiple decision periods organised into a planning horizon, whereas the model for the strategic phase of the FVRP computes master routes which are not associated with any specific decision period. This difference in capability required by the model (4.1)–(4.18) is accommodated by retaining the multiple decision periods attribute of the HGSADC algorithm and creating "dummy" decision periods when solving instances of the model (4.1)–(4.18). Customers requiring multiple visits by different master routes may then be visited during different dummy decision periods of the planning horizon. Once a final solution has been obtained, all routes in the dummy decision periods are simply combined into a single period, resulting in master routes that are not associated with any specific decision period. The number of dummy decision periods specified for each problem instance is at least as large as the largest number of visits required by any single customer.

4.5.5 Penalising the number of arc overlaps

The original HGSADC algorithm does not, of course, penalise overlapping master route arcs. When evaluating a solution, the number of penalised arc overlaps is therefore counted and penalised accordingly in the fitness values of an individual, as required in the model (4.1)–(4.18) for the strategic phase of the FVRP.

4.5.6 The service start times at customers

No mention was made by Vidal *et al.* [140] as to which approach should be followed to determine the service start time of delivery vehicles at customers in order to identify instances of time-warp in solutions. An approach was therefore adopted in which each delivery vehicle departs from its current location and arrives at its next location as early as possible. Although this approach may result in large amounts of vehicle waiting time, it results in the minimum amount of time-warp. At termination of the HGSADC algorithm, before returning the best feasible solution found during the entire search, this best feasible solution is once again provided as a warm start to CPLEX and is returned as soon as CPLEX recognises the warm start as a feasible solution. The optimal service start times for the given solution may then be deduced from the solution returned by CPLEX, which are returned as the service start times for the best solution returned by the approximate solution approach.

4.5.7 Parameters of the algorithm

Since the parameter values do not depend on the type of VRP being solved, all parameters included in both the proposed approximate solution approach and those that were in the original HGSADC algorithm were set to recommended values based on the parameter calibration performed by Vidal *et al.* [139, 140] and described in §3.5.4. The only parameter value which would seem to depend on the type of VRP being solved is the generation size λ which was fixed at the value proposed for the MDPVRP, since its structure is most similar to that of the FVRP. Additional parameters not included in the original HGSADC algorithm are the parameters T_{exact} and It_{exact} described in §4.5.2. When adopting the CPLEX model implementation as the exact solution approach, the parameters were set to $T_{exact} = 20n$ and $It_{exact} = It_{NI}/2$, allowing for sufficient time to find an improving solution each time the component is executed, and executing the component twice between the iterations during which an improving solution is found and before the maximum number of non-improving iterations is reached.

4.5.8 Classes created during the implementation

A alluded to, proposed approximate solution approach was implemented in the Python programming language, adopting an object oriented approach. The following classes were created during the implementation:

The Globals class stores global variables which should be accessible to all objects and functions in the implementation.

The ProblemInstance class stores the input parameters required in an instance of the model (4.1)–(4.18) for the strategic phase of the FVRP.

The Settings class stores all the configuration parameters of the metaheuristic.

The Fleet class stores information associated with the available fleet of delivery vehicles stationed at the depot.

The Individual class stores an individual (solution) in the population as well as all information associated with the individual.

The Population class stores the current population of individuals.

The Phase_I class stores the main function responsible for executing the metaheuristic.

Variables stored in the Globals class include an object of the ProblemInstance class, an object of the Settings class, and an object of the Population class. This allows all functions and classes access to these variables without the need to pass information regularly between functions. The input parameters stored in the ProblemInstance class may either be provided manually or generated by invoking the problem instance generator described in §4.3. Furthermore, an object of the Fleet class also forms part of the ProblemInstance class. An object of the ProblemInstance class cannot be created without specifying all input parameters required for an instance of the model (4.1)–(4.18), therefore serving the purpose of input data validation. When an object of the ProblemInstance class is created, the pre-processing of input parameters is also performed.

The Individual class calculates and stores all information associated with an individual, such as the chromosomes and fitness values associated with the individual. This information is also continually updated throughout the execution of the metaheuristic in order to store new fitness values after having updated penalty parameters or after operations, such as Education, have been performed on the individual. An object of the Population class stores the current feasible and infeasible subpopulations which, in turn, each consists of potentially many objects of the Individual class.

Finally, the Phase_I class stores the main function which executes the metaheuristic by accessing and changing the variables stored in an object of the Globals class.

4.5.9 Functions created during the implementation

The main functions used during the execution of the metaheuristic are as follows:

The ParentSelection function forms part of the Population class and returns two parents selected from the population, each an object of the Individual class, to participate in the crossover procedure.

The Crossover function is a stand-alone function which takes two individuals, each an object of the Individual class, as input to perform the PIX crossover, and returns the offspring generated as an object of the Individual class.

The Evaluate function forms part of the Individual class and is responsible for evaluating all fitness measures of an individual based on the three chromosomes representing it.

The EvaluatePopulation function forms part of the Population class and facilitates the calculation of all fitness measures for individuals in the population.

The Educate function forms part of the Individual class and is responsible for performing the local search procedure after which the updated chromosomes representing an individual are stored and the individual is re-evaluated.

The Diversify function forms part of the Population class and its purpose is to carry out the population diversification component of the metaheuristic.

The SurvivorSelection function forms part of the Population class and carries out the survivor selection component of the metaheuristic.

The AdjustPPs function is a stand-alone function which adjusts the penalty parameters based on the number of feasible offspring generated.

The Decompose function is a stand-alone function which is responsible for carrying out the decomposition component of the metaheuristic.

The Solve function forms part of the Phase_1 class and executes the metaheuristic.

While many smaller functions not mentioned above also appear in the Python implementation of the approximate solution approach, the above-mentioned functions are the main functions utilised to execute the metaheuristic. Two smaller functions, for example, are a function for plotting the routes of a solution and a function responsible for counting the number of non-improving iterations. These functions typically form part of the Phase_I class.

When the Solve function is called, the metaheuristic is executed to solve an instance of the model (4.1)–(4.18) represented by the objects stored in the Globals class. The population of individuals, also stored in the Globals class, is continually updated and is accessible to all functions.

In the Decomposition function, the problem instance is decomposed into smaller subproblem instances and each subproblem instance is solved approximately by the proposed approximate solution approach. This is done by temporarily replacing the objects in the Globals class with objects representing the problem instance only containing those customers associated with the decomposed problem instance. Once each subproblem instance has been solved, the variables in the Globals class are replaced with the objects representing the original problem instance.

4.6 Systematic approximate solution approach verification

A verification of the approximate solution approach proposed for the strategic phase of the FVRP discussed in §4.5 was performed in order to ensure its capability of returning high-quality feasible solutions. The sixty problem instances solved during the systematic model verification performed in §4.3 were again solved, this time approximately, during a verification of the approximate solution approach. Furthermore, the solutions returned by CPLEX for these sixty instances were taken as reference points to verify the implementation of the approximate solution approach.

The run time and objective function value of the best solution found were recorded for each instance. Furthermore, a maximum run time of eight hours and a stopping criterion of five thousand non-improving iterations were specified for each instance. The average differences in objective function values of the solutions returned by the exact solution approach and those of the best solutions returned by the approximate solution approach are summarised in Table 4.6. The approximate solution approach returned solutions which are identical to those returned by CPLEX for fifty five of the sixty problem instances. Four of the solutions returned were worse than the solutions returned by the exact solution approach, whereas one solution returned was an improvement. This was possible since not all instances were solved to optimality by the exact solution approach. Furthermore, an average gap of 0.0154% was achieved for the 54 instances for which an optimal solution is known. The approximate solution approach is therefore considered to have been verified successfully.

A comparison between the run times recorded when solving the aforementioned sixty problem instances according to the exact solution approach and those required by the approximate solution approach is presented graphically on a logarithmic scale in Figure 4.5. The run times returned by the exact solution approach are indicated in green, whereas those returned by the approximate solution approach are indicated in blue. Furthermore, the average run time returned by each solution approach for each number of customers is indicated by the orange lines. It is evident that the slope of increase of the average run times required by the approximate solution approach is much smaller than that of the exact solution approach. The run times of the exact solution approach also fluctuated to a much larger extent than those of the approximate solution approach. It is important to note that the average run time of the exact solution approach for instances containing twelve or thirteen customers was affected by the

Table 4.6: *Mean percentage differences in objective function values, denoted by Δ Cost, between the solutions returned by the exact solution approach and those returned by the approximate solution approach of the model (4.1)–(4.18) for the strategic phase of the FVRP. The proportions of identical, worse, and improving solutions returned by the approximate solution approach are also shown.*

Number of customers, n	Δ Cost	Identical	Worse	Improving
8	0%	100%	0%	0%
9	0%	100%	0%	0%
10	0.0072%	90%	10%	0%
11	0.0403%	90%	10%	0%
12	0.0523%	80%	20%	0%
13	−0.3321%	80%	0%	20%

run time limit of eight hours imposed, whereas the approximate solution approach never met this stopping criterion. Moreover, the memory of the computer was never exceeded when solving instances approximately by invoking the approximate solution approach. A further set of problem instances (containing fourteen to eighteen customers) was also solved according to the approximate solution approach and a similar slope of increase was observed as for the smaller sized problem instances. The solution time incurred by the approximate solution approach is considered to be acceptable.

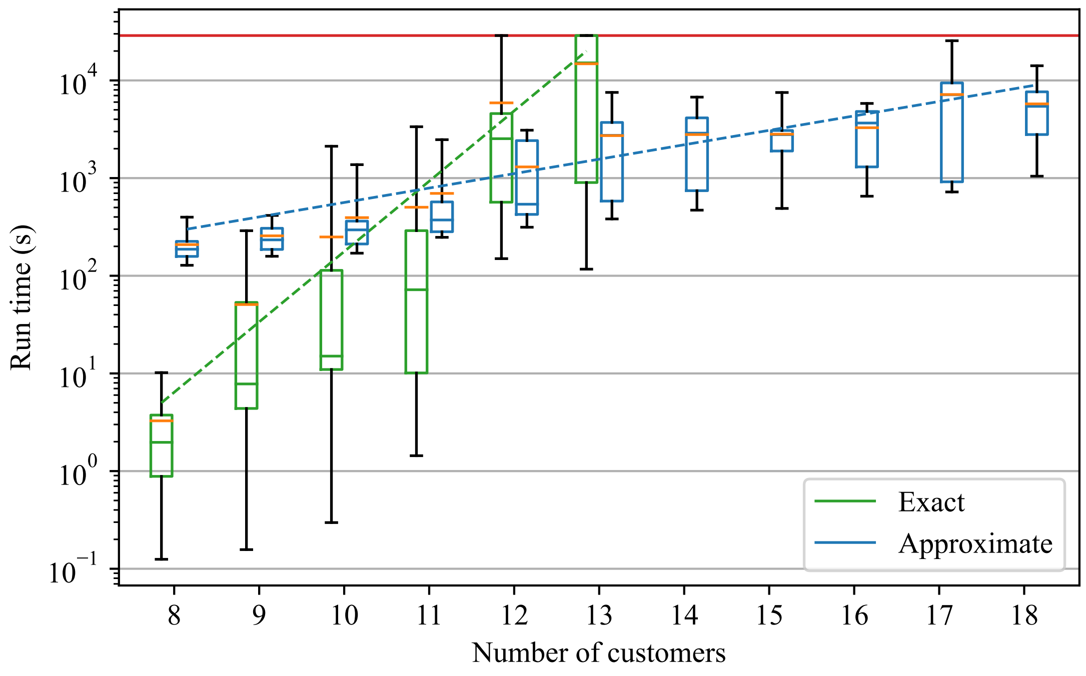


Figure 4.5: *The run times recorded when solving synthetic instances of the model (4.1)–(4.18), involving different numbers of customers, during the strategic phase of the FVRP exactly and approximately. The mean trends of the run times are indicated by means of dotted lines.*

4.7 Chapter summary

In this chapter, a mathematical model was proposed for the strategic phase of the FVRP. The model was derived in §4.1 after which an exact solution approach was proposed in §4.2 and implemented for the model in the CPLEX optimisation environment using its Python interface. This was followed by a systematic model verification in §4.3 in which a random problem instance generator was described and used to generate sixty problem instances for model verification purposes. The time complexity of the exact solution approach was discussed in §4.4 and it was found that the exact solution approach is not scalable to the size of real-world problem instances. An approximate solution approach was therefore proposed in §4.5, based on the HGSADC algorithm of Vidal *et al.* [140]. A systematic verification of the approximate solution approach followed in §4.6 during which it was found that the approximate solution approach is acceptable and that it is capable of returning high-quality solutions.

CHAPTER 5

Model for the operational phase

Contents

In this chapter, a mathematical model is proposed for the final phase of the FVRP. This model again assumes the form of a mixed binary programming problem. The model is applicable during the operational phase of the FVRP for which decision support is proffered by the framework of the previous chapter. First, the model is derived in §5.1, after which an exact solution approach for the model, involving the branch-and-cut method, is described in §5.2. The model is again implemented in the CPLEX optimisation environment in order to invoke this solution methodology. The methodology followed to verify this model implementation is discussed in §5.3. This is followed by a discussion on the approximate solution approach proposed for this model in §5.5. The chapter closes in §5.7 with a summary of its contents.

5.1 Model derivation

This section is devoted to a derivation of the aforementioned model for the operational phase of the FVRP. First, the various input data required are elucidated in §5.1.1 in terms of the model parameters required to configure an instance of the model. The decision variables of the model are then declared in §5.1.2, after which the constraints of the model are derived and explained in §5.1.3. The objective functions of the model are then derived and motivated in §5.1.4, and this is followed by the derivation of an additional constraint set in §5.1.5, aimed at improving the exact solution duration of the model.

5.1.1 Model parameters

In the model for the operational phase of the VRP proposed in this chapter, an input set of master routes is taken as blueprints to create actual delivery schedules for multiple decision periods in a planning horizon. Let $\mathcal{V} = \{0, 1, \dots, n, n+1\}$ again index the set of FVRP vertices, with 0 and $n+1$ representing the depot — the former representing the depot at vehicle departure and the latter representing the depot at vehicle return. The set $\mathcal{V}' = \mathcal{V} \setminus \{0, n+1\}$ furthermore represents the set of customers to be serviced. Let $\mathcal{A}$ denote the set of directed road links, called *arcs*, that delivery vehicles are allowed to traverse. The directed travel graph on which the FVRP is defined, is denoted by $G(\mathcal{V}, \mathcal{A})$, which has $\mathcal{V}$ as vertex set and $\mathcal{A}$ as arc set. Let $\delta^+(i)$ denote the set of vertices to which there are arcs from vertex $i \in \mathcal{V}$ in the travel graph $G(\mathcal{V}, \mathcal{A})$ and let $\delta^-(i)$ denote the set of vertices from which there are arcs to vertex $i \in \mathcal{V}'$.

In order to compute delivery schedules over a planning horizon containing multiple periods, let $\mathcal{P} = \{1, \dots, |\mathcal{P}|\}$ index the set of decision periods contained within the planning horizon. Furthermore, let $\mathcal{L} = \{1, 2, \dots, |\mathcal{L}|\}$ index the set of vehicle types to which each delivery vehicle may belong. Moreover, let $\mathcal{K} = \{1, 2, \dots, |\mathcal{K}|\}$ index the set of (heterogeneous) delivery vehicles available at the depot to service customers, ordered in such a way that delivery vehicles of the same type are indexed successively. Let K_ℓ denote the number of identical delivery vehicles of vehicle type $\ell \in \mathcal{L}$, let Q_k denote the cargo volume capacity of delivery vehicle $k \in \mathcal{K}$, and let C_k denote the fixed cost associated with utilising a delivery vehicle $k \in \mathcal{K}$. Also, let β_k be a cost coefficient associated with lengthening the trip duration of vehicle $k \in \mathcal{K}$ by one minute. Moreover, define the binary parameters

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

in order to allow for potential size restrictions preventing certain vehicles from visiting certain customers. Since different vehicle types may have different travel times and costs associated with them, let c_{ijk} denote the variable cost associated with vehicle $k \in \mathcal{K}$ when travelling from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$ and let t_{ijk} denote the expected time (in minutes) spent by vehicle $k \in \mathcal{K}$ travelling from vertex $i \in \mathcal{V}$ to vertex $j \in \mathcal{V}$.

In order to specify time-windows for each customer, let a_i denote the earliest possible vehicle service start time at customer $i \in \mathcal{V}'$ and let b_i denote the latest possible vehicle service start time at customer $i \in \mathcal{V}'$ (both measured in minutes after some fixed reference time). Also, let q_i denote the total demand volume of customer $i \in \mathcal{V}'$ over the entire planning horizon. Let e_i denote the maximum storage capacity (*i.e.* the maximum volume of commodities that can be delivered during a single decision period) at customer $i \in \mathcal{V}'$, measured in units of demand exhibited. Moreover, let F_i denote the number of visits required by a customer $i \in \mathcal{V}'$ during the planning horizon in order for its total demand q_i to be satisfied without exceeding its maximum storage capacity e_i during any decision period. The number of visits required by each customer over the entire planning horizon may therefore be calculated as $F_i = \lceil q_i/e_i \rceil$. Furthermore, let s_i denote the average service time duration at customer $i \in \mathcal{V}'$, measured in minutes and calculated as a function based on the demand volume q_i/F_i satisfied per visit during the planning horizon associated with each customer, to which a fixed service setup time is added. Finally, define the binary parameters

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

based on the master routes computed for the same depot and corresponding set of customers by solving the model for the strategic phase of the FVRP derived in §4.1.

5.1.2 Model variables

The binary variables

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

capture all vehicle flows and are stored in row i and column j of slice k in a three-dimensional $(n+2) \times (n+2) \times |\mathcal{K}|$ flow matrix $\boldsymbol{X}^p$ for each decision period $p \in \mathcal{P}$. The binary auxiliary variable

$$y_k^p = \begin{cases} 1 & \text{if vehicle } k \in \mathcal{K} \text{ is utilised during decision period } p \in \mathcal{P}, \\ 0 & \text{otherwise} \end{cases}$$

is also defined and stored in a $|\mathcal{K}| \times |\mathcal{P}|$ matrix $\boldsymbol{Y}$.

The service start time at vertex $i \in \mathcal{V}$, when serviced by vehicle $k \in \mathcal{K}$ during decision period $p \in \mathcal{P}$, is captured in a real-valued variable T_{ik}^p and stored in an $(n+2) \times |\mathcal{K}| \times |\mathcal{P}|$ matrix $\boldsymbol{T}$. The service start time of delivery vehicle k (if utilised) at vertex 0 during decision period p, is denoted by T_{0k}^p and represents the time at which the delivery vehicle departs from the depot during that period. In addition, the service start time at vertex $n+1$ of the same delivery vehicle during the same decision period (if utilised) is denoted by $T_{n+1,k}^p$ and represents the time at which the delivery vehicle returns to the depot, after having departed from the depot and having served all customers assigned to it during that decision period. The trip duration of vehicle $k \in \mathcal{K}$ (if utilised) during decision period $p \in \mathcal{P}$ is therefore $T_{n+1,k}^p - T_{0k}^p$.

5.1.3 Model constraints

A number of constraints are imposed in the model to ensure the practical feasibility of solutions. In order to ensure that each customer $i \in \mathcal{V}'$ is visited exactly F_i times throughout the planning horizon, the constraint set

$$\sum_{k \in \mathcal{K}} \sum_{j \in \delta^+(i)} \sum_{p \in \mathcal{P}} x_{ijk}^p = F_i, \qquad i \in \mathcal{V}' \tag{5.1}$$

is imposed. Moreover, in order to ensure that each customer $i \in \mathcal{V}'$ is visited no more than once during any single decision period within the planning horizon, the constraint set

$$\sum_{k \in \mathcal{K}} \sum_{j \in \delta^+(i)} x_{ijk}^p \leq 1, \qquad i \in \mathcal{V}', \ p \in \mathcal{P} \tag{5.2}$$

is imposed. In order to enforce a source (depot) to sink (depot) path for each vehicle utilised during each decision period, the three constraint sets

$$\begin{aligned} \sum_{j\in\delta^+(0)} x^p_{0jk} &= y^p_k, \quad k\in\mathcal{K},\ p\in\mathcal{P}, && (5.3)\\ \sum_{i\in\delta^-(j)} x^p_{ijk} - \sum_{i\in\delta^+(j)} x^p_{jik} &= 0, \quad j\in\mathcal{V}',\ k\in\mathcal{K},\ p\in\mathcal{P}, && (5.4)\\ \sum_{i\in\delta^-(n+1)} x^p_{i,n+1,k} &= y^p_k, \quad k\in\mathcal{K},\ p\in\mathcal{P} && (5.5) \end{aligned}$$

are imposed. Imposition of the constraint sets

$$\begin{aligned} T^p_{ik}+s_i+t_{ijk}-T^p_{jk} &\le M^p_{ijk}(1-x^p_{ijk}), && k\in\mathcal{K},\ (i,j)\in\mathcal{A},\ p\in\mathcal{P}, && (5.6)\\ a_i\sum_{j\in\delta^+_i} x^p_{ijk} \le T^p_{ik} &\le b_i\sum_{j\in\delta^+_i} x^p_{ijk} && i\in\mathcal{V}\setminus\{n+1\},\ k\in\mathcal{K},\ p\in\mathcal{P}, && (5.7)\\ a_{n+1}\sum_{i\in\delta^-_{n+1}} x^p_{i,n+1,k} \le T^p_{n+1,k} &\le b_{n+1}\sum_{j\in\delta^-_{n+1}} x^p_{j,n+1,k} && k\in\mathcal{K},\ p\in\mathcal{P} && (5.8) \end{aligned}$$

furthermore enforce adherence to customer delivery time-windows during every decision period, while constraint set (5.6) also implicitly avoids subtour formation along vehicle routes. The parameter M^p_{ijk} in constraint set (5.6) represents a large constant, which can be set to $\max\{b^p_i + s_i + t_{ijk} - a^p_j, 0\}$, with the maximum taken over all $i,j\in\mathcal{V}'$, $k\in\mathcal{K}$, and $p\in\mathcal{P}$. In order to ensure that each delivery vehicle utilised is able to service all customers along its assigned route during each decision period without exceeding its capacity, the constraint set

$$\sum_{i\in\mathcal{V}'}\sum_{j\in\delta^+_i} \frac{q_i}{F_i}\times x^p_{ijk} \le Q_k y^p_k, \quad k\in\mathcal{K},\ p\in\mathcal{P} \tag{5.9}$$

is imposed. In order to ensure that only vehicles which are, in fact, able to visit a customer is assigned to that customer during every decision period, the constraint set

$$\sum_{j\in\delta^+(i)} x^p_{ijk} \le g_{ik}, \quad i\in\mathcal{V}',\ k\in\mathcal{K},\ p\in\mathcal{P} \tag{5.10}$$

is imposed, while the constraint set

$$\sum_{(i,j)\in\mathcal{A}} x^p_{ijk} \le (n+1)y^p_k, \quad k\in\mathcal{K},\ p\in\mathcal{P} \tag{5.11}$$

ensures that a vehicle is considered utilised during any specific decision period if it services at least one customer during that period. As a mere convention for a valid model formulation, the constraint

$$s_0 = 0, \tag{5.12}$$

is also imposed. The domain constraints

$$\begin{aligned} x^p_{ijk} &\in \{0,1\}, \quad k\in\mathcal{K},\ (i,j)\in\mathcal{A},\ p\in\mathcal{P}, && (5.13)\\ y^p_k &\in \{0,1\}, \quad k\in\mathcal{K},\ p\in\mathcal{P}, && (5.14)\\ T^p_{ik} &\ge 0, \quad k\in\mathcal{K},\ p\in\mathcal{P} && (5.15) \end{aligned}$$

are finally imposed to enforce variable nature (binary and real-valued).

5.1.4 Objective function

During the creation of delivery schedules, the model objectives are to

$$\text{minimise} \quad \sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}} C_k y_k^p + \sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}} c_{ijk}x_{ijk}^p + \sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}} \beta_k(T_{n+1,k}^p - T_{0,k}^p) \tag{5.16}$$

and to

$$\text{maximise} \quad \frac{\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}} d_{ij}x_{ijk}^p m_{ij}}{\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}} d_{ij}x_{ijk}^p}. \tag{5.17}$$

The objective function in (5.16) represents the total transportation cost associated with the delivery schedule over the entire planning horizon. In the first term in (5.16), the fixed cost C_k associated with utilising each vehicle is added to the objective function value if the vehicle is utilised during any decision period (*i.e.* if $y_k^p = 1$), but no contribution is made if the vehicle is not utilised during the period (*i.e.* if $y_k^p = 0$). This ensures that the number of vehicles actually utilised is minimised. By summing across all elements in the matrices $\boldsymbol{X}^p$ for all $p \in \mathcal{P}$, the second term in the function in (5.16) captures the cost of each vehicle's route. If vehicle k travels from customer i to customer j during decision period p (*i.e.* if $x_{ijk}^p = 1$), then the associated cost, denoted by c_{ijk}, contributes to the objective function value; otherwise no contribution is made (*i.e.* if $x_{ijkp}^p = 0$). In the third term of the function in (5.16), the trip duration of vehicle k during decision period $p \in \mathcal{P}$ is calculated as $T_{n+1,k}^p - T_{0,k}^p$ and multiplied by the cost coefficient β_k. This ensures that the trip duration of each vehicle is minimised by scheduling the start of service at each customer as early as possible and ensuring that a vehicle k which does not visit customer i during decision period $p \in \mathcal{P}$ will have a service start time of $T_{ik}^p = 0$ at that customer.

The conflicting objective function in (5.17) represents the proportion of the total distance travelled by all delivery vehicles along master route arcs over the entire planning horizon. The denominator of the fraction in the objective function in (5.17) is the total distance travelled by all delivery vehicles over all decision periods, while the numerator of this fraction is the total distance travelled along the master route arcs by all delivery vehicles over all decision periods.

5.1.5 Symmetry-breaking constraints

Symmetry breaking constraints of the form

$$y_k^p \;\geq\; y_{k+1}^p, \quad k = \sum_{\ell\in\{1,\dots,j-1\}} K_\ell + 1, \dots, \sum_{\ell\in\{1,\dots,j\}} K_\ell - 1, \quad j \in \{0,\dots,|\mathcal{L}|-1\},\; p \in \mathcal{P} \tag{5.18}$$

may additionally be imposed so as to speed up any exact solution duration of the model.

5.2 Exact solution approach

The model derived in §5.1 was also implemented in CPLEX *via* its Python programming language interface for verification purposes. Due to the limited functionality of CPLEX when solving bi-objective optimisation problems, an approach was adopted in which a subset of the Pareto set is generated by implementing the familiarity objective function in (5.17) as an additional constraint in the model (5.1)–(5.16) and iteratively optimising the cost objective function for different minimum values of the familiarity objective function.

The code for this implementation in the Python interface of CPLEX is shown in Figure 5.1. The first two sets of decision variables, $\boldsymbol{X}$ and $\boldsymbol{Y}$, are defined as binary variables and are therefore already constrained to the set $\{0, 1\}$ as a result enforcing constraint sets (5.13) and (5.14), respectively. The decision variables captured in the set $\boldsymbol{T}$ are defined as continuous variables with a lower bound of zero, thereby imposing constraint set (5.15). Furthermore, the objective function in (5.17) was imposed as an additional constraint in the model implementation by limiting the objective function value to be at least as large as a specified value, called *familiarity*. Furthermore, the variable name *vehicles* represents the set $\mathcal{K}$, whereas the variable name $K[\ell]$ represents the parameter K_ℓ.

```python
# Decision variables
x = mdl.binary_var_dict(keysx, name = 'x')
y = mdl.binary_var_dict(keysy, name = 'y')
T = mdl.continuous_var_dict(keysT, lb = 0, name = 'T')

# Objective function in (6.16):
mdl.minimize(mdl.sum(C[k]*y[k,p] for k in vehicles for p in P) + mdl.sum(c[i,j,k]*x[i,j,k,p] for (i,j) in A for k in vehicles for p in P)
             + mdl.sum(beta[k]*(T[n+1,k,p]-T[0,k,p]) for k in vehicles for p in P))

# Constraint set in (6.1):
mdl.add_constraints(mdl.sum(x[i,j,k,p] for j in delta_plus[i] for k in vehicles for p in P) == F[i] for i in V_)
# Constraint set in (6.2):
mdl.add_constraints(mdl.sum(x[i,j,k,p] for j in delta_plus[i] for k in vehicles <= 1 for i in V_ for p in P)
# Constraint set in (6.3):
mdl.add_constraints(mdl.sum(x[0,j,k,p] for j in delta_plus[0]) == y[k,p] for k in vehicles for p in P)
# Constraint set in (6.4):
mdl.add_constraints(mdl.sum(x[i,j,k,p] for i in delta_minus[j]) - mdl.sum(x[j,i,k,p] for i in delta_plus[j]) == 0
            for j in V_ for k in vehicles for p in P)
# Constraint set in (6.5):
mdl.add_constraints(mdl.sum(x[i,n+1,k,p] for i in delta_minus[n+1]) == y[k,p] for k in vehicles for p in P)
# Constraint set in (6.6):
mdl.add_constraints(T[i,k,p] + s[i] + t[i,j,k] - T[j,k,p] <= M[i,j,k,p]*(1-x[i,j,k,p]) for k in vehicles for (i,j) in A for p in P)
# Constraint set in (6.7):
mdl.add_constraints(a[i]*mdl.sum(x[i,j,k,p] for j in delta_plus[i]) <= T[i,k,p] for k in vehicles for i in V if i!=n+1 for p in P)
mdl.add_constraints(T[i,k,p] <= b[i]*mdl.sum(x[i,j,k,p] for j in delta_plus[i]) for k in vehicles for i in V if i!=n+1 for p in P)
# Constraint set in (6.8):
mdl.add_constraints(a[n+1]*mdl.sum(x[i,n+1,k,p] for i in delta_minus[n+1]) <= T[n+1,k,p] for k in vehicles for p in P)
mdl.add_constraints(T[n+1,k,p] <= b[n+1]*mdl.sum(x[j,n+1,k,p] for j in delta_minus[n+1]) for k in vehicles for p in P)
# Constraint set in (6.9):
mdl.add_constraints(mdl.sum(q[i]/f[i]*mdl.sum(x[i,j,k,p] for j in delta_plus[i]) for i in V_ if f[i]>0) <= Q[k]*y[k,p]
            for k in vehicles for p in P)
# Constraint set in (6.10):
mdl.add_constraints(mdl.sum(x[i,j,k,p] for j in delta_plus[i]) <= g[i,k] for i in V_ for k in vehicles for p in P)
# Constraint set in (6.11):
mdl.add_constraints(mdl.sum(x[i,j,k,p] for (i,j) in A) <= (n+1)*y[k,p] for k in vehicles for p in P)
# Familiarity objective function in (6.17) as constraint:
mdl.add_constraint(mdl.sum(m[i,j]*x[i,j,k,p]*d[i,j] for (i,j) in A for k in vehicles for p in P) >= mdl.sum(x[i,j,k,p]*d[i,j]
            for (i,j) in A for k in vehicles for p in P)*familiarity)
# Symmetry-breaking constraints in (6.18):
for p in P:
    for j in range(0,len(L)):
        mdl.add_constraints(y[k,p]>=y[k+1,p] for k in range(sum(K[l] for l in range(0,j))+1, sum(K[l] for l in range(0,j+1))))
```

FIGURE 5.1: *The code for the implementation of the model in §5.1 in the Python interface of CPLEX.*

A pseudo-code description of the exact solution approach adopted is described in Algorithm 5.1. The function Solve2() returns an optimal solution and corresponding familiarity objective function value when first maximising familiarity, fixing the familiarity objective function value returned, and then minimising cost. The function Solve1(i,$S[j]$) returns an optimal solution and corresponding familiarity objective function value when minimising cost as well as specifying a minimum familiarity value of i and providing the starting solution $S[j]$ as a warm start to CPLEX. Let P denote the number of portions into which the range of the familiarity objective function is partitioned in order to find a subset of the Pareto set across the entire range. Furthermore, let $S[t]$ denote the solution returned during the t^{th} iteration and let $F[t]$ denote the familiarity objective function value associated with solution $S[t]$.

The exact solution approach starts by returning a solution corresponding to the maximum possible familiarity objective function value (Line 1). The solution corresponding to the minimal cost, regardless of familiarity, is returned next by providing the solution returned in Line 1 as a starting solution, as well as specifying a minimum familiarity value of zero (Line 2). The two solutions, $S[0]$ and $S[P]$, therefore represent the solutions at the extreme points of the Pareto-optimal set. Furthermore, the minimum and maximum familiarity objective function

Algorithm 5.1: Exact solution approach for the model (5.1)–(5.17)

```
1 $S[P]$, $F[P] \leftarrow$ Solve2()
2 $S[0]$, $F[0] \leftarrow$ Solve1(0, $S[P]$)
3 $I = -(F[P] - F[0])/P$
4 for $i = 1$ to $P - 1$ do
5     $S[P-i]$, $F[P-i] \leftarrow$ Solve1($F[P] + i \times I$, $S[P-i+1]$)
```

values of solutions in the Pareto set correspond to $F[0]$ and $F[P]$, respectively. In the remainder of the algorithm, minimum familiarity values ranging from $F[P]$ to $F[0]$ in increments of $I = -(F[P] - F[0])/P$ are specified and the solutions returned as well as their corresponding familiarity objective function values are stored in $\boldsymbol{S}$ and $\boldsymbol{F}$, respectively. Furthermore, since each solution $S[i]$ is a feasible solution when solving for $S[j]$, where $i > j$ (since the minimum familiarity is already adhered to), a starting solution $S[i+1]$ is provided as a warm start when solving for solution $S[i]$ in order to speed up the run time of CPLEX during each iteration. The output of the algorithm is a subset $\boldsymbol{S}$ of the Pareto set for the problem instance.

In the remainder of this section, a small example problem instance containing ten customers is described in order to illustrate the input data and verify the output data of an instance of the model (5.1)–(5.17). This small problem instance corresponds to that solved in §4.2, with the addition of a planning horizon consisting of $|\mathcal{P}| = 2$ decision periods and a maximum storage capacity e_i associated with each customer $i \in \mathcal{V}$. Furthermore, a different demand volume q_i, service duration s_i, and number of required visits F_i is associated with each customer $i \in \mathcal{V}$. The demand volume associated with each customer is normally distributed from the average demand volume of the corresponding customer used to compute the master routes, with a standard deviation $\sigma = 5$. The number of required visits F_i of each customer $i \in \mathcal{V}$ was then calculated as described in §5.1.1. The information associated with each vertex in this problem instance is presented in Table 5.1, whereas information about the types of delivery vehicles to which each delivery vehicle may belong are presented in Table 4.2. Two of each type of delivery vehicle were made available to service customers during each decision period, therefore resulting in the same information pertaining to the fleet of delivery vehicles available as in Table 4.3. Furthermore, the set of master route arcs is based on the solution to the problem instance solved in §4.2 and illustrated graphically in Figure 4.2. If an arc (i, j) forms part of the set of master arcs, the arc (j, i) in the opposite direction is also considered to form part of the set of master route arcs.

The small problem instance was solved by invoking the aforementioned exact solution approach to generate thirty solutions. Among the thirty solutions generated, there were nine distinct solutions which are plotted in objective space in Figure 5.2. The familiarity objective function value of the two solutions corresponding to the extreme points of the Pareto-optimal set were 0.7582 and 0.9229, whereas the cost objective function values of these two solutions were R14 026.94 and R27 464.57, respectively. It is clear that a trade-off exists between the transportation cost and the driver-route familiarity of solutions. Furthermore, relatively large sparse segments are observed in terms of the cost objective function along the frontier of non-dominated solutions returned, but not in terms of the familiarity objective function. This phenomenon may be ascribed to the fixed costs associated with utilising delivery vehicles. When an increased minimum familiarity objective function value is specified, it may be required to utilise an additional or more expensive delivery vehicle, resulting in a significant increase in transportation cost and a sparse segment in the cost objective function values of solutions returned.

The solution indicated in red in Figure 5.2 is illustrated graphically in Figure 5.3. Once again, the depot is represented by a black square, whereas each customer is represented by either a yellow or

Table 5.1: *The input data related to vertices in an illustrative example problem instance. The horizontal axis and vertical axis coordinates of customers are denoted by ζ_i and η_i, respectively, and are measured in kilometres. The number of visits required by each customer is denoted by F_i. Furthermore, the demand volume in cubic metres associated with each customer is denoted by q_i, and the maximum volume of demand in cubic metres allowed to be delivered to each customer during a single period is denoted by e_i. Furthermore, the service duration in minutes at each customer is denoted by s_i. Finally, the time-window start time and end time associated with each customer is denoted by a_i and b_i, respectively, and is measured in minutes from the start of the day.*

Vertex, i	ζ_i	η_i	F_i	q_i	e_i	s_i	a_i	b_i
0	50	50	–	–	–	0	0	600
1	60.28	54.49	2	22.74	16.98	44.10	0	270
2	42.37	64.59	2	14.75	10.60	32.12	0	270
3	43.76	89.18	1	8.54	16.67	35.63	270	540
4	96.37	38.34	1	11.03	16.71	43.10	0	270
5	79.17	52.89	2	15.76	12.10	33.64	270	540
6	56.80	92.56	1	7.94	11.29	33.82	270	540
7	7.10	8.71	2	23.94	13.15	45.91	270	540
8	2.02	83.26	1	10.34	13.64	41.02	270	540
9	77.82	87.00	1	14.24	15.70	52.72	0	270
10	97.86	79.92	2	16.50	14.39	34.74	0	270

a green vertex (in the case of a morning or afternoon time-window, respectively). Furthermore, arcs traversed that form part of the set of master route arcs are represented by dotted lines. This solution corresponds to a cost objective function value of R14 912.23, comprising a fixed cost of R10 250, a variable cost of R4 522.07, and a trip duration penalty cost of R140.16, as well as a familiarity objective function value of 0.8970. In this solution, a single delivery vehicle (of type $\ell = 3$) is utilised during decision period $p = 1$ and two delivery vehicles (of types $\ell = 2$ and $\ell = 3$) are utilised during decision period $p = 2$.

The route of each delivery vehicle utilised during each decision period, the volume of commodities delivered to each customer, the total volume of commodities delivered by each delivery vehicle, and the service start times at customers are summarised in Table 5.2. By inspecting these output data, it may be confirmed that the solution retured by CPLEX satisfies all of the constraints in the model and is therefore feasible. Each delivery vehicle utilised during each period departs from the depot and returns to the depot after having serviced the customers along its route. Each customer is also visited the specified number of times throughout the planning horizon, yet no more than once during any decision period, and service at each customer starts during its specified time-window. Furthermore, each customer's demand for the entire planning horizon is satisfied and the capacity of no delivery vehicle utilised is exceeded. The overall utilisation of the capacity of those delivery vehicles used is 76.72%.

5.3 Model verification

In order to ensure that the model derivation in §5.1 and its implementation in §5.2 are correct, further verification was performed. This involved solving the model for thirty randomly generated instances of the model (5.1)–(5.17) by obtaining five Pareto-optimal solutions for each instance and ensuring that feasible solutions were returned by CPLEX for each of the hundred

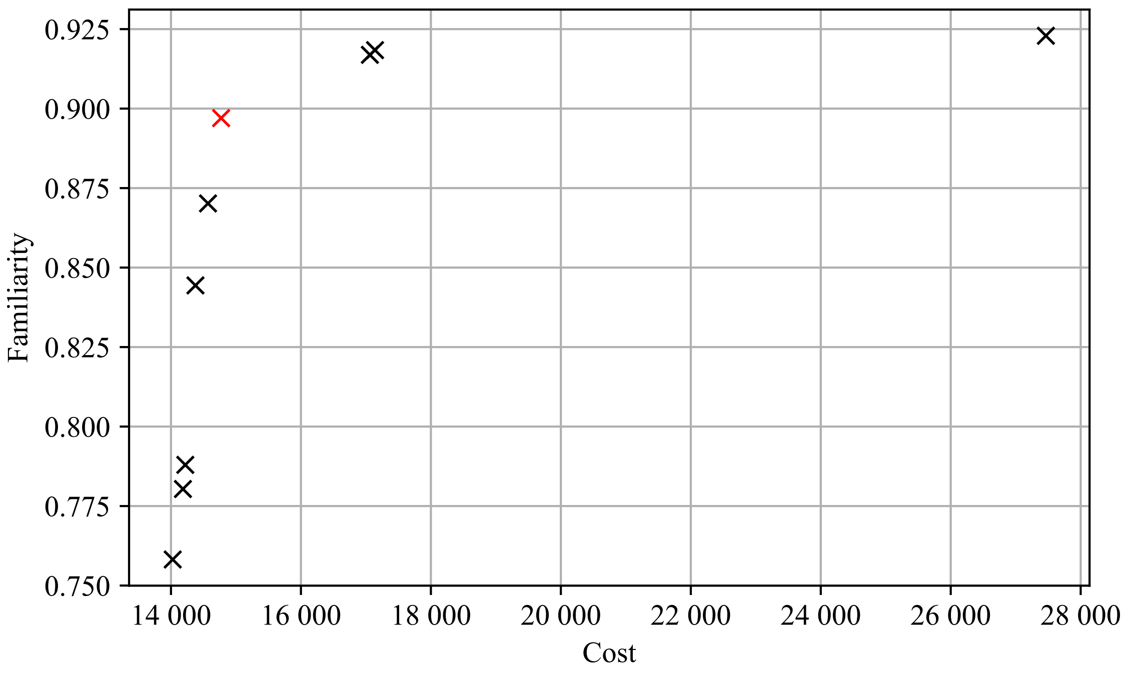


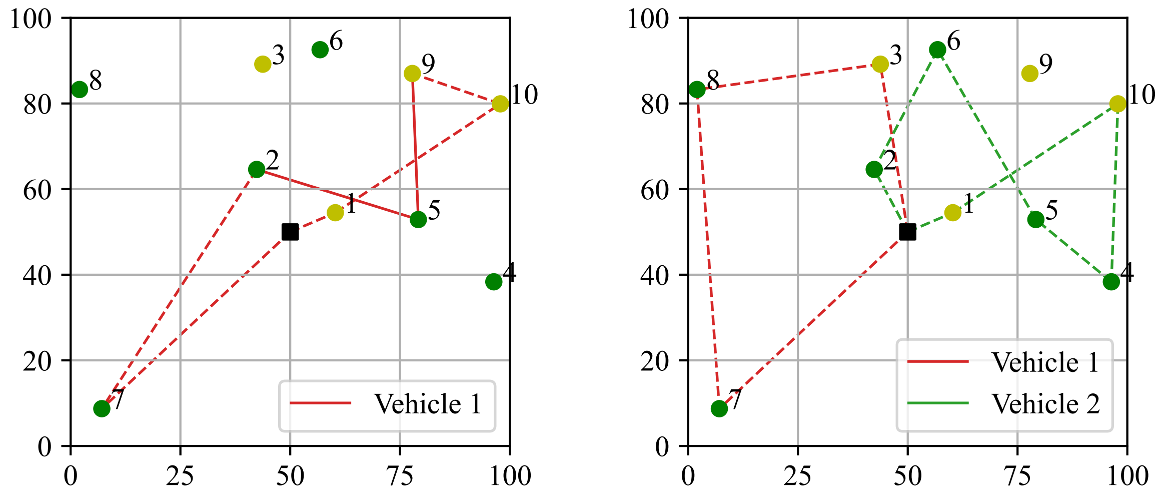
FIGURE 5.2: *A subset of the Pareto set returned by the exact solution approach for the model (5.1)–(5.17) of the operational phase of the FVRP for a small example problem instance.*

and fifty solutions, similar to the procedure followed for the small, illustrative example problem instance solved in §5.2.

In order to generate reproducible synthetic problem instances of the model (5.1)–(5.17) for the operational phase of the FVRP, a problem instance generator was once again created in the programming language Python, as was done in §4.3. This problem instance generator may be downloaded from the author's Gitthub repository[1]. The synthetic problem instance generator for the operational phase takes twelve parameters as input. Once again, the input parameters w_c, w_d, q_{min}, q_{max}, ϕ, ψ, b_{max} and $seed_1$ are taken as input in order to generate the same locations in the transportation network, as well as the same average demand volumes, service duration times, and time-windows associated with each customer for the corresponding $seed_1$ value. Two new input parameters, denoted by $seed_2$ and σ, are used to update the average demand volume q_i associated with each customer $i \in \mathcal{V}'$ over the planning horizon with the actual demand volumes for the planning horizon. This is achieved for each customer by randomly assigning its value according to a normal distribution with the average demand volume as mean and σ as standard deviation. By specifying different values for $seed_2$, an arbitrary number of actual demand volumes may therefore be generated for a transportation network and corresponding set of master routes, in order to simulate multiple future planning horizons for which actual delivery routes may be computed. The number of required visits F_i associated with each customer i in $\mathcal{V}'$ is calculated as $F_i = \lceil q_i/e_i \rceil$. Finally, the number of decision periods in the planning horizon, denoted by p, is also taken as input.

The output of the problem instance generator for the operational phase is also a *Problem_Instance* class consisting of multiple Python dictionaries, one for each input parameter to the model (5.1)–(5.17) for the operational phase of the FVRP. The standard input parameters of the problem

[1]Gitthub repository for producing problem instances for the operational phase of the FVRP: https://github.com/JacobusKing/Problem-Instance-Generator-for-Phase-2.git



(a) Period 1 (b) Period 2

Figure 5.3: *A Pareto-optimal solution to a small problem instance of the model (5.1)–(5.17) for the operational phase of the FVRP. Arcs traversed that also form part of the set of master route arcs are indicated using dotted lines.*

Table 5.2: *The output data related to the solution to a small problem instance of the model for the operational phase.*

Decision period, p	Vehicle, k	From i	To j	q_j/f_j	T^p_{jk}
1	5	Depot	1	11.37	17.495
		1	10	8.25	116.054
		10	9	14.24	176.311
		9	5	7.88	270.000
		5	2	7.37	349.987
		2	7	11.97	461.394
		7	Depot	–	578.753
Total				61.08	
2	3	Depot	3	8.54	190.323
		3	8	10.34	270.000
		8	7	11.97	389.106
		7	Depot	–	497.237
Total				30.85	
2	5	Depot	1	11.37	86.779
		1	10	8.25	185.338
		10	4	11.03	270.000
		4	5	7.88	340.128
		5	6	7.94	428.419
		6	2	7.37	500.009
		2	Depot	–	551.888
Total				53.84	

instance generator (used if no value is specified for a parameter) are $w_c = 100$, $w_d = 0$, $q_{min} = 10$, $q_{max} = 20$, $\phi = 3$, $\psi = 10$, $b_{max} = 540$, and $\sigma = 5$. The only compulsory input parameters required for the generation of a problem instance is therefore the number of customers in each problem instance n, the number of decision periods in the planning horizon p, and the seed numbers, $seed_1$ and $seed_2$. The small problem instance solved in §5.2 can be generated by providing the parameters $n = 10$, $p = 2$, $seed_1 = 1$, and $seed_2 = 1$ to the problem instance generator for the operational phase.

The thirty problem instances solved in pursuit of further verifying the model (5.1)–(5.17) for the operational phase of the FVRP in §5.1 and its implementation in §5.2 were generated by providing all combinations of the input parameters $n = 8, \ldots, 13$, $p = 2$, and seed numbers $seed_1 = 1, \ldots, 5$ and $seed_2 = 1$ to the problem instance generator. The information associated with the vehicle types to which each of the available delivery vehicles may belong is the same as the information presented in Table 4.2, and five delivery vehicles of each type were made available to service customers during each period. Furthermore, five Pareto-optimal solutions were computed for each of the thirty problem instances according to the method described in Algorithm 5.1, resulting in a total of one hundred and fifty solutions returned. The synthetic problem instances were again solved on a computer with an Intel Core i7 CPU operating at 2.90GHz with 16GB of memory. Furthermore, a run time limit of eight hours was imposed for each solution computed to any problem instance, resulting in a maximum possible run time duration of forty hours per problem instance.

For each of the aforementioned solutions, the output returned by CPLEX was inspected and compared with the input parameters of the problem instance in order to verify the feasibility of solutions. The following aspects of the output data were inspected to ensure feasibility:

- Each delivery vehicle utilised during any decision period must be assigned a route that departs from the depot and returns to the depot after having serviced the customers along its route during that decision period.
- The sum of all delivery quantities assigned to each delivery vehicle utilised during any decision period may not exceed the capacity associated with the delivery vehicle.
- The sum of all delivery quantities received by each customer throughout the planning horizon must equal the total quantity of demand associated with the customer.
- Customers may only be assigned to delivery vehicles that are considered utilised and of a type compatible with visiting the customer.
- The service start time at each customer must be within its specified time-window.
- The service start time at each customer must be later than the service start time at the previous customer along the route, or the departure time at the depot if it is the first customer along the route.
- Each customer must be visited the specified number of times throughout the planning horizon.
- No customer may be visited more than once during any decision period.

The final solution returned by CPLEX for each of the hundred and fifty solutions were inspected and the above requirements were indeed satisfied in each case. The model derived in §5.1 and its implementation in §5.2 have therefore been verified empirically.

5.4 Time complexity of the exact model solution approach

In order to determine the relationship between the number of customers in a randomly generated instance of the model (5.1)–(5.17) and the run time required by CPLEX to return Pareto-optimal solutions to the corresponding problem instance, the run times and optimality gaps returned by CPLEX for each of the hundred and fifty solutions generated in §5.3 were analysed. The run time of the exact solution approach for the model (5.1)–(5.17) of the operational phase of the FVRP for each solution generated in each problem instance is plotted on a logarithmic scale in Figure 5.4. The remaining optimality gaps associated with generating each solution are plotted in Figure 5.5. Furthermore, the total run times associated with the thirty problem instances (summing the run times of the five solutions returned for each instance), are plotted on a logarithmic scale in Figure 5.6.

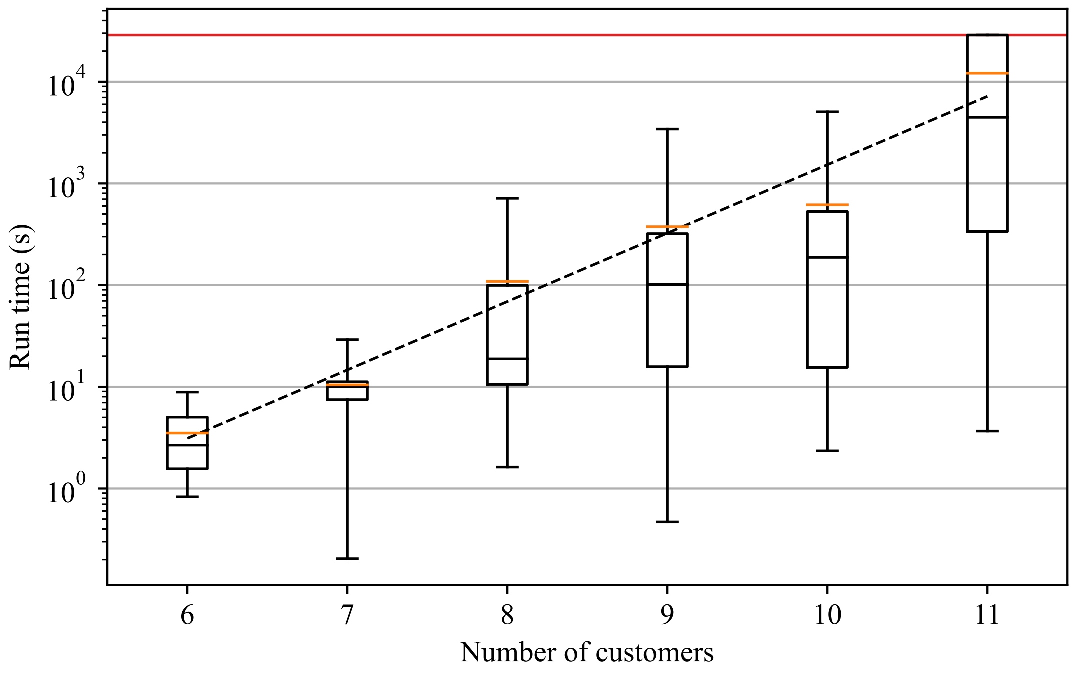


FIGURE 5.4: *The run times recorded when generating a single Pareto-optimal solution to synthetic instances of the model (5.1)–(5.17), involving different numbers of customers, during the operational phase of the FVRP. The red horizontal line represents the eight hour run time limit specified when generating each Pareto-optimal solution to each problem instance. The mean trend of the run times is indicated by means of a dotted line.*

Yet again, despite being plotted on a logarithmic scale, the aforementioned run times associated with generating each solution seem to increase with a strong linear trend as the number of customers in the problem instance increases. This clearly demonstrates that the computational complexity of the model increases exponentially as the number of customers in the problem instances increase for a fixed number of available delivery vehicles. Furthermore, seven of the solutions generated for instances containing eleven customers could not be solved to optimality due to CPLEX not being able to return proven optimal solutions within the eight hour time limit, resulting in the optimality gaps observed in Figure 5.5. Upon analysis of the total run times recorded for generating five solutions to each problem instance, an even larger slope of increase is observed, as expected. As in the case of the strategic phase of the FVRP, the exact solution approach for the model of the operational phase of the FVRP would therefore not seem

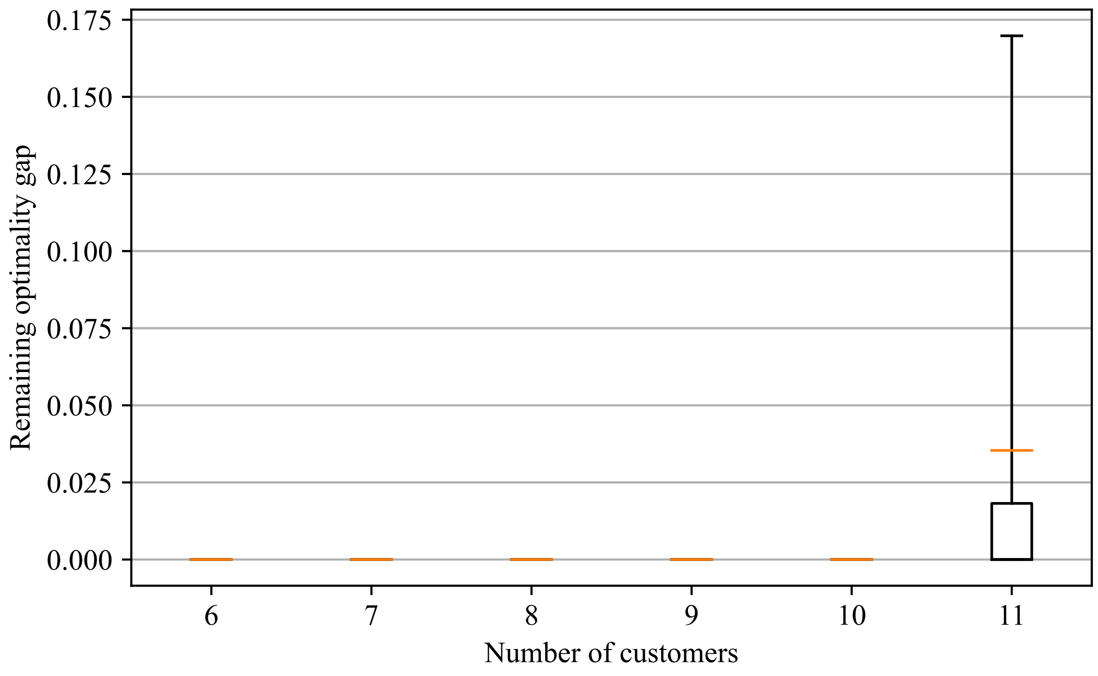


FIGURE 5.5: *The remaining optimality gaps recorded when generating Pareto-optimal solutions to synthetic instances of the model (5.1)–(5.17) involving different numbers of customers during the operational phase of the FVRP.*

to be scalable to the size of real-world problem instances in which it may be required to generate more non-dominated solutions to problem instances, typically containing hundreds of customers. This motivates the need for an approximate solution approach in which many non-dominated high-quality, but not necessarily Pareto-optimal, solutions may be achieved for large problem instances within an acceptable time-frame.

5.5 Approximate model solution approach

An approximate solution approach is proposed in this section for the model (5.1)–(5.17) pertaining to the operational phase of the FVRP. As in the case of the approximate solution approach proposed for the model (4.1)–(4.18) pertaining to the strategic phase of the FVRP, this solution approach is based on the second version of the HGSADC algorithm proposed by Vidal *et al.* [140], with some adaptions made to the structure and the working of the algorithm in order to be able to accommodate instances of the model (5.1)–(5.17). Furthermore, in order for the approximate solution approach to be able to solve bi-objective optimisation model instances, the non-dominated sorting procedure proposed by Srinivas and Deb [124] and the CDD measure proposed by Deb *et al.* [38] are employed when evaluating solutions.

A pseudo-code description of the proposed solution approach is presented in §5.5.1, and this is followed by a description of the adaptions made to the original HGSADC algorithm. Adaptions made to the trajectory-based algorithmic component aimed at accommodating bi-objective problems are discussed in §5.5.2. Furthermore, an additional set of penalty parameters required to allow for the infeasibility of solutions are described in §5.5.3. Thereafter, the non-dominated sorting of solutions and the application of the CDD measure during the execution of the algorithm are explained in §5.5.4. An additional step performed during the initialisation of the

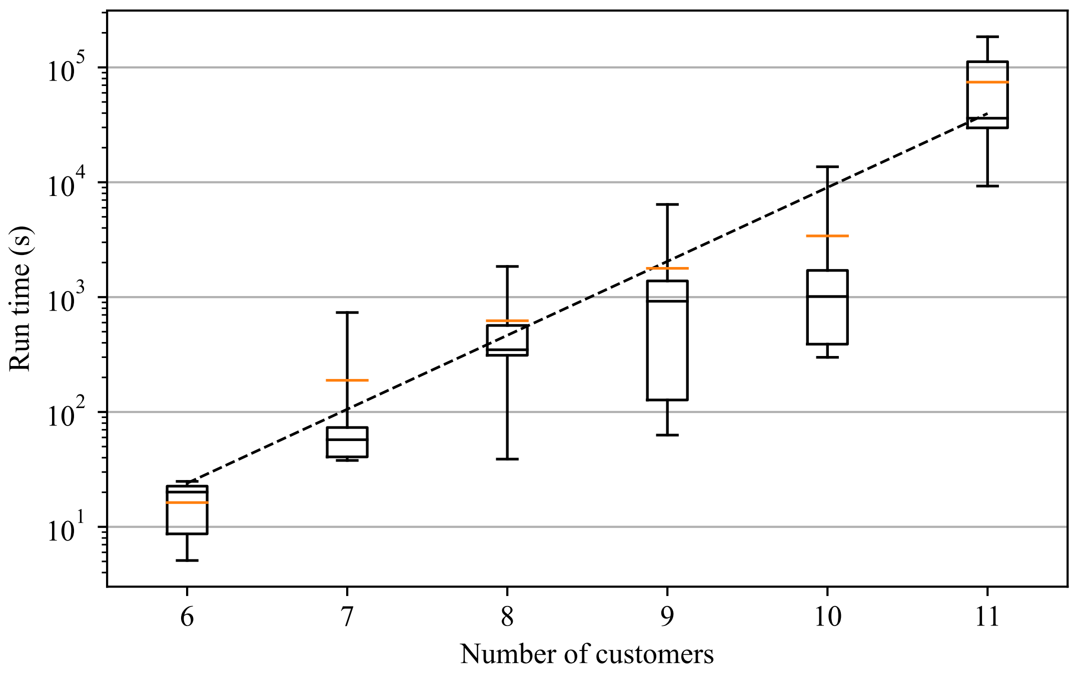


FIGURE 5.6: *The run times recorded when generating five single Pareto-optimal solutions to synthetic instances of the model (5.1)–(5.17), involving different numbers of customers, during the operational phase of the FVRP. The mean trend of the run times is indicated by means of a dotted line.*

population, where the master routes computed during the strategic phase of the FVRP are embedded into a solution and included in the population, is discussed in §5.5.5. The addition of a heterogeneous fleet attribute is described next in §5.5.6, and this is followed by a discussion in §5.5.7 on the manner in which the service start times of delivery vehicles at customers were determined. Finally, the parameter values adopted during the implementation of the approximate solution approach are described in §5.5.8, whereas the classes and functions created during the implementation of the approximate solution approach in the programming language Python are discussed in §5.5.9 and §5.5.10, respectively.

5.5.1 Pseudo-code description

The pseudo-code description of the proposed solution approach is the same as that for the second version of the HGSADC algorithm provided in Algorithm 3.4. The proposed solution approach begins by initialising a population consisting of a feasible subpopulation and an infeasible subpopulation, by randomly generating individuals representing "candidate" solutions, as well as constructing a solution based on the master routes computed during the strategic phase to be included in the population. The main loop of the algorithm is then executed over It_{NI} non-improving iterations or for a maximum duration of T_{max} minutes, after which the set of non-dominated solutions found during the search is returned. During each iteration, two parent solutions, P_1 and P_2, are selected according to a binary tournament selection scheme from the union of the feasible and infeasible subpopulations. An offspring solution C is created from the two parent solutions by invoking the PIX crossover operator, after which the offspring solution undergoes education during which two local search procedures are performed. If the offspring solution is not feasible, it is placed in the infeasible subpopulation and undergoes repair with probability P_{rep}, during which education is performed with increased penalty weights for infeasibility. If, however, the offspring solution generated is feasible, it placed in the feasible subpopulation.

Both subpopulations are managed to contain between μ and $\mu + \lambda$ individuals. If any subpopulation contains more than $\mu + \lambda$ individuals, survivor selection takes place during which λ individuals are removed from the corresponding subpopulation. In order to explore a larger area of the search space, diversification takes place every It_{div} non-improving iterations, during which $\mu/3$ individuals are retained in each subpopulation and 4μ new individuals are generated randomly and introduced into the population in the same manner as when the population was initialised. Furthermore, in order to solve large problem instances effectively, decomposition occurs every It_{dec} iterations. During decomposition, the problem instance is decomposed into smaller subproblem instances and each subproblem instance is solved approximately after which they are reconstituted once again to obtain complete solutions which are inserted into the appropriate subpopulations. Finally, upon termination of the algorithm, the set of non-dominated solutions found during the search is returned.

5.5.2 Alternating objectives during local search procedures

During the education and repair procedures, two local search procedures are performed in order to improve the offspring solution generated. When executing these procedures, however, solutions are only improved in terms of a single objective. In order to approximate Pareto-optimal solutions to instances to the bi-objective model (5.1)–(5.17) of the operational phase of the FVRP, an approach was adopted in which the objective function to be optimised during the education and repair procedures is alternated. Each one hundred iterations, when the penalty parameters for infeasibility are updated, the objective function to be optimised during these procedures is alternated. This ensures that the approximate Pareto front is improved in both directions of increased familiarity and decreased cost in the objective space.

5.5.3 The evaluation of solutions

Recall from §3.5.2 that in the original HGSADC algorithm solutions are allowed to be infeasible in terms of their load, duration, and time-warp, by penalising these infeasibilities with the appropriate penalty weights (ω^D, ω^Q, and ω^{TW}, respectively) when calculating the penalised cost of a solution. In the proposed approximate solution approach, solutions are evaluated based on their penalised cost (representing their cost objective function values to which penalised infeasibilities are added) as well as their *penalised familiarity* (representing their familiarity objective function values to which penalised infeasibilities are added). Since the range of the familiarity objective function values differs from that of the cost objective function values, separate sets of penalty weights are considered when calculating the penalised cost and penalised familiarity of solutions. Denote the set of penalty weights considered when penalising infeasible load, duration, and time-warp in the penalised cost of solutions by ω^D_{cost}, ω^Q_{cost}, and ω^{TW}_{cost}, respectively. Furthermore, denote the set of penalty weights considered when penalising the aforementioned infeasibilities in the penalised familiarity of solutions by ω^D_{fam}, ω^Q_{fam}, and ω^{TW}_{fam}, respectively. These two sets of penalty weights are changed dynamically during execution of the approximate solution approach, yet independently of one another. When updating the penalty parameters each one hundred iterations, only the set of penalty parameters associated with the objective function which was optimised during the education and repair procedures is updated. This ensures that the target portion of naturally feasible individuals can be achieved when optimising either the cost objective function or the familiarity objective function during the education and repair procedures.

5.5.4 The non-dominating sorting of solutions

In order to direct the search towards approximately Pareto-optimal solutions, a combination of the non-dominated sorting procedure, proposed by Srinivas and Deb [124] and discussed in §3.6, and the CDD measure, proposed by Deb *et al.* [38] and also reviewed in §3.6, are employed. In the event of a new solution being inserted into the population, a rank is assigned to each solution (according to the non-dominated sorting procedure) and a CDD measure is calculated for each solution (based on the non-dominated front in which the solution resides). These measures are based on the union of both the feasible and infeasible subpopulations and are therefore based on the penalised cost and penalised familiarity of solutions, discussed in §5.5.3. When, for example, comparing two solutions, the solution residing within the better ranked non-dominated front is preferred. If both solutions reside in the same non-dominated front, the solution achieving the larger CDD value is preferred.

This combination of the non-dominated sorting procedure and the CDD measure is employed during the parent selection, survivor selection, diversification, and decomposition procedures. During the decomposition procedure, the solution selected to determine which customers are to be included in each subproblem is selected as the feasible solution corresponding to the best non-dominated front and subsequent CDD measure. In the case of the feasible subpopulation being empty, the infeasible solution corresponding to the best non-dominated front and subsequent CDD measure is selected. Furthermore, when reconstituting the solutions returned for each subproblem, the feasible solutions in each subproblem corresponding to the best non-dominated front and the best objective function value (for the objective function being optimised at the time) within this front are selected to reconstitute solutions to be inserted into the population.

5.5.5 Initialising the master routes as an additional solution

When initialising the population of solutions at the start of the approximate solution approach, the master routes computed during the strategic phase of the FVRP are configured into a solution and inserted into the population. This solution is expected to achieve a relatively large familiarity objective function value at an acceptable cost and therefore directs the search towards the approximate Pareto front from the start of the execution of the approximate solution approach. First, routes forming part of the set of master routes are assigned in a random order to decision periods in such a manner that no customer is visited more once during any decision period. It may not be possible to assign a route to a decision period not containing any of the customers being visited along the route, in which case the route is not included in the solution. Once all routes have been considered, the number of visits received by each customer throughout the planning horizon is determined after which each customer is considered in random order. If a customer considered is visited more often than required, the cost savings associated with removing the customer from each of the decision periods in which it is visited is determined, and the removals associated with the largest cost saving are performed until the customer receives the correct number of visits throughout the planning horizon. If, on the other hand, a customer under consideration receives fewer visits than required, the cost of the best possible insertion of that customer into any period during which the customer is not yet visited is determined, and the insertions associated with the lowest cost are performed until the customer is visited the required number of times. The solution thus constructed is then inserted as an additional solution in the appropriate subpopulation.

5.5.6 Heterogeneous fleet attribute

Whereas the original HGSADC algorithm is capable of solving VRP instances containing multiple depots, delivery routes are computed for a single depot and its assigned customers in the model (5.1)–(5.17) of the operational phase of the FVRP. This model, however, makes provision for having a heterogeneous fleet of delivery vehicles stationed at the depot — an attribute not included in the original HGSADC algorithm. This difference in capability required by the model (5.1)–(5.17) is accommodated in the same manner as for the approximate solution approach pertaining to the model for the strategic phase described in §4.5.3, by replacing the depot chromosome according to which individuals are represented with a vehicle type chromosome.

Once again, instead of assigning each customer to a depot, customers are assigned to multiple vehicle types, and a giant tour is generated for each decision period and vehicle type combination (p, ℓ). This giant tour is then partitioned into multiple routes, each performed by a delivery vehicle of type ℓ during decision period p. When evaluating a route, the fixed cost associated with the delivery vehicle to which the route is assigned, is added to the penalised cost. Moreover, the distance measure incorporated into the penalised cost of a route is adapted to take into account the different costs associated with travelling between vertices by the different vehicle types.

When initialising the population, individuals are generated by randomly assigning each customer to a feasible visiting pattern, and then randomly adding the customer to the giant tour of any type of delivery vehicle within each of the decision periods associated with the assigned pattern. Furthermore, during any stage of the algorithm, a customer may only be assigned to a giant tour of a vehicle type and period combination (p, ℓ) if the customer is, in fact, allowed to be visited by a vehicle of type ℓ, thereby ensuring feasible vehicle-customer assignments.

5.5.7 The service start times at customers

The same approach adopted towards determining the service start times at customers for the approximate solution approach for the strategic phase of the FRP, described in §4.5.6, is also adopted in the approximate solution approach for the operational phase of the FVRP. This approach entails having each delivery vehicle depart from its current location and arrives at its next location as early as possible. Although this approach may result in large amounts of vehicle waiting time, it results in the minimum amount of time-warp. At termination of the approximate solution approach for the model (5.1)–(5.17) pertaining to the operational phase of the FVRP, before returning the set of non-dominated solutions found during the entire search, each non-dominated solution is provided as a warm start to CPLEX and is returned as soon as CPLEX recognises the warm start as a feasible solution. The optimal service start times for each non-dominated solution may then be deduced from the solutions returned by CPLEX, which are returned as the service start times for the set of non-dominated solutions by the approximate solution approach.

5.5.8 Parameters of the algorithm

Since the parameter values of the HGSADC algorithm do not depend on the type of VRP being solved, all parameters included in both the proposed approximate solution approach and those that were in the original HGSADC algorithm are set to recommended values based on the parameter calibration performed by Vidal *et al.* [139, 140] and described in §3.5.4. The only parameter value which would seem to depend on the type of VRP being solved is the generation size λ, which is fixed at the value proposed for the MDPVRP, since its structure is most similar to that of the FVRP.

5.5.9 Classes created during the implementation

The proposed approximate solution approach was implemented in the Python programming language, adopting an object oriented approach. The following classes were created during the implementation:

The Globals class stores global variables which should be accessible to all objects and functions in the implementation.

The ProblemInstance class stores the input parameters required in an instance of the model (5.1)–(5.17) for the operational phase of the FVRP.

The Settings class stores all the configuration parameters of the approximate solution approach.

The Fleet class stores information associated with the available fleet of delivery vehicles stationed at the depot.

The Individual class stores an individual (solution) in the population as well as all information associated with the individual.

The Population class stores the current population of individuals.

The PhaseII class stores the main function responsible for executing the approximate solution approach.

Variables stored in the Globals class include an object of the ProblemInstance class, an object of the Settings class, and an object of the Population class. This allows all functions and classes access to these variables without the need to pass information regularly between functions. The input parameters stored in the ProblemInstance class may either be provided manually or may else be generated by invoking the problem instance generator described in §5.3. Furthermore, an object of the Fleet class also forms part of the ProblemInstance class. An object of the ProblemInstance class cannot be created without specifying all input parameters required for an instance of the model (5.1)–(5.17), therefore serving the purpose of input data validation. When an object of the ProblemInstance class is created, the pre-processing of input parameters is also performed. Finally, the Globals class stores a list containing the set of non-dominated solutions found throughout the execution of the approximate solution approach.

The Individual class calculates and stores all information associated with an individual, such as the chromosomes and fitness values associated with the individual. This information is also continually updated throughout the execution of the approximate solution approach in order to store new fitness values after having updated penalty parameters or after operations, such as

Education, have been performed on the individual. An object of the Population class stores the current feasible and infeasible subpopulations which, in turn, each consists of potentially many objects of the Individual class.

Finally, the PhaseII class stores the main function which executes the approximate solution approach by accessing and changing the variables stored in an object of the Globals class.

5.5.10 Functions created during the implementation

The main functions used during the execution of the approximate solution approach are as follows:

The Initialise function is a stand-alone function which generates random solutions as objects of the Individual class and stores them in an object of the Population class. This function also generates the solution based on the master routes as an object of the Individual class, which is also stored in the object of the Population class.

The ParentSelection function forms part of the Population class and returns two parents selected from the population, each an object of the Individual class, to participate in the crossover procedure.

The Crossover function is a stand-alone function which takes two individuals, each an object of the Individual class, as input to perform the PIX crossover, and returns the offspring thus generated as an object of the Individual class.

The Evaluate function forms part of the Individual class and is responsible for evaluating all fitness measures of an individual based on the three chromosomes representing it.

The EvaluatePopulation function forms part of the Population class and facilitates the calculation of all fitness measures for individuals in the population, as well as the non-dominated sorting procedure and assignment of CDD measures.

The Educate function forms part of the Individual class and is responsible for performing the local search procedure after which the updated chromosomes representing an individual are stored and the individual is re-evaluated.

The Diversify function forms part of the Population class and its purpose is to carry out the population diversification component of the approximate solution approach.

The SurvivorSelection function forms part of the Population class and carries out the survivor selection component of the approximate solution approach.

The AdjustPPs function is a stand-alone function which adjusts the penalty parameters based on the number of feasible offspring generated.

The Decompose function is a stand-alone function which is responsible for carrying out the decomposition component of the metaheuristic.

The Solve function forms part of the Phase_1 class and executes the metaheuristic.

While many smaller functions not mentioned above also appear in the Python implementation of the approximate solution approach, the above-mentioned functions are the main functions utilised to execute the approximate solution approach.

When the Solve function is called, the approximate solution approach is executed to solve an instance of the model (5.1)–(5.17) represented by the objects stored in the Globals class. The population of individuals, also stored in the Globals class, is continually updated and is accessible to all functions.

In the Decomposition function, the problem instance is decomposed into smaller subproblem instances and each subproblem instance is solved approximately by the proposed approximate solution approach. This is achieved by temporarily replacing the objects in the Globals class with objects representing the problem instance only containing those customers associated with the decomposed problem instance. Once each subproblem instance has been solved, the variables in the Globals class are replaced with the objects representing the original problem instance.

5.6 Systematic approximate solution approach verification

A verification of the approximate solution approach proposed for the operational phase of the FVRP discussed in §5.5 was performed in order to ensure its capability of returning high-quality feasible solutions. The thirty problem instances solved during the systematic model verification performed in §5.3 were once again solved, this time approximately. The non-dominated solutions returned by CPLEX for these thirty instances were taken as reference points to verify the implementation of the approximate solution approach. The run times and objective function values of the sets of non-dominated solutions found were recorded for each instance. Furthermore, a maximum run time of eight hours and a stopping criterion of five thousand non-improving iterations were specified for each instance.

The HV quality indicator discussed in §3.6 was employed to evaluate the quality of the sets of non-dominated solutions returned by the approximate solution approach relative to those returned by CPLEX. The reference point used to calculate the HV was chosen as 1.1 times the nadir point of each instance. Furthermore, normalisation was performed with respect to the minimum and maximum objective function values returned by any of the two solution approaches, in order for each objective to exhibit equal importance during the optimisation process. The average differences in HV values of the sets of non-dominated solutions returned by the exact solution approach and those returned by the approximate solution approach are summarised in Table 5.3. Furthermore, the average number of non-dominated solutions returned for each instance by CPLEX and by the approximate solution approach is also shown.

TABLE 5.3: *Mean percentage differences in HV measures, denoted by Δ HV, between the non-dominated solutions returned by the exact solution approach and those returned by the approximate solution approach for the model (5.1)–(5.17) of the operational phase of the FVRP. The average numbers of non-dominated solutions returned by CPLEX and the approximate solution approach are denoted by N_{exact} and $N_{approximate}$, respectively.*

Number of customers, n	Δ HV	N_{exact}	$N_{approximate}$
6	2.67%	3.6	5.6
7	−4.93%	4.2	7.6
8	9.08%	4.4	9.8
9	−8.29%	4.2	9
10	8.08%	3.4	11
11	−5.78%	3.75	10.6

Although five solutions were returned by CPLEX for each instance, not all of the solutions were distinct, resulting in an average of fewer than five solutions returned for the instances. For instances containing 7, 9, and 11 customers, the average HVs of the sets of non-dominated solutions returned by the approximate solution were worse (*i.e.* smaller) than those returned by CPLEX. For instances containing 6, 8, and 10 customers, on the other hand, the average HVs of the sets of non-dominated solutions returned by the approximate solution were better (*i.e.* larger) than those returned by CPLEX. Although CPLEX returned Pareto-optimal solutions to some of the instances, a larger HV could be obtained by the approximate solution for these instances, since the Pareto-optimal solutions as well as additional non-dominated solutions could be returned. Furthermore, the average number of non-dominated solutions returned by the approximate solution approach was more than the average number of distinct solutions returned by CPLEX for each number of customers in the instances. Upon visual inspection of the objective space of the non-dominated solutions returned by both solution approaches for each instance, it was clear that the approximate solution approach regularly returned the same solutions as those returned by CPLEX, as well as additional non-dominated solutions. The approximate solution approach is therefore considered to have been verified successfully.

A comparison between the run times recorded when solving the aforementioned thirty problem instances according to the exact solution approach (the total time required for returning five solutions to each instance) and those required by the approximate solution approach is presented graphically on a logarithmic scale in Figure 5.7. The run times returned by the exact solution approach are indicated in green, whereas those returned by the approximate solution approach are indicated in blue. Furthermore, the average run times returned by each solution approach for each number of customers is indicated by the orange lines. It is evident that the slope of increase of the average run times required by the approximate solution approach is much smaller than

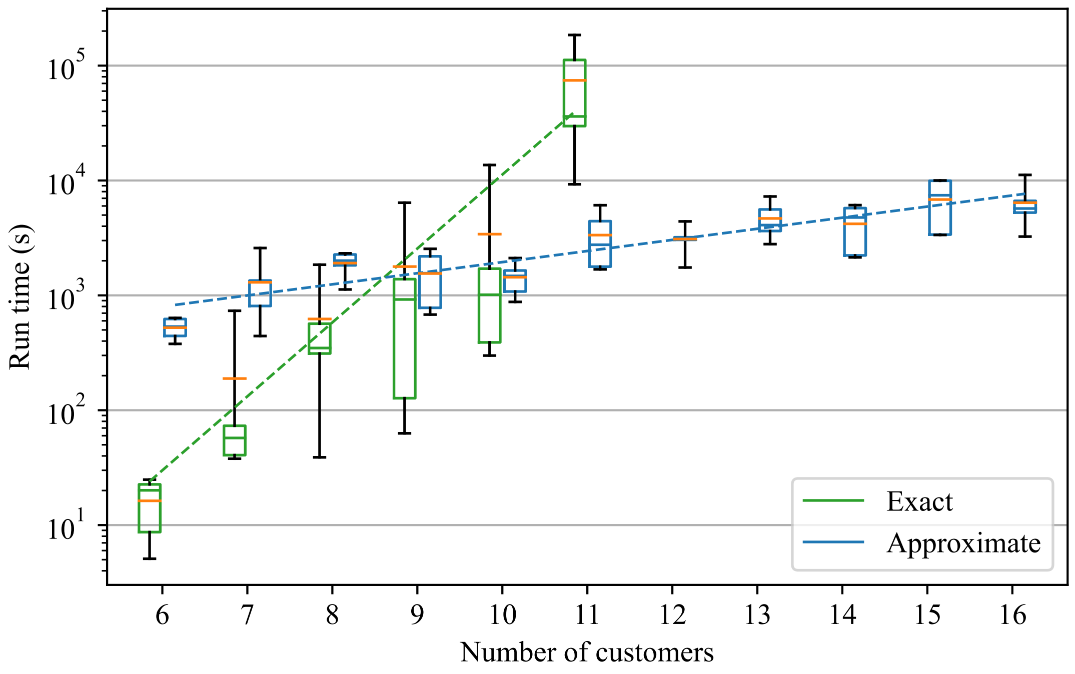


FIGURE 5.7: *The run times recorded when generating non-dominated solutions to synthetic instances of the model (5.1)–(5.17), involving different numbers of customers, during the operational phase of the FVRP exactly and approximately. The mean trend of the run times is indicated by means of dotted lines.*

that of the exact solution approach. The run times of the exact solution approach also fluctuated to a much larger extent than those of the approximate solution approach. It is important to note that the average run times of the exact solution approach for instances containing eleven customers were affected by the run time limit of eight hours imposed for each solution in the instance, whereas the approximate solution approach never met this stopping criterion. A further set of problem instances (containing twelve to sixteen customers) was also solved according to the approximate solution approach and a similar slope of increase was observed as for the smaller sized problem instances. The solution time incurred by the approximate solution approach is considered to be acceptable.

5.7 Chapter summary

In this chapter, a mathematical model was proposed for the operational phase of the FVRP. The model was derived in §5.1 after which an exact solution approach was proposed in §5.2 and implemented for the model in the CPLEX optimisation environment, using its Python interface. This was followed by a systematic model verification in §5.3 in which a random problem instance generator was described and used to generate thirty problem instances for model verification purposes. The time complexity of the exact solution approach was discussed in §5.4 and it was found that the exact solution approach is not scalable to the size of real-world problem instances. An approximate solution approach was therefore proposed in §5.5, based on the HGSADC algorithm of Vidal *et al.* [140] and solution evaluation techniques proposed by Deb *et al.* [38]. A systematic verification of the approximate solution approach followed in §5.6 during which it was found that the approximate solution approach is acceptable and that it is capable of returning high-quality sets of non-dominated solutions.

Part III

Validation Case Study

CHAPTER 6

Case study background and data

Contents

In an attempt to demonstrate the practical applicability of the FVRP proposed in this thesis, a case study is performed based on real-world input data to the two mathematical models proposed in Chapters 4 and 5, provided by the industry partner attached to this thesis. The models are solved approximately. The aim in this chapter is to provide background information about the industry partner and to present the input data related to the case study. The industry partner and its current logistical operations are discussed briefly in §6.1. This is followed in §6.2 by a discussion on the real-world input data related to the case study. The chapter is brought to a close with a brief summary of its contents in §6.3.

6.1 The industry partner

The industry partner attached to this thesis is a clothing retailer with one of the largest retail store footprints in Southern Africa. It owns 5 470 stores distributed across ten African countries, sells predominantly clothing, footwear, and textiles, and boasts an annual revenue of R77.3 billion. A large part of its supply chain competitiveness relies on an ability to distribute 24 million boxes of retail goods annually from its depots to its stores.

The current logistical operations of the industry partner in South Africa involve stock arriving at ports, from where it is distributed to DCs by road. From the DCs, stock is distributed further by road to 23 depots located across South Africa. At these depots, different stock items are merged into pallets which are distributed by road to 3 500 stores, with each store only being serviced by a single depot. During the demand planning process at the industry partner, the volume of stock to be delivered to each store during a specific week is determined well in advance. The products that make up the assigned volume of stock are only determined at a later stage, and may change throughout the planning process. Stock is procured and delivered to the relevant depots before the start of the week during which it should be delivered to stores. The demand exhibited by each store may then be satisfied on any day, or on multiple days, throughout the week. Stores are therefore not assigned any specific day in advance on which delivery will occur, but rather a specific week during which the delivery of certain stock items will take place.

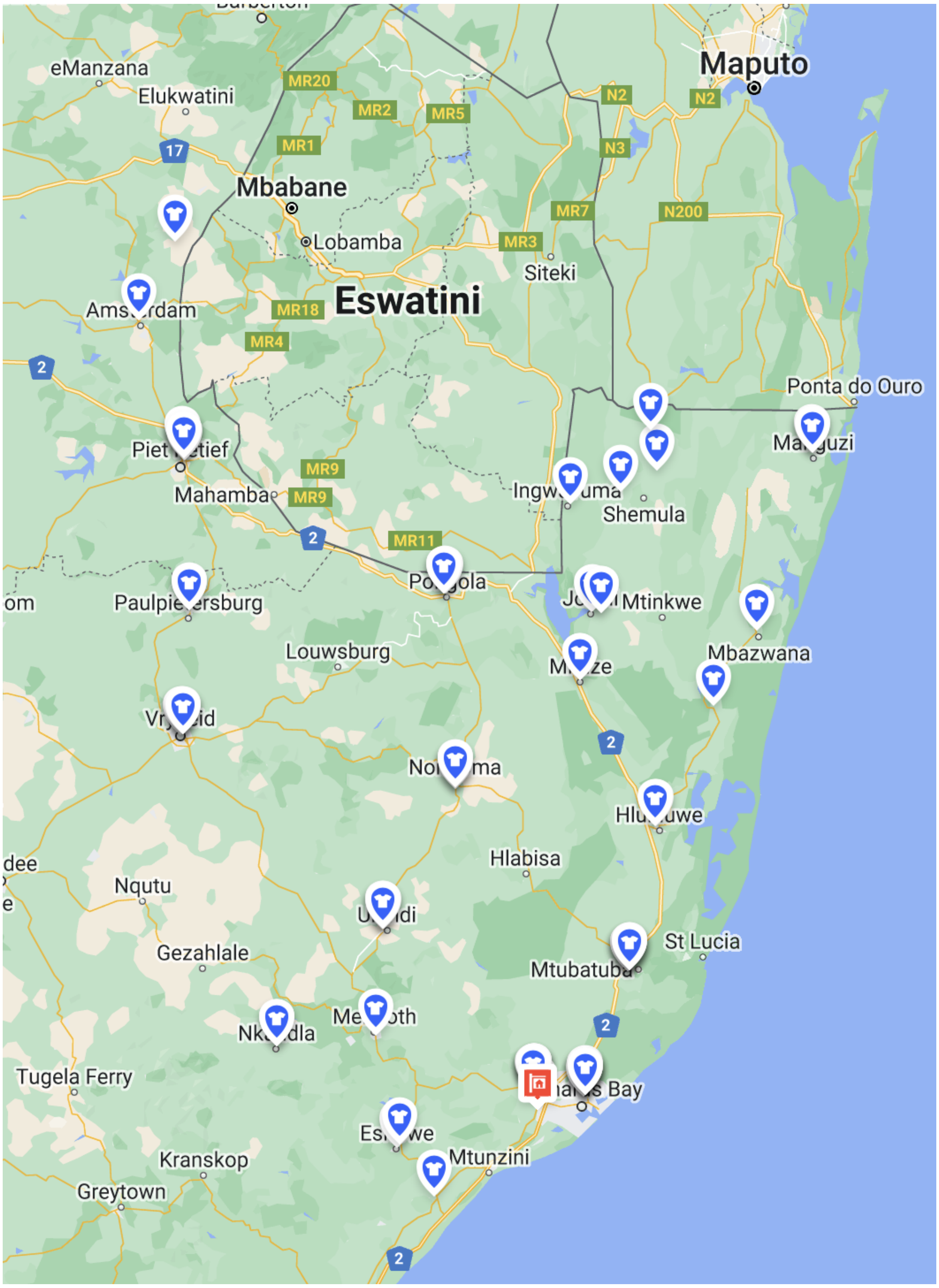


FIGURE 6.1: *Locations of the EPH depot and its stores. Stores are represented by blue flags, whereas the depot is represented by a red flag.*

The volume of stock delivered to a single store during any day may, however, not exceed a specified threshold. This ensures that the storage capacity of a store is not exceeded and allows staff to unpack stock delivered to the store at convenient off-peak times throughout the day. Furthermore, deliveries at stores are only allowed to take place during the period 09:00–17:00 on weekdays, and depots operate during the period 08:00–20:00. The industry partner does not own a dedicated fleet of delivery vehicles, but hires delivery vehicles from a third party. The company therefore essentially has an unlimited fleet of delivery vehicles available to service customers. Two types of delivery vehicles, small or large delivery vehicles, may be rented, and different fixed and variable costs are associated with renting these delivery vehicles.

As mentioned in §1.2, the industry partner often experiences practical outbound logistics challenges when attempting to implement the recommendations stemming from solving VRP instances by means of standard commercial software. This reportedly occurs frequently because there is little similarity between the day-to-day routes assigned to delivery vehicle drivers, and the industry partner therefore has expressed the need to increase driver-route familiarity. Although increasing driver-route familiarity may result in an increased transportation cost of planned delivery routes, the final cost of implementing these solutions and the total cost across all supply chain activities are nevertheless expected to decrease if the implementation of planned delivery routes can be improved.

6.2 Input data

A medium-sized depot of the industry partner, servicing 93 stores, is its *Empangeni hub* (EPH) situated near Richards Bay in the Kwazulu-Natal Province of South Africa. The historical demand data associated with the EPH depot for the period 13/09/2021–26/12/2021, spanning 16 weeks, serves as input data to the case study reported in this chapter and the next. The locations of the EPH depot and its assigned stores are provided in Table 6.1 and illustrated graphically on a road map in Figure 6.1. Further information associated with each store is also summarised in Table 6.1, while information pertaining to the types of delivery vehicles available for renting is provided in Table 6.2. In particular, the locations, time-window start and end times, and the storage capacity of each store are provided in Table 6.1, which also contains an indication of the average weekly demand volume over the first 13 of the aforementioned 16 weeks (for the weeks 13/09/2021–05/12/2021), as well as the demand volume for three subsequent weeks (for the weeks 06/12/2021–26/12/2021) associated with each store.

Some differences exist between the real-world data employed in this case study and the test data sets employed in Chapters 4 and 5. In the test data sets, the depot was always placed relatively in the centre of the transportation network (although a change of the parameters to the data set generators allowed for varying depot locations slightly), whereas the EPH depot is placed near a port at the outskirts of the transportation network. Furthermore, the locations of customers in the test data sets were uniformly distributed, whereas the stores serviced by the EPH depot are semi-clustered. The Google Maps application programming interface was invoked to determine the travel distances and expected travel times between all pairs of coordinates of locations in Table 6.1. The results indicated that stores are located at a travel distance of up to four hundred kilometres, involving a travel time of up to 287 minutes, away from the depot — much further than the maximum travel distance of 70km and travel time of 70 minutes assumed in the test data sets. Finally, the volumes of demand associated with customers in relation to the capacities of delivery vehicles are much larger in the current case study than that in the test data sets. Due to the larger demand volumes and stores being further located from the depot, delivery vehicles are not able to service as many customers as in the solutions to the test data set instances.

Table 6.1: *The input data associated with stores. The longitude and latitude coordinates of each store i are given. The time-window start and end times, measured in minutes from 08:00 and denoted by a_i and b_i, respectively, as well as the maximum storage capacities of stores, measured in cubic meters and denoted by e_i, are also given. The average demand volumes associated with stores over the period of 13 weeks 13/09/2021–05/12/2021 and the demand volume exhibited by these stores during three subsequent weeks 06/12/2021–26/12/2021 are denoted by q_i^{avg}, q_i^1, q_i^2, and q_i^3, respectively, and are measured in cubic meters.*

Store, i	Longitude	Latitude	a_i	b_i	q_i^{avg}	q_i^1	q_i^2	q_i^3	e_i
1	–28.4141	32.1875	60	540	57.35	63.34	37.84	24.75	25
2	–28.7523	32.0507	60	540	87.33	113.70	66.72	37.92	25
3	–27.0055	30.8020	60	540	63.11	63.02	39.07	24.43	25
4	–27.3792	31.6110	60	540	72.87	85.76	41.54	23.93	25
5	–28.7523	32.0507	60	540	16.81	32.76	16.09	15.39	25
6	–28.7523	32.0507	60	540	14.68	9.79	21.65	5.83	25
7	–27.4328	32.1005	60	540	42.25	46.98	35.30	21.52	25
8	–28.7504	32.0477	60	540	38.53	41.04	28.00	15.93	25
9	–26.9919	30.7987	60	540	37.66	26.21	21.09	12.93	25
10	–28.6198	31.0914	60	540	49.36	55.65	25.74	20.32	25
11	–28.7429	31.8879	60	540	60.30	73.05	42.71	17.36	25
12	–27.7650	30.8006	60	540	63.96	65.84	34.17	26.42	25
13	–27.9115	31.6474	60	540	59.04	67.40	40.25	22.46	25
14	–28.7418	31.8904	60	540	78.77	47.24	26.72	29.64	25
15	–28.7590	32.0440	60	540	55.59	58.02	35.67	29.72	25
16	–27.4245	30.8200	60	540	54.98	55.10	27.79	20.08	25
17	–27.0105	30.8063	60	540	55.83	51.71	31.04	20.44	25
18	–27.6174	32.0330	60	540	33.85	37.79	22.29	12.74	25
19	–28.4113	32.1865	60	540	41.99	47.03	31.12	20.75	25
20	–28.0183	32.2681	60	540	39.09	43.54	24.60	13.28	25
21	–28.3001	31.4220	60	540	52.94	67.46	38.35	24.45	25
22	–28.8892	31.4720	60	540	41.20	41.03	29.95	20.22	25
23	–27.7659	30.7998	60	540	12.17	24.44	19.26	5.61	25
24	–27.4241	30.8190	60	540	24.37	49.19	37.29	9.67	25
25	–28.7431	31.8915	60	540	15.72	26.20	24.91	8.22	25
26	–27.0094	30.8023	60	540	20.03	35.17	30.00	7.24	25
27	–27.3779	31.6151	60	540	15.70	28.38	26.29	7.20	25
28	–28.7517	32.0488	60	540	38.22	68.78	59.61	13.59	25
29	–28.5910	31.3985	60	540	16.36	31.61	30.63	6.41	25
30	–28.4141	32.1873	60	540	17.18	34.17	31.06	6.41	25
31	–28.4114	32.1865	60	540	16.21	33.21	29.71	6.32	25
32	–29.0306	31.5815	60	540	18.33	35.35	29.55	7.01	25
33	–27.9113	31.6473	60	540	22.11	44.11	37.09	8.23	25
34	–28.2984	31.4225	60	540	30.50	55.41	47.18	9.18	25
35	–27.7646	30.7955	60	540	16.22	31.84	18.77	6.88	25
36	–28.4136	32.1854	60	540	25.46	43.11	38.74	9.72	25
37	–27.7643	30.8006	60	540	22.80	42.16	26.51	11.39	25
38	–27.0054	30.8033	60	540	13.06	23.85	19.42	4.59	25
39	–27.7670	30.7967	60	540	15.41	31.19	22.79	6.27	25
40	–28.7492	32.0473	60	540	19.97	44.63	22.65	7.01	25

Continued on next page

Store, i	Longitude	Latitude	a_i	b_i	q_i^{avg}	q_i^1	q_i^2	q_i^3	e_i
41	–28.7448	31.8891	60	540	24.64	44.83	32.90	9.64	25
42	–26.9905	32.7551	60	540	20.56	38.08	29.22	7.68	25
43	–27.7694	30.7919	60	540	16.19	32.82	20.91	7.48	25
44	–27.9168	31.6465	60	540	11.20	22.75	16.71	3.77	25
45	–28.7523	32.0507	60	540	23.44	49.69	30.60	9.84	25
46	–26.9260	32.2519	60	540	15.37	31.70	20.77	5.68	25
47	–28.3022	31.4217	60	540	19.53	45.28	28.27	8.85	25
48	–27.4300	32.0691	60	540	18.51	35.74	26.59	6.73	25
49	–28.8987	31.4640	60	540	19.81	34.07	29.87	5.86	25
50	–28.0190	32.2689	60	540	28.75	52.15	35.09	11.33	25
51	–28.7442	31.8862	60	540	10.56	15.12	25.14	3.69	25
52	–28.8963	31.4664	60	540	29.65	55.71	37.56	11.75	25
53	–28.7733	31.9035	60	540	18.49	30.36	24.24	8.13	25
54	–26.6239	30.6615	60	540	12.65	24.84	17.82	5.54	25
55	–28.7499	32.0474	60	540	18.26	35.51	24.64	9.78	25
56	–27.9085	31.6469	60	540	30.04	52.71	37.95	10.24	25
57	–27.3788	31.6111	60	540	24.66	43.34	31.78	9.26	25
58	–27.6171	32.0336	60	540	18.72	38.47	26.63	6.21	25
59	–26.9885	32.7561	60	540	16.83	31.36	24.37	6.82	25
60	–28.6213	31.0913	60	540	28.96	50.87	37.12	10.19	25
61	–27.4277	32.0706	60	540	19.16	31.04	28.20	8.44	25
62	–27.4815	32.5827	60	540	31.11	55.71	37.04	11.79	25
63	–27.0066	30.8037	60	540	12.97	23.11	19.35	5.09	25
64	–27.1334	32.0040	60	540	17.55	31.26	25.36	6.45	25
65	–28.7418	31.8894	60	540	29.45	56.06	40.48	11.48	25
66	–28.7431	31.8880	60	540	25.68	44.55	31.88	11.00	25
67	–28.4153	32.1807	60	540	8.85	26.36	9.40	4.76	25
68	–27.0990	32.1598	60	540	16.62	29.27	18.86	7.25	25
69	–27.9120	31.6471	60	540	15.55	31.60	21.43	7.61	25
70	–27.4328	32.1005	60	540	15.01	25.69	22.14	4.50	25
71	–28.7418	31.8900	60	540	20.32	35.37	28.69	6.75	25
72	–28.2982	31.4221	60	540	12.82	24.62	19.49	4.18	25
73	–28.2982	31.4221	60	540	7.70	19.64	10.65	4.02	25
74	–28.4156	32.1806	60	540	15.27	32.59	22.80	5.63	25
75	–28.7435	31.8877	60	540	8.25	18.66	9.52	4.10	25
76	–28.7494	32.0468	60	540	16.68	31.76	25.50	7.28	25
77	–28.6213	31.0913	60	540	10.35	23.89	10.90	3.79	25
78	–28.7460	32.0509	60	540	18.02	40.96	20.49	8.03	25
79	–26.4032	30.7754	60	540	17.70	34.61	19.42	8.25	25
80	–27.7670	30.7931	60	540	11.48	31.81	13.04	4.91	25
81	–28.4144	32.1858	60	540	12.04	20.76	17.80	5.39	25
82	–27.0407	32.2721	60	540	15.83	29.36	23.87	6.97	25
83	–27.3762	31.6128	60	540	14.28	25.67	19.12	6.42	25
84	–26.9862	32.7540	60	540	16.99	33.42	25.96	7.48	25
85	–26.9886	32.7542	60	540	21.51	37.62	25.03	9.30	25
86	–27.4245	30.8200	60	540	14.50	25.33	20.79	5.50	25
87	–27.3781	31.6093	60	540	10.24	28.53	12.01	6.48	25
88	–27.6893	32.4486	60	540	12.18	26.33	16.23	5.51	25

Continued on next page

Store, i	Longitude	Latitude	a_i	b_i	q_i^{avg}	q_i^1	q_i^2	q_i^3	e_i
89	–28.7412	31.8899	60	540	13.07	7.67	15.62	5.55	25
90	–28.7417	31.8903	60	540	71.10	35.62	23.73	16.93	25
91	–26.9919	30.7988	60	540	9.58	16.95	8.99	2.05	25
92	–26.9917	30.7987	60	540	11.29	18.51	15.71	4.18	25
93	–28.4147	32.1805	60	540	18.09	22.53	7.88	2.65	25

TABLE 6.2: *Input data related to delivery vehicle types available for renting. The fixed cost associated with each delivery vehicle (in Rand) is denoted by C_z, whereas the maximum cargo capacity in cubic metres of each delivery vehicle is denoted by Q_z. The variable cost (in Rand per kilometre travelled) of each delivery vehicle is denoted by h_z. A speed factor, denoted by v_z, is associated with each type of delivery vehicle. Finally, each type of delivery vehicle is associated with a cost coefficient, denoted by β_z Rand per minute, with which the duration of each route is penalised in the objective function.*

Vehicle type, z	Size	C_z	Q_z	h_z	v_z	β_z
1	Small	2 600	25	14	0.9	0.1
2	Large	3 600	50	20	1	0.1

Since demand planning at the industry partner is performed for individual weeks and deliveries may take place on weekdays, a planning horizon in the FVRP is taken as a week in this case study, consisting of five decision periods, each representing a weekday. The historical demand for 13 consecutive planning horizons (weeks) were used to determine the average weekly demand volume associated with stores. The master routes computed during the strategic phase are based on these average demand volumes, which are given in the q^{avg}-column in Table 6.1. During the operational phase of the FVRP, actual delivery routes are computed for three subsequent planning horizons (weeks) for which the demand volumes associated with stores are given in the q^1-, q^2-, and q^3-columns in Table 6.1. The average demand volume associated with customers, when computing the master routes, is 26.40m^3. Moreover, the average demand volumes associated with customers when computing the actual delivery routes are 10.53m^3, 26.83m^3, and 38.91m^3 for the first, second, and third planning horizon, respectively. The distribution of these demand volumes is illustrated graphically in Figure 6.2. The demand volumes of the three planning horizons for which actual delivery routes are to be computed are low, medium, and high demand volume planning horizons compared to those used to compute the master routes, and are therefore expected to provide insight into the effect of varying demand volumes on the effectiveness of the FVRP.

The deviations of the demand volumes associated with stores from their average demand volumes during each of the planning horizons are illustrated graphically in Figure 6.3. Recall from Chapter 5 that in the randomly generated test data sets used for verification purposes, the actual demand volumes associated with customers were distributed randomly according to a normal distribution with the average demand volumes associated with customers as mean. Upon visual inspection of Figure 6.3 it would seem that the deviation of demand volumes associated with stores of the EPH depot are also approximately normally distributed, but with means distinct from the average demand volumes of stores used to compute the master routes.

The variation in demand volumes associated with stores results in varying visitation frequencies required by these stores during future planning horizons. The visitation frequencies of stores calculated for master route generation are illustrated graphically in Figure 6.4. Stores of the EPH depot require a total of 139 visits by the master routes, whereas 97, 146, and 193 visits

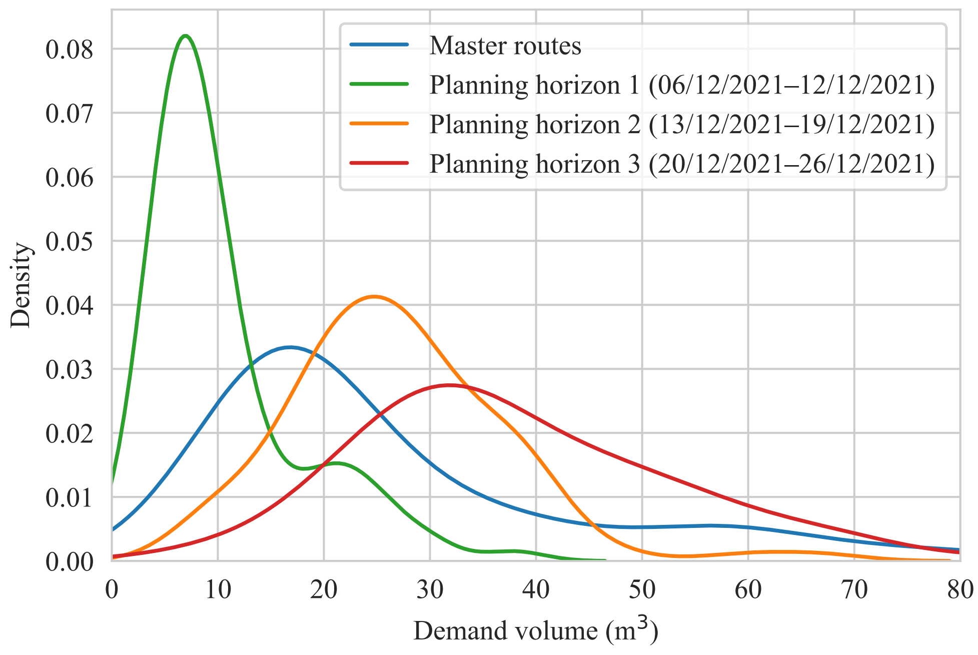


FIGURE 6.2: *The distribution of demand volumes exhibited by stores when computing the master routes and actual delivery over for different planning horizons.*

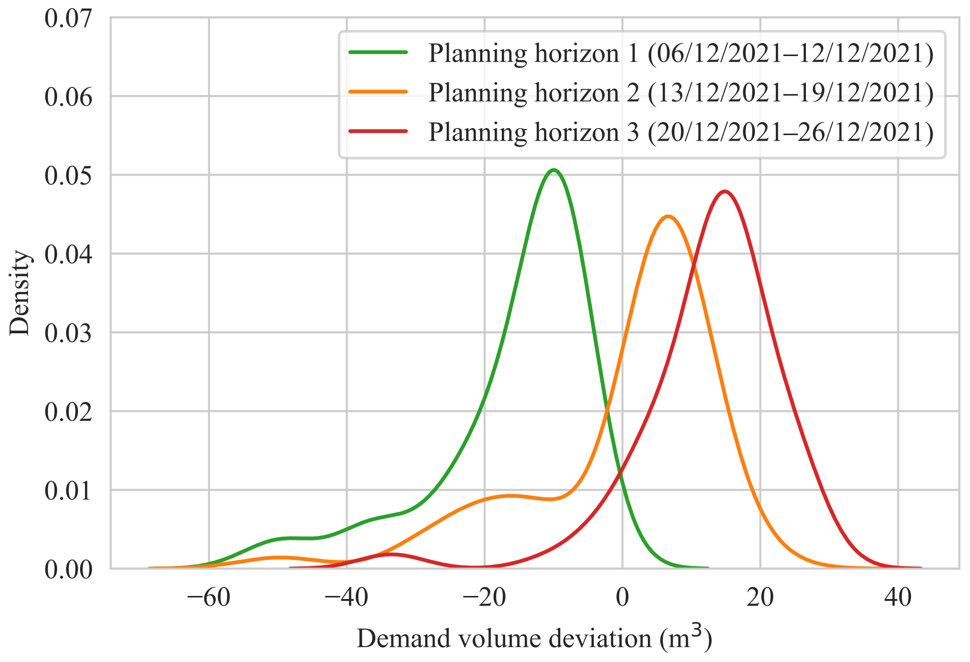


FIGURE 6.3: *The distribution of demand volume deviations exhibited by stores when computing actual delivery routes over three different planning horizons.*

are required in total during the first, second, and third planning horizon, respectively. The total number of visits required by customers is expected to have a significant effect on the run time required by the approximate solution approach described in Chapter 5, since a larger total number of visits would require invoking the split algorithm more often to determine the best insertions of customers into giant tours.

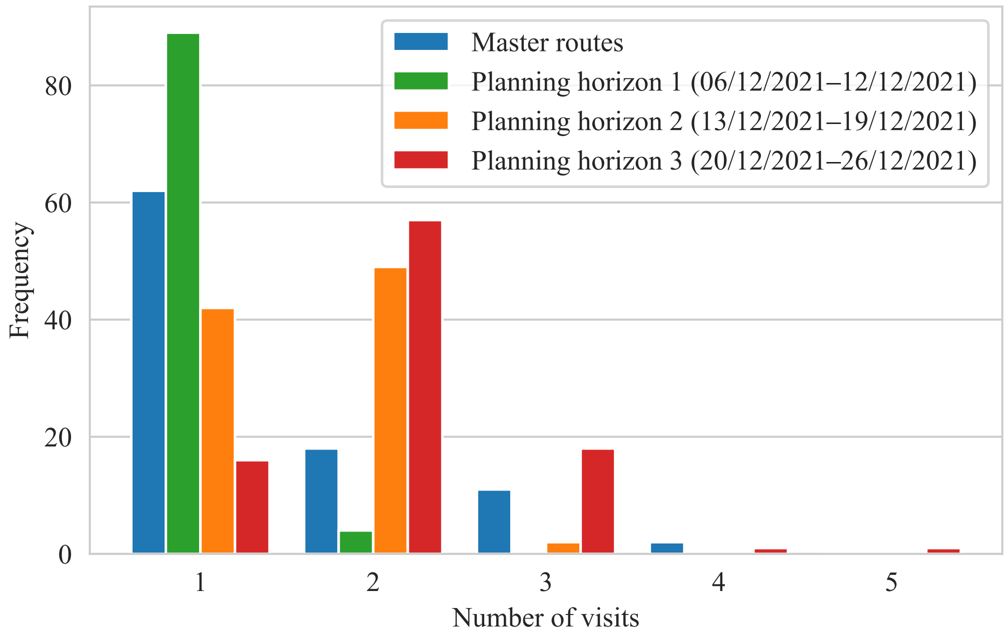


FIGURE 6.4: *The number of visits required by stores when computing the master routes and actual delivery routes, over different planning horizons.*

6.3 Chapter Summary

The purpose of this chapter was to provide background information on the industry partner attached to this thesis, and to present the input data related to the case study performed in the following chapter. The industry partner and its current logistical operations were discussed briefly in §6.1. This was followed in §6.2 by a discussion on the real-world input data related to the case study performed.

CHAPTER 7

Case study results

Contents

This chapter is devoted to a presentation of the numerical results obtained during the case study carried out based on the real-world data described in Chapter 6. The two mathematical models proposed in Chapters 4 and 5 are solved approximately for these data. The master routes computed for the EPH depot of the industry partner during the strategic phase of the FVRP are described in §7.1. This is followed in §7.2 by a discussion on the corresponding solutions obtained during the operational phase of the FVRP for three planning horizons. A brief discussion of the results obtained during the case study is presented in §7.3, and this is followed in §7.4 by an expert validation of the results by an employee of the industry partner. This chapter is brought to a close in §7.5 with a summary of its contents.

7.1 The strategic phase

During the strategic phase of the FVRP, master routes were computed for the EPH depot and its assigned stores by providing the data described in Table 6.1 as input to the model for the strategic phase of the FVRP, and solving the model by invoking the approximate solution approach proposed in §4.5. As additional input to the model for the strategic phase of the FVRP, the allowable threshold in terms of arc overlaps tolerated without penalty amongst the master routes was set to $\alpha_1 = 0$. Furthermore, the cost coefficient associated with penalising the number of arc overlaps amongst the master routes over and above the maximally tolerated (not penalised) number was taken as $\rho = 400$. This cost coefficient was set to a larger value than that for the test data instances in order to compensate for the larger variable cost per kilometre associated with delivery vehicles available in the case study. The approximate solution approach for the strategic phase was then invoked for a maximum of ten thousand non-improving iterations and a maximum run time duration of seven days. Furthermore, ten dummy periods were initialised in order to allow for stores to receive multiple visits, as described in §4.5.4. All other parameters of the approximate solution approach were set to their default values described in §4.5.7.

After seven days of run time on a computer with an Intel Core i7 CPU operating at 2.90GHz with 16GB of memory, the approximate solution approach was terminated and the best feasible solution uncovered throughout the search was returned. The approximate solution approach was able to perform more than twenty thousand iterations during this period and uncovered the first feasible solution within the first day of run time, after which many improving solutions were uncovered. The best solution returned corresponds to an objective function value of R614 731.08, comprising a fixed cost component of R215 800, a variable cost component of R389 135.28, a duration penalty cost of R2 595.80, and an overlap penalty cost of R7 200. There were 18 overlapping arcs amongst the master routes. Of the 8 742 arcs in the 94 vertex transportation network, 174 master route arcs were returned in total as part of the master routes.

A total of 63 master routes were computed, comprising routes for 52 large delivery vehicles and 11 small delivery vehicles. Information about the route for each delivery vehicle is summarised in Table 7.1. The average utilisation of the vehicle capacities was 85.18%. The volume of commodities delivered to stores was much larger than that in the test data instances, which resulted in fewer stores being visited by each delivery vehicle. An average of 2.44 stores were assigned to large delivery vehicles, whereas an average of 1.09 stores were assigned to small delivery vehicles. Since the capacity of a small delivery vehicle is equal to the storage capacities of stores, a few master routes consist merely of a small delivery vehicle departing from the depot and returning back directly to the depot after having visited only a single customer. More large delivery vehicles were therefore utilised since they are able to service at least two customers (their capacities are twice as large as the maximum storage capacity of customers).

TABLE 7.1: *Master routes returned for the case study. Routes are given by listing the stores assigned to a vehicle in the order in which they are visited. Arrival times are given in minutes after 08:00. Vehicle utilisation is given as a fraction between 0 and 1, and vehicle distances travelled are specified in kilometres.*

Vehicle type	Delivery route	Arrival times	Vehicle utilisation	Distance travelled
Large	(36, 74, 2)	(0, 153.92, 286.35, 594.92)	0.9966	120.69
Large	(76, 18, 31)	(0, 155.18, 387.71)	0.9963	370.23
Large	(44, 13, 58)	(0, 102.38, 265.55)	0.9920	369.71
Large	(56, 87, 4)	(0, 120.44, 298.46)	0.9911	438.93
Large	(17, 92, 20)	(0, 108.64, 267.09)	0.9888	632.14
Large	(50, 9, 35)	(0, 102.2, 184.98, 357.65)	0.9885	632.76
Large	(56, 27, 17)	(0, 60, 150.78, 227.86)	0.9865	626.71
Large	(83, 82, 61)	(0, 60, 159.33)	0.9855	578.31
Large	(60, 52, 14)	(0, 140.0, 364.24)	0.9799	271.91
Large	(57, 4)	(0, 107.68, 284.06)	0.9789	432.16
Large	(69, 34, 73, 77)	(0, 155.18, 235.51, 449.87)	0.9771	240.98
Large	(71, 62, 36)	(0, 107.68, 242.64, 466.94)	0.9720	428.69
Large	(18, 20, 81)	(0, 222.53, 363.56, 496.48, 640.41)	0.9702	316.97
Large	(14, 67, 49)	(0, 93.77, 254.77)	0.9670	152.46
Large	(86, 65, 1)	(0, 227.6, 294.61, 583.85)	0.9669	120.06
Large	(66, 6, 42)	(0, 140.86, 340.55)	0.9617	526.24
Large	(38, 17, 43)	(0, 148.98, 231.66, 439.49)	0.9572	610.92
Large	(60, 26, 72)	(0, 133.92, 343.41)	0.9468	705.53
Large	(75, 22, 32)	(0, 60, 145.19)	0.9436	152.72
Large	(24, 37)	(0, 168.42, 403.63)	0.9434	507.00

Continued on next page

Vehicle type	Delivery route	Arrival times	Vehicle utilisation	Distance travelled
Large	(28, 52, 89)	(0, 169.33, 403.97, 616.99)	0.9402	179.46
Large	(80, 16, 68)	(0, 147.9, 274.49, 395.85, 603.67)	0.9284	694.33
Large	(45, 2)	(0, 79.38, 184.57, 293.03, 351.61)	0.9054	36.78
Large	(10, 11)	(0, 103.38, 200.12, 325.57)	0.8956	250.40
Large	(46, 84, 88)	(0, 222.03, 358.45, 587.3)	0.8909	575.26
Large	(3, 85)	(0, 60, 184.47)	0.8509	611.29
Large	(12, 7)	(0, 184.65, 425.13)	0.8489	564.02
Large	(21, 90)	(0, 103.15, 172.04, 354.17)	0.8270	239.39
Large	(8, 2)	(0, 139.8, 337.57)	0.8219	36.89
Large	(33, 48)	(0, 130.67, 327.04)	0.8123	426.31
Large	(15, 2)	(0, 60, 146.36)	0.8072	36.40
Large	(79, 91, 63)	(0, 60, 177.3)	0.8051	785.65
Large	(29, 90)	(0, 60, 157.53)	0.8012	148.88
Large	(11, 14)	(0, 173.76, 404.53)	0.7959	11.32
Large	(11, 14)	(0, 60, 182.94, 269.09)	0.7959	11.32
Large	(12, 16)	(0, 136.47, 350.12)	0.7930	506.58
Large	(41, 65)	(0, 178.77, 415.78)	0.7873	11.37
Large	(16, 3)	(0, 184.65, 425.13)	0.7872	610.76
Large	(90, 70)	(0, 136.52, 348.46)	0.7743	148.88
Large	(13, 21)	(0, 60, 176.2, 296.84)	0.7466	326.32
Large	(13, 64)	(0, 153.78, 365.65)	0.7445	546.70
Large	(47, 21)	(0, 137.37, 356.77)	0.7435	238.72
Large	(28, 78)	(0, 102.38, 265.55)	0.7427	37.59
Large	(15, 55)	(0, 60.73, 189.92, 283.06)	0.7358	37.13
Large	(12, 39)	(0, 60, 143.58, 252.17)	0.7347	404.97
Large	(7, 62)	(0, 60, 151.71)	0.7336	426.47
Large	(1, 5)	(0, 261.23, 585.3)	0.7185	120.02
Large	(9, 59)	(0, 132.32, 329.33)	0.7132	843.71
Large	(25, 53)	(0, 60, 138.19, 262.1)	0.6842	10.40
Large	(66, 22)	(0, 103.23, 182.86, 345.55)	0.6688	150.71
Large	(50, 30)	(0, 61.48, 114.24, 181.44, 302.59)	0.6310	217.10
Large	(51, 8)	(0, 60, 152.18)	0.5966	42.87
Small	(54, 23)	(0, 71.77, 221.82)	0.9925	706.67
Small	(10)	(0, 101.97, 269.93)	0.9871	249.87
Small	(4)	(0, 60, 186.8)	0.9716	432.16
Small	(3)	(0, 60, 264.4, 378.03, 587.47)	0.8414	610.93
Small	(19)	(0, 60, 157.53)	0.8398	113.52
Small	(19)	(0, 60, 163.16, 271.36)	0.8398	113.52
Small	(40)	(0, 203.86, 482.08)	0.7988	37.13
Small	(1)	(0, 60, 144.29, 202.41)	0.7646	112.55
Small	(15)	(0, 60, 176.14, 256.51)	0.7412	35.54
Small	(93)	(0, 103.23, 232.96, 457.1)	0.7236	111.41
Small	(34)	(0, 61.03, 180.3)	0.6100	239.09

7.2 The operational phase

During the operational phase, actual delivery routes were computed for three planning horizons (weeks) by providing the data described in Table 6.1 and the master routes computed in §7.1 as input to the model for the operational phase of the FVRP. The model was solved approximately by invoking the approximate solution approach proposed in §5.5 for each planning horizon. The demand volumes exhibited by stores for these three planning horizons correspond to the q^1-, q^2-, and q^3-columns in Table 6.1. All parameters of the algorithm were set to their default values as described in §5.5.8. The approximate solution approach of the model for the operational phase was invoked for a maximum of ten thousand non-improving iterations or a maximum run time duration of seven days on a computer with the same specifications as those listed in the previous section, whichever occurred first.

After one day of run time, no feasible solutions had been uncovered yet for the third planning horizon (corresponding to high demand volumes). This occurred due to the split algorithm extracting routes from giant tours in such a manner that the load associated with each route was just less than double the capacity of the delivery vehicle utilised, as mentioned in §3.5. Since relatively large demand quantities are associated with stores, the education procedure was not able to reduce the loads associated with routes enough in order for solutions to be feasible. The approximate solution approach was therefore adapted slightly, so that the split algorithm could find optimal route delimiters in such a manner that the load of a route does not exceed 1.1 times (rather than twice) the capacity of the delivery vehicle utilised. After this adaption, the algorithm was restarted and was able to find feasible solutions for all planning horizons (including the third planning horizon) within the first day of run time.

The maximum number of non-improving iterations was never reached when computing actual delivery routes for any of the three planning horizons. During each run, the approximate solution approach terminated after the maximum run time duration of seven days, and a set of non-dominated solutions was returned for each instance. Since stores required varying numbers of visits during each planning horizon, the approximate solution approach also exhibited varying performance in terms of speed. When solving for the first planning horizon which required the smallest total number of visits, approximately 38 000 iterations were completed before termination of the algorithm. For the second and third planning horizons, requiring larger numbers of total visits, approximately 18 000 and 7 000 iterations were completed before termination of the algorithm, respectively.

The set of non-dominated solutions returned for each planning horizon is illustrated graphically in objective function space in Figure 7.1. For each planning horizon, multiple non-dominated solutions were returned, as may be seen in the figure. As was the case for the testing data instances, a clear trade-off is observable between the transportation cost and the degree of familiarity associated with solutions. As expected, the transportation cost objective function values are larger for solutions returned for planning horizons during which stores exhibit larger demand volumes and require more visits. It is evident that more high-familiarity solutions could be returned for the first and second planning horizons than for the third planning horizon. This can be ascribed to the slower performance of the approximate solution approach when computing actual delivery routes for planning horizons during which larger total numbers of visits are required.

A final trade-off solution was selected for each planning horizon (indicated by the black circles in Figure 7.1). In the case of the first and second planning horizons, solutions on the elbow of the set of non-dominated solutions were selected. For the third planning horizon, the solution corresponding to the highest familiarity objective function value was selected, since the solution

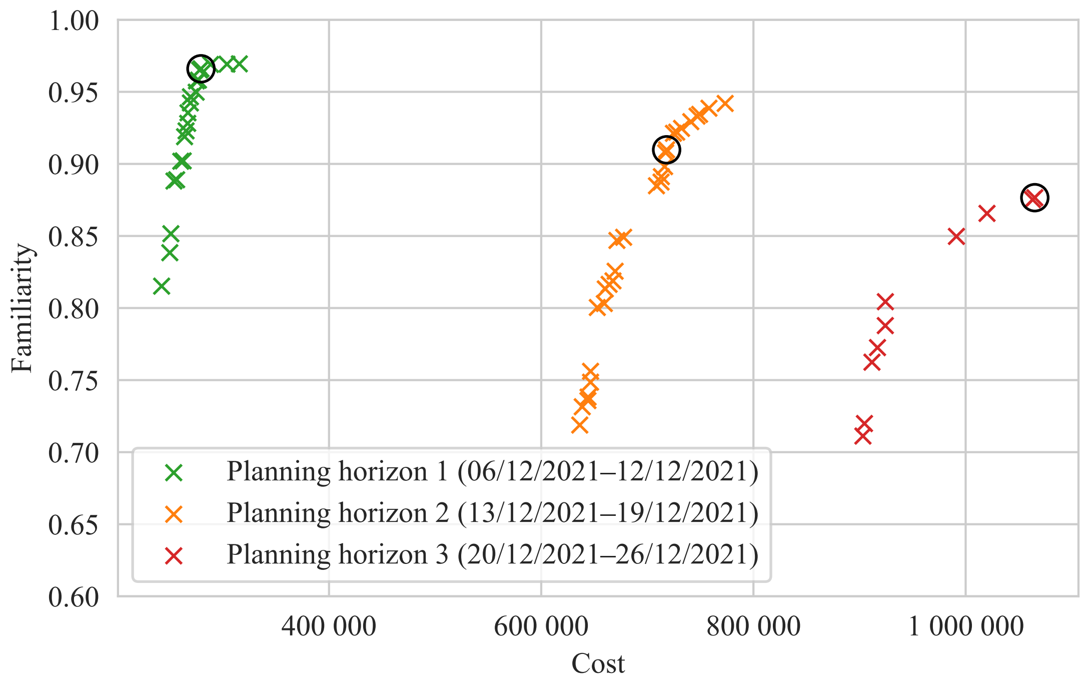


FIGURE 7.1: *The non-dominated solutions returned for three planning horizons, plotted in objective function space.*

on the elbow of the set of non-dominated solutions corresponds to a relatively small familiarity objective function value compared to the solution achieving the largest familiarity objective function value. Information about the sets of non-dominated solutions returned and the solution selected for each planning horizon is summarised in Table 7.2. Significantly more non-dominated solutions were returned for the first and second planning horizons than for the third planning horizon. Moreover, the range of the cost objective function value between the extremal solutions of the non-dominated front returned was larger for planning horizons during which a larger total number of visits were required. The range of the familiarity objective function values between extremal solutions was, however, largest for the second planning horizon (which corresponds with a total number of visits that is closest to that of the master routes). For each planning horizon, the selected trade-off solution was compared with the solution achieving the lowest transportation cost objective function value. During the second planning horizon, the largest increase in familiarity could be achieved for the smallest percentage increase in transportation cost. For an increase of 0.1910 in familiarity, a cost increase of only 12.87% was observed.

TABLE 7.2: *Information about the sets of non-dominated solutions returned for each planning horizon. The number of non-dominated solutions returned for each planning horizon, as well as the range of the cost (in Rand) and familiarity objective function values corresponding to the extremal solutions is given. The percentage increase in cost and corresponding increase in familiarity of the selected trade-off solutions (indicated by circles in Figure 7.1) are compared with these solutions achieving the lowest cost objective function value in each planning horizon.*

Planning horizon	No of solutions	Cost range	Familiarity range	Δ Cost	Δ Familiarity
1	21	73 231.38	0.1542	15.31%	0.1508
2	28	137 054.39	0.2231	12.87%	0.1910
3	10	161 993.85	0.1654	17.94%	0.1654

The selected trade-off solution for each planning horizon was further inspected. Information on these solutions are presented in Table 7.3. For each solution, it was verified that the capacity of no delivery vehicle is exceeded. Furthermore, each delivery vehicle starts servicing stores assigned to it within their specified time-windows and departs from and returns back to the depot within the specified time-window of the depot. In each of the three selected solutions, an average utilisation of more than 0.75 was obtained. More delivery vehicles were utilised during those planning horizons corresponding to larger demand volumes.

TABLE 7.3: *Information about the trade-off solutions selected for each planning horizon (those indicated by circles in Figure 7.1). For each solution, the cost objective function value (in Rand) and the familiarity objective function value are given. Furthermore, the fixed cost, variable cost, and duration penalty of each solution is given (in Rand), as well as the number of delivery vehicles utilised and their average utilisation.*

Panning horizon	Cost	Famil-iarity	Fixed cost	Variable cost	Duration penalty	Number of vehicles	Average utilisation
1	279 347.47	0.9660	108 000	171 347.47	1 213.15	35	0.7576
2	718 175.66	0.9098	274 200	443 975.66	3 071.67	92	0.7638
3	1 065 165.92	0.8766	401 200	663 965.92	4 427.17	132	0.7663

Information about the routes of the selected trade-off solutions for each decision period of each planning horizon is presented in the remainder of this section. For each route, the type of delivery vehicle utilised, the sequence of stores visited, the service start times at these stores (measured in minutes after 08:00), the utilisation of the delivery vehicle, the distance of the route (in kilometres), and the driver-route familiarity associated with the route are given. Information about the routes associated with the first, second, and third planning horizons are summarised in Tables 7.4–7.6, respectively.

During the first planning horizon, 17 large delivery vehicles and 18 small delivery vehicles are utilised. The large and small delivery vehicles utilised are able to service more stores per route (up to six stores for large delivery vehicles and up to four stores for small delivery vehicles) than in the master routes, due to the lower demand volumes exhibited by stores. Furthermore, routes that are associated with a low degree of driver-familiarity are generally shorter in distance, whereas the routes having achieved a large degree of driver-familiarity are longer routes.

During the second planning horizon, 35 large delivery vehicles and 57 small delivery vehicles are utilised. Generally, fewer customers are visited by each delivery vehicle than in the routes of the first planning horizon. This is because the demand volumes exhibited by stores are generally larger in the second planning horizon than in the first. Large delivery vehicles service up to three stores per route while small delivery vehicles service up to two stores per route. Most of the small delivery vehicles only service a single store.

As in the case of the second planning horizon, the routes of the third planning horizon are much shorter than the master routes and the routes of the first planning horizon. During the third planning horizon, 58 large delivery vehicles and 74 small delivery vehicles are utilised. The large and small delivery vehicles only service up to three and two stores per route, respectively. Due to the large demand volumes exhibited by stores, most large delivery vehicles only service two customers per route while small delivery vehicles only service a single customer.

Table 7.4: *Delivery routes of the solution selected for the first planning horizon (06/12/2021–12/12/2021), namely the first circled solution in Figure 7.1. The type of delivery vehicle utilised, the sequence of stores visited, the service start times at these stores (measured in minutes after 08:00), the utilisation of the delivery vehicle, the distance of the route (in kilometres), and the driver-route familiarity associated with each route are given.*

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Familiarity
First decision period (the planning horizon 06/12/2021–12/12/2021)					
Large	(51, 89, 78, 15, 45, 6)	(0, 60, 83.26, 136.23, 177.38, 235.48, 274.99, 323.71)	0.9558	49.70	0.1108
Large	(61, 48, 7, 71)	(0, 132.48, 169.84, 207.05, 281.6, 450.79)	0.8686	372.53	0.9886
Small	(1)	(0, 60, 187.25)	0.9900	112.55	1.0000
Small	(77, 2)	(0, 60, 104.11, 190.29)	0.9100	44.08	0.5432
Small	(13)	(0, 134.02, 344.8)	0.8986	305.44	1.0000
Small	(53, 86, 25)	(0, 60, 102.68, 132.23, 174.87)	0.8740	11.90	0.5174
Small	(62, 36)	(0, 130.67, 271.22, 353.03)	0.8606	376.19	0.6465
Second decision period (the planning horizon 06/12/2021–12/12/2021)					
Large	(2, 5, 28)	(0, 60, 126.89, 185.38, 257.19)	0.9587	37.08	0.9857
Large	(8, 15, 40, 55)	(0, 60, 124.33, 182.77, 214.31, 277.77)	0.9516	41.02	0.8949
Large	(80, 16, 12, 35)	(0, 153.92, 214.47, 322.89, 374.83, 561.6)	0.9016	507.62	0.7973
Large	(14, 67, 22)	(0, 60, 172.27, 196.55, 328.01)	0.7961	151.34	0.5731
Large	(64, 46, 82, 68)	(0, 191.13, 269.34, 315.69, 360.51, 562.65)	0.5269	544.53	0.8519
Small	(38, 26, 9)	(0, 203.27, 229.38, 267.37, 523.81)	0.9902	615.44	0.9955
Small	(49, 52, 32)	(0, 60, 89.08, 156.13, 225.2)	0.9847	165.32	0.8529
Small	(50, 20)	(0, 72.17, 119.14, 243.71)	0.9844	215.60	0.9955
Small	(42, 84, 59)	(0, 183.85, 218.1, 252.91, 466.8)	0.8792	513.15	0.9978
Small	(72, 92, 91, 63)	(0, 203.77, 231.01, 253.7, 275.29, 503.71)	0.6202	614.08	0.9942
Third decision period (the planning horizon 06/12/2021–12/12/2021)					
Large	(60, 10)	(0, 112.62, 154.19, 337.7)	0.6101	250.24	0.9991
Large	(34, 69, 73, 47)	(0, 103.15, 142.19, 178.4, 202.49, 343.23)	0.5931	240.63	0.9916
Large	(17, 43)	(0, 222.53, 364.81, 552.37)	0.5583	609.91	0.8317
Large	(29, 70, 90)	(0, 67.13, 96.77, 120.84, 250.77)	0.5569	148.88	0.9978
Small	(81, 31, 74, 30)	(0, 60, 87.99, 122.09, 153.3, 225.57)	0.9498	117.17	0.9688
Small	(18, 58)	(0, 97.89, 146.32, 274.84)	0.7583	306.50	0.9998
Small	(65)	(0, 60, 114.45)	0.4591	11.52	1.0000
Fourth decision period (the planning horizon 06/12/2021–12/12/2021)					
Large	(85, 3, 87, 57)	(0, 222.53, 264.24, 429.12, 460.63, 650.91)	0.9893	621.66	0.8381

Continued on next page

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Familiarity
Large	(66, 41, 75, 11)	(0, 60, 104.38, 144.8, 168.57, 240.67)	0.8419	11.41	0.9329
Large	(83, 27, 4)	(0, 150.78, 182.69, 217.2, 450.38)	0.7510	433.40	0.9976
Small	(54, 79)	(0, 234.85, 289.42, 586.33)	0.5518	786.08	0.9486
Small	(24)	(0, 168.82, 378.72)	0.3870	506.20	1.0000
Small	(88)	(0, 108.64, 244.11)	0.2203	310.71	1.0000
Fifth decision period (the planning horizon 06/12/2021–12/12/2021)					
Large	(23, 39, 12, 37)	(0, 154, 181.8, 213.01, 262.96, 462.73)	0.7295	405.30	0.9977
Large	(56, 44, 33, 76)	(0, 145.68, 190.85, 215.05, 250.78, 429.88)	0.5905	307.29	0.9931
Large	(21)	(0, 103.23, 291.15)	0.4890	238.86	1.0000
Small	(93, 19)	(0, 60, 82.77, 198.96)	0.9362	114.62	0.9812
Small	(14)	(0, 60, 123.51)	0.5927	10.60	1.0000

Table 7.5: *Delivery routes of the solution selected for the second planning horizon (13/12/2021–19/12/2021), namely the second circled solution in Figure 7.1. The type of delivery vehicle utilised, the sequence of stores visited, the service start times at these stores (measured in minutes after 08:00), the utilisation of the delivery vehicle, the distance of the route (in kilometres), and the driver-route familiarity associated with each route are given.*

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Familiarity
First decision period (the planning horizon 13/12/2021–19/12/2021)					
Large	(44, 87, 63)	(0, 148.98, 279.45, 405.11, 696.65)	0.9614	629.87	0.7285
Large	(42, 84, 62)	(0, 202.23, 257.4, 366.63, 575.92)	0.9222	512.98	0.8634
Large	(86, 25)	(0, 60, 135.75, 229.27)	0.9141	11.39	0.9151
Large	(80, 23, 37)	(0, 153.92, 205.47, 275.2, 480.57)	0.9112	406.21	0.9965
Large	(55, 36)	(0, 60, 187.01, 302.02)	0.8802	120.49	0.6200
Large	(1, 74)	(0, 60, 131.35, 255.8)	0.8344	113.46	0.4962
Large	(26, 91, 7)	(0, 222.03, 283.87, 468.4, 670.28)	0.8329	671.32	0.2768
Large	(57, 4)	(0, 150.12, 207.82, 431.52)	0.7332	432.16	1.0000
Large	(2, 8)	(0, 60, 139.59, 213.16)	0.7248	37.52	0.9779
Large	(51, 11)	(0, 60, 108.66, 192.76)	0.6785	11.23	0.9780
Small	(59)	(0, 184.65, 451.19)	0.9747	512.24	1.0000
Small	(34)	(0, 93.77, 270.13)	0.9436	239.09	1.0000
Small	(70)	(0, 61.39, 200.06)	0.8856	148.51	1.0000
Small	(9)	(0, 206.95, 487.89)	0.8437	614.85	1.0000
Small	(43)	(0, 139.71, 353.48)	0.8366	404.74	1.0000
Small	(13)	(0, 134.02, 337.79)	0.8050	305.44	1.0000

Continued on next page

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Familiarity
Small	(28)	(0, 60, 148.75)	0.7948	35.93	1.0000
Small	(3)	(0, 203.86, 477.67)	0.7814	610.93	1.0000
Small	(21)	(0, 93.85, 256.44)	0.7670	238.86	1.0000
Small	(68)	(0, 153.94, 375.43)	0.7544	442.37	1.0000
Small	(60)	(0, 102.38, 270.37)	0.7424	250.14	1.0000
Small	(88)	(0, 108.64, 276.28)	0.6491	310.71	1.0000
Small	(29)	(0, 61.03, 178.82)	0.6126	148.24	1.0000
Small	(22)	(0, 60, 170.19)	0.5991	163.41	1.0000
Small	(61)	(0, 120.44, 294.22)	0.5641	364.96	1.0000
Second decision period (the planning horizon 13/12/2021–19/12/2021)					
Large	(41, 15, 45)	(0, 60, 139.47, 206.51, 283.64)	0.9917	42.25	0.5463
Large	(66, 56, 42)	(0, 60, 265.34, 535.45, 790.3)	0.9906	609.97	0.4278
Large	(18, 84, 58)	(0, 107.68, 314.31, 492.48, 652.28)	0.9718	580.56	0.5279
Large	(40, 28)	(0, 60, 143.27, 233.93)	0.8505	37.22	0.9751
Large	(20, 50)	(0, 78.95, 163.16, 307.58)	0.8429	214.52	0.9995
Large	(72, 85)	(0, 224.15, 295.89, 566.26)	0.6400	611.44	0.9981
Large	(71, 7)	(0, 137.37, 190.4, 392.29)	0.6399	371.60	1.0000
Small	(90)	(0, 61.91, 205.95)	0.9493	148.88	1.0000
Small	(21)	(0, 93.85, 256.44)	0.7670	238.86	1.0000
Small	(24)	(0, 168.82, 405.63)	0.7458	506.20	1.0000
Small	(12)	(0, 141.08, 344.78)	0.6835	404.67	1.0000
Small	(57)	(0, 136.47, 332.78)	0.6356	432.19	1.0000
Small	(31)	(0, 60, 158.55)	0.5942	113.29	1.0000
Small	(16)	(0, 168.42, 390.22)	0.5558	505.69	1.0000
Small	(64)	(0, 173.76, 395.68)	0.5072	485.35	1.0000
Third decision period (the planning horizon 13/12/2021–19/12/2021)					
Large	(49, 52, 22)	(0, 60.73, 117.2, 186.56, 302.27)	0.9739	164.47	0.9914
Large	(15, 19, 31)	(0, 60, 167.26, 224.79, 327.73)	0.9650	120.80	0.6087
Large	(36, 27, 17)	(0, 60, 239.36, 368.35, 647.73)	0.9607	623.19	0.5794
Large	(93, 30, 6)	(0, 60, 98.67, 198.09, 294.28)	0.9012	121.10	0.4603
Large	(2, 28)	(0, 60, 139.06, 229.72)	0.8422	37.08	0.9857
Large	(41, 66)	(0, 60, 120.85, 188.47)	0.6478	10.40	0.9784
Large	(10, 60)	(0, 112.17, 161.9, 340.11)	0.6286	250.24	0.9991
Small	(39)	(0, 139.8, 359.16)	0.9116	404.03	1.0000
Small	(69)	(0, 93.98, 263.99)	0.8573	239.35	1.0000
Small	(46)	(0, 178.77, 430.53)	0.8307	495.40	1.0000
Small	(13)	(0, 134.02, 337.79)	0.8050	305.44	1.0000
Small	(3)	(0, 203.86, 477.67)	0.7814	610.93	1.0000
Small	(56)	(0, 132.44, 331.29)	0.7590	304.76	1.0000
Small	(35)	(0, 140.86, 349.11)	0.7509	404.55	1.0000
Small	(62)	(0, 130.67, 326.89)	0.7407	373.97	1.0000
Small	(81)	(0, 60, 165.69)	0.7120	113.36	1.0000
Small	(45)	(0, 60, 135.2)	0.6120	36.78	1.0000
Small	(32)	(0, 60, 152.36)	0.5909	119.34	1.0000

Continued on next page

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Famil-liarity
Small	(47)	(0, 92.91, 240.04)	0.5653	238.12	1.0000
Small	(48)	(0, 120.29, 291.45)	0.5319	364.99	1.0000
Fourth decision period (the planning horizon 13/12/2021–19/12/2021)					
Large	(85, 92, 38)	(0, 222.53, 275.98, 338.72, 630.95)	0.9529	614.60	0.9928
Large	(83, 27, 71)	(0, 150.78, 220.78, 335.0, 526.97)	0.9321	479.40	0.8388
Large	(32, 77, 65)	(0, 60, 159.54, 202.47, 284.2)	0.9182	126.79	0.5172
Large	(76, 33, 47)	(0, 147.9, 197.59, 314.87, 471.47)	0.9086	326.83	0.8315
Large	(78)	(0, 60, 153.55)	0.4098	37.96	1.0000
Small	(29, 67)	(0, 61.03, 156.93, 250.39)	0.9887	204.22	0.3633
Small	(79)	(0, 261.23, 591.64)	0.7768	785.30	1.0000
Small	(1)	(0, 60, 169.77)	0.7569	112.55	1.0000
Small	(54)	(0, 234.85, 533.94)	0.7129	706.11	1.0000
Small	(14)	(0, 60, 119.14)	0.5344	10.60	1.0000
Small	(48)	(0, 120.29, 291.45)	0.5319	364.99	1.0000
Small	(10)	(0, 101.97, 252.9)	0.5148	249.87	1.0000
Small	(64)	(0, 173.76, 395.68)	0.5072	485.35	1.0000
Fifth decision period (the planning horizon 13/12/2021–19/12/2021)					
Large	(12, 26, 17)	(0, 155.18, 291.89, 349.75, 629.12)	0.9522	611.14	0.8299
Large	(14, 52, 49)	(0, 60, 169.38, 237.21, 352.07)	0.9416	152.21	0.5674
Large	(76, 73, 34)	(0, 147.9, 246.69, 290.21, 476.13)	0.9398	326.74	0.8332
Large	(75, 51, 65)	(0, 60, 99.37, 149.28, 231.01)	0.8466	11.64	0.9253
Large	(82, 61)	(0, 183.35, 315.67, 501.59)	0.7595	470.51	0.3878
Large	(8, 2)	(0, 60, 114.95, 212.91)	0.7248	36.89	0.9808
Small	(53)	(0, 60, 145.48)	0.9696	2.06	1.0000
Small	(11)	(0, 60, 143.19)	0.8542	11.00	1.0000
Small	(4)	(0, 136.52, 346.45)	0.8309	432.16	1.0000
Small	(24)	(0, 168.82, 405.63)	0.7458	506.20	1.0000
Small	(33)	(0, 133.92, 332.88)	0.7418	305.40	1.0000
Small	(50)	(0, 72.17, 209.15)	0.7018	214.52	1.0000
Small	(5)	(0, 60, 137.56)	0.6434	36.78	1.0000
Small	(89)	(0, 60, 126.92)	0.6249	11.51	1.0000
Small	(19)	(0, 60, 160.6)	0.6224	113.52	1.0000
Small	(30)	(0, 60, 159.62)	0.6212	112.53	1.0000
Small	(16)	(0, 168.42, 390.22)	0.5558	505.69	1.0000
Small	(58)	(0, 98.09, 247.91)	0.5325	306.50	1.0000
Small	(37)	(0, 140.52, 331.74)	0.5302	404.65	1.0000

Table 7.6: *Delivery routes of the solution selected for the third planning horizon (20/12/2021–26/12/2021), namely the third circled solution in Figure 7.1. The type of delivery vehicle utilised, the sequence of stores visited, the service start times at these stores (measured in minutes after 08:00), the utilisation of the delivery vehicle, the distance of the route (in kilometres), and the driver-route familiarity associated with each route are given.*

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Familiarity
First decision period (the planning horizon 20/12/2021–26/12/2021)					
Large	(21, 13)	(0, 103.23, 232.96, 457.1)	0.8991	326.51	0.8338
Large	(1, 93)	(0, 60, 138.19, 262.1)	0.8728	113.66	0.9852
Large	(2, 78)	(0, 60, 142.26, 235.77)	0.8644	38.36	0.9688
Large	(47, 73)	(0, 102.2, 184.98, 357.65)	0.8456	239.54	0.4971
Large	(28, 52)	(0, 60, 206.88, 332.89)	0.8299	189.87	0.0955
Large	(7, 71)	(0, 137.37, 217.83, 419.82)	0.8234	371.60	1.0000
Large	(66, 75)	(0, 60, 136.82, 212.61)	0.8187	10.30	1.0000
Large	(24, 35)	(0, 185.7, 307.23, 521.11)	0.8103	506.52	0.8986
Large	(91, 85)	(0, 227.6, 294.61, 583.85)	0.7153	614.92	0.4959
Large	(12)	(0, 155.18, 387.71)	0.4389	404.67	1.0000
Small	(4)	(0, 136.52, 348.46)	0.8576	432.16	1.0000
Small	(3)	(0, 203.86, 482.08)	0.8402	610.93	1.0000
Small	(22)	(0, 60, 186.8)	0.8206	163.41	1.0000
Small	(55)	(0, 60, 145.19)	0.7102	37.13	1.0000
Small	(60)	(0, 102.38, 265.55)	0.6783	250.14	1.0000
Small	(64)	(0, 173.76, 404.53)	0.6253	485.35	1.0000
Small	(39)	(0, 139.8, 337.57)	0.6238	404.03	1.0000
Second decision period (the planning horizon 20/12/2021–26/12/2021)					
Large	(31, 82, 62)	(0, 60, 264.4, 378.03, 587.47)	0.9971	500.92	0.4867
Large	(17, 27, 50)	(0, 222.53, 363.56, 496.48, 640.41)	0.9762	627.45	0.6573
Large	(19, 30)	(0, 60, 143.58, 252.17)	0.8120	113.62	0.9948
Large	(44, 56)	(0, 148.98, 231.66, 439.49)	0.8064	307.30	0.9964
Large	(4, 84)	(0, 150.17, 382.52, 645.56)	0.7630	651.54	0.3317
Large	(51, 28)	(0, 60, 140.51, 240.34)	0.7609	42.71	0.5458
Large	(49, 8)	(0, 60.73, 189.92, 283.06)	0.7511	189.95	0.5243
Large	(55, 65)	(0, 60, 150.78, 227.86)	0.7288	44.54	0.5426
Large	(18, 68)	(0, 107.68, 242.64, 466.94)	0.6706	444.71	0.8418
Large	(74, 76)	(0, 60, 229.32, 434.22)	0.6436	308.53	0.4954
Large	(7)	(0, 137.37, 356.77)	0.4698	371.60	1.0000
Large	(40)	(0, 60, 161.57)	0.4463	37.13	1.0000
Large	(67)	(0, 61.48, 171.81)	0.2636	163.41	0.0000
Small	(45)	(0, 60, 163.84)	0.9938	36.78	1.0000
Small	(24)	(0, 168.82, 423.48)	0.9837	506.20	1.0000
Small	(77)	(0, 60, 151.71)	0.9557	11.51	1.0000
Small	(63)	(0, 202.83, 485.33)	0.9244	610.20	1.0000
Small	(2)	(0, 60, 157.53)	0.9096	36.78	1.0000
Small	(33)	(0, 133.92, 343.41)	0.8822	305.40	1.0000
Small	(57)	(0, 136.47, 350.12)	0.8669	432.19	1.0000
Small	(34)	(0, 93.77, 254.77)	0.7388	239.09	1.0000

Continued on next page

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Familliarity
Small	(16)	(0, 168.42, 403.63)	0.7346	505.69	1.0000
Small	(79)	(0, 261.23, 585.3)	0.6923	785.30	1.0000
Small	(35)	(0, 140.86, 340.55)	0.6368	404.55	1.0000
Small	(29)	(0, 61.03, 180.3)	0.6322	148.24	1.0000
Small	(61)	(0, 120.44, 298.46)	0.6207	364.96	1.0000
Small	(83)	(0, 137.08, 324.5)	0.5133	432.61	1.0000
Third decision period (the planning horizon 20/12/2021–26/12/2021)					
Large	(25, 82, 42)	(0, 60, 293.45, 393.56, 661.7)	0.9365	549.18	0.4751
Large	(38, 3)	(0, 223.6, 306.44, 605.17)	0.8970	611.05	0.9996
Large	(33, 21)	(0, 147.32, 274.73, 456.76)	0.8908	326.32	0.8336
Large	(27, 87, 83)	(0, 149.55, 205.7, 261.91, 463.23)	0.8258	433.66	0.4988
Large	(14, 86)	(0, 60, 144.29, 202.41)	0.7257	11.60	0.9204
Large	(10, 32)	(0, 112.17, 283.1, 387.97)	0.7245	284.13	0.6487
Large	(28, 6)	(0, 60, 141.94, 202.54)	0.6542	36.69	0.4940
Large	(68, 80)	(0, 169.33, 403.97, 616.99)	0.6108	635.25	0.6669
Large	(18)	(0, 107.68, 284.06)	0.3779	306.40	1.0000
Large	(48)	(0, 132.32, 329.33)	0.3574	364.99	1.0000
Small	(11)	(0, 60, 152.18)	0.9741	11.00	1.0000
Small	(2)	(0, 60, 157.53)	0.9096	36.78	1.0000
Small	(13)	(0, 134.02, 344.81)	0.8987	305.44	1.0000
Small	(16)	(0, 168.42, 403.63)	0.7346	505.69	1.0000
Small	(60)	(0, 102.38, 265.55)	0.6783	250.14	1.0000
Small	(5)	(0, 60, 138.44)	0.6552	36.78	1.0000
Small	(46)	(0, 178.77, 415.78)	0.6341	495.40	1.0000
Small	(88)	(0, 108.64, 267.09)	0.5267	310.71	1.0000
Small	(70)	(0, 61.39, 172.17)	0.5137	148.51	1.0000
Fourth decision period (the planning horizon 20/12/2021–26/12/2021)					
Large	(50, 30, 53)	(0, 79.38, 184.57, 293.03, 351.61)	0.9930	217.31	0.4968
Large	(67, 52, 49)	(0, 61.48, 114.24, 181.44, 302.59)	0.9757	164.53	0.4925
Large	(76, 87, 56)	(0, 147.9, 274.49, 395.85, 603.67)	0.9544	446.30	0.6838
Large	(2, 15)	(0, 60, 144.42, 232.62)	0.8416	40.12	0.9262
Large	(34, 47)	(0, 103.15, 172.04, 354.17)	0.8222	239.15	0.9977
Large	(80, 72)	(0, 153.92, 286.35, 594.92)	0.8106	611.24	0.8309
Large	(56, 13)	(0, 145.68, 210.11, 434.24)	0.8007	305.44	0.9989
Large	(12, 43)	(0, 155.18, 235.51, 449.87)	0.7671	406.05	0.9965
Large	(1, 5)	(0, 60, 176.14, 256.51)	0.7499	120.02	0.6194
Large	(34, 90)	(0, 103.15, 207.34, 339.9)	0.7256	239.62	0.8093
Large	(61, 58)	(0, 132.48, 219.66, 397.22)	0.6950	367.37	0.9139
Large	(26, 84)	(0, 222.03, 520.64, 783.68)	0.6859	839.28	0.0000
Large	(9, 85)	(0, 227.65, 283.18, 572.42)	0.6382	614.95	0.9958
Large	(69, 29)	(0, 103.38, 200.12, 325.57)	0.6321	239.67	0.8084
Large	(39)	(0, 153.78, 365.65)	0.3119	404.03	1.0000

Continued on next page

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Familiarity
Small	(11)	(0, 60, 152.18)	0.9741	11.00	1.0000
Small	(19)	(0, 60, 184.47)	0.9406	113.52	1.0000
Small	(41)	(0, 60, 145.34)	0.8966	10.06	1.0000
Small	(40)	(0, 60, 159.33)	0.8925	37.13	1.0000
Small	(66)	(0, 60, 145.73)	0.8909	10.30	1.0000
Small	(20)	(0, 71.77, 221.82)	0.8709	214.52	1.0000
Small	(57)	(0, 136.47, 350.12)	0.8669	432.19	1.0000
Small	(36)	(0, 60, 177.3)	0.8621	112.32	1.0000
Small	(4)	(0, 136.52, 348.46)	0.8576	432.16	1.0000
Small	(37)	(0, 140.52, 355.22)	0.8433	404.65	1.0000
Small	(3)	(0, 203.86, 482.08)	0.8402	610.93	1.0000
Small	(81)	(0, 60, 174.57)	0.8304	113.36	1.0000
Small	(22)	(0, 60, 186.8)	0.8206	163.41	1.0000
Small	(15)	(0, 60, 146.36)	0.7735	35.54	1.0000
Small	(58)	(0, 98.09, 265.67)	0.7693	306.50	1.0000
Small	(42)	(0, 183.85, 433.71)	0.7616	511.81	1.0000
Small	(65)	(0, 60, 136.08)	0.7475	11.52	1.0000
Small	(10)	(0, 101.97, 269.93)	0.7420	249.87	1.0000
Small	(16)	(0, 168.42, 403.63)	0.7346	505.69	1.0000
Small	(32)	(0, 60, 161.07)	0.7070	119.34	1.0000
Small	(60)	(0, 102.38, 265.55)	0.6783	250.14	1.0000
Small	(43)	(0, 139.71, 339.97)	0.6564	404.74	1.0000
Small	(59)	(0, 184.65, 425.13)	0.6272	512.24	1.0000
Fifth decision period (the planning horizon 20/12/2021–26/12/2021)					
Large	(70, 90, 15)	(0, 67.53, 116.63, 260.39, 348.59)	0.9998	180.22	0.5146
Large	(17, 92, 9)	(0, 222.53, 290.15, 355.77, 633.5)	0.9771	614.98	0.9957
Large	(36, 41)	(0, 60, 182.94, 269.09)	0.8794	118.25	0.5175
Large	(8, 2)	(0, 60, 134.52, 233.97)	0.8653	36.89	0.9808
Large	(52, 14)	(0, 61.47, 186.28, 277.11)	0.8438	152.58	0.0346
Large	(65, 1)	(0, 60, 176.2, 296.84)	0.7960	119.83	0.5188
Large	(26, 37)	(0, 222.03, 358.45, 587.3)	0.7733	610.69	0.3313
Large	(21, 69)	(0, 103.23, 182.86, 345.55)	0.7658	239.57	0.9981
Large	(53, 31)	(0, 60, 163.16, 271.36)	0.6357	113.80	0.5052
Large	(88, 62)	(0, 119.5, 194.87, 404.31)	0.6347	374.18	0.9149
Small	(45)	(0, 60, 163.84)	0.9938	36.78	1.0000
Small	(54)	(0, 234.85, 554.98)	0.9935	706.11	1.0000
Small	(23)	(0, 140.0, 364.24)	0.9776	404.09	1.0000
Small	(11)	(0, 60, 152.18)	0.9741	11.00	1.0000
Small	(12)	(0, 141.08, 359.36)	0.8779	404.67	1.0000
Small	(20)	(0, 71.77, 221.82)	0.8709	214.52	1.0000
Small	(4)	(0, 136.52, 348.46)	0.8576	432.16	1.0000
Small	(78)	(0, 60, 151.51)	0.8191	37.96	1.0000
Small	(86, 89)	(0, 60, 110.35, 153.4)	0.8135	11.74	0.9331
Small	(62)	(0, 130.67, 327.04)	0.7428	373.97	1.0000

Continued on next page

Vehicle type	Route	Arrival times	Vehicle utilisation	Distance travelled	Famil-liarity
Small	(10)	(0, 101.97, 269.93)	0.7420	249.87	1.0000
Small	(48)	(0, 120.29, 305.17)	0.7148	364.99	1.0000
Small	(71)	(0, 124.88, 314.23)	0.7073	371.60	1.0000
Small	(50)	(0, 72.17, 208.67)	0.6954	214.52	1.0000
Small	(79)	(0, 261.23, 585.3)	0.6923	785.30	1.0000
Small	(17)	(0, 202.3, 466.58)	0.6895	609.83	1.0000
Small	(74)	(0, 60, 160.76)	0.6519	111.22	0.0000
Small	(46)	(0, 178.77, 415.78)	0.6341	495.40	1.0000
Small	(59)	(0, 184.65, 425.13)	0.6272	512.24	1.0000
Small	(64)	(0, 173.76, 404.53)	0.6253	485.35	1.0000
Small	(25)	(0, 60, 117.28)	0.5240	10.19	1.0000

7.3 Discussion

From the results obtained during the case study for the EPH depot of the industry partner, it is concluded that the FVRP and its approximate solution approaches proposed in this thesis are able to return high-quality solutions to a real-world problem instance within an acceptable time-frame. For each of the planning horizons, multiple non-dominated solutions were returned, including solutions corresponding to relatively high degrees of driver-route familiarity.

Upon consideration of all three planning horizons and their varying demand, driver-route familiarity could best be achieved for the second planning horizon during which the demand volumes exhibited by stores were most similar to the average demand volumes. For the second planning horizon, the largest increase in familiarity could be obtained for the smallest percentage increase in transportation cost. For the first and third planning horizons during which stores exhibited larger demand variations, good driver-route familiarity could also be achieved, although for a higher increase in transportation cost.

The total number of visits required by stores during a planning horizon had a significant effect on the performance of the approximate solution approach for the operational phase of the FVRP. Fewer iterations could be completed for planning horizons during which a larger total number of visits were required. Since demand planning at the industry partner is performed well in advance of the start of a week, it is recommended that delivery routes be computed more than a week in advance for weeks requiring large total numbers of visits. This will allow for a longer run time of the approximate solution approach for the operational phase of the FVRP, resulting in a larger number of algorithmic iterations and more solutions of high quality.

It is finally worth mentioning that an efficient implementation of the proposed approximate solution approaches in a different programming language and by a professional programmer may be able to perform better within the same time-frame than the author's implementation in Python. Furthermore, the use of specialised and higher-capability computers will also improve the performance of the approximate solution approaches within the same time-frame.

7.4 Expert validation

The results obtained during the case study were presented to a *subject matter expert* (SME) employed by the industry partner attached to this thesis, Dr Jonas Stray [126], in pursuit of expert validation as well as recommendations as to possible improvements on the modelling process followed and the results obtained. As part of the presentation, the current logistical operations at the industry partner were presented in order to receive confirmation of the process described in §6.1. The SME confirmed that the current logistical operations described in §6.1 are correct and that the input parameters to the FVRP models (such as the number of delivery vehicles available and the number of decision periods within a planning horizon) based thereon were also correct.

The FVRP proposed in this thesis was explained to the SME and the results for the strategic and operational phases of the case study were presented to him. As first reaction to the results represented, the SME was impressed by the quality of the results and the ability of the FVRP proposed in this thesis and its accompanying approximate solution approaches in terms of recommending alternatives which incorporate driver-route familiarity into delivery routes, based on real-world data. The familiarity objective function values achieved in each set of non-dominated solutions were deemed satisfactory. Upon further investigation of the routes of each selected trade-off solution, the SME stated that these routes would seem appropriate for implementation at the industry partner's EPH depot and that the utilisations of delivery vehicles in the solutions in Tables 7.4–7.6 are similar to those of their current implementations. The SME also reiterated the significance of the problems that occur currently when implementing delivery routes based on the minimisation of cost only. These problems include service level requirements of stores often not being met and additional delivery vehicles having to be utilised when planned delivery routes are not performed as planned. Furthermore, smaller stores often struggle to unpack large deliveries if maximum volume thresholds for deliveries per day are exceeded. As final remarks on the results, the SME stated that the proposed FVRP and approximate solution approaches are expected to be a valuable tool to have and that the multi-objective approach adopted is preferred, since the cost of familiarity can easily be quantified.

The SME stated that the computational time of the solution approaches employed are not problematic and that a more efficient implementation thereof can easily be achieved through parallelisation of the solution approaches as well as by the use of efficient packages in the programming language Python, such as NumPy, for performing operations on arrays. The run time duration was not as important to the SME as the implementation of the concept of creating driver-route familiarity.

As a final remark, the SME mentioned an additional problem that often arises during the implementation of planned delivery routes, namely that drivers are not only often unfamiliar with the routes along which they travel, but also sometimes with the stores that they visit. This often leads to drivers not knowing the exact drop-off locations or delivery processes followed at specific stores. It also sometimes leads to inefficient communication if the staff at stores are not familiar with the drivers carrying out their deliveries.

7.5 Chapter Summary

This chapter was devoted to a presentation and analysis of the results obtained during a case study based on real-world data. The master routes computed during the strategic phase of the FVRP for the EPH depot of the industry partner attached to this thesis, were presented in §7.1. This was followed in §7.2 by a discussion on the corresponding solutions obtained during the operational phase of the FVRP for three planning horizons. A discussion on the results obtained during the case study was presented in §7.3, and this was followed in §7.4 by an expert validation of the results by an employee of the industry partner.

Part IV

Conclusion

CHAPTER 8

Thesis summary and appraisal

Contents

This chapter serves as a conclusion to the work reported in this thesis. Each chapter of this thesis is summarised in §8.1 after which an appraisal of the contributions made in this thesis is provided in §8.2.

8.1 Thesis summary

Apart from the stand-alone Chapter 1 and the current Part IV, the other six chapters of thesis were partitioned into three parts, titled *Literature review* (Chapters 2 and 3), *Mathematical models* (Chapters 4 and 5), and *Validation case study* (Chapters 6 and 7).

Chapter 1 served as an introduction to the VRP and the problems that arise when implementing solutions that stem from solving VRP instances. The chapter opened in §1.1 with a discussion on the supply chain of retail organisations and highlighted the importance of the VRP within these supply chains. This was followed by an introduction to the basic VRP and some of its most common variants applicable to the problem considered in this thesis, after which the difficulty of solving VRP instances was highlighted. In §1.2, the industry partner attached to this thesis was introduced, and this was followed by a discussion on the practical problems experienced by the industry partner when implementing solutions that stem from solving VRP instances by means of standard commercial software. This discussion served as a motivational argument for the informal problem description of this thesis, presented in §1.3, namely to propose a novel VRP for improved delivery routing which addresses the practical limitations derived from a lack of driver-route familiarity prevalent in standard VRP formulations. After having presented the problem statement, the eight research objectives pursued in this thesis were outlined in §1.4. This was followed by a delimitation of the scope of the research carried out in this thesis, pertaining to the model solution methods adopted, the input data and input parameters to the proposed mathematical models, and the computer implementations of the solution approaches. The chapter closed in §1.5 and §1.6 with a description of the manner in which the research would be carried out and would be reported in this thesis.

Part I comprised of two chapters, Chapters 2 and 3, and was devoted to a review of the literature relevant to the research carried out in this thesis, in fulfilment of Objective I of §1.4. Chapter 2 was devoted to a discussion on various VRP variants, in fulfilment of Objective I(a), based on a taxonomy proposed by Toth and Vigo [137]. A brief history of the VRP, as well as an overview of the aforementioned taxonomy, was given in §2.1. VRP variants arising from different network structures, transportation requests, intra-route constraints, inter-route constraints, fleet characteristics, and types of objectives were discussed in §2.2–§2.7, respectively.

Chapter 3 was devoted to a discussion on three algorithmic solution methodologies for solving VRPs, namely exact, heuristic, and metaheuristic solution approaches, in fulfilment of Objectives I(b) and I(c). The chapter opened in §3.1 with a prerequisite discussion on the simplex algorithm for solving linear programming problems, upon which all three exact solution approaches discussed thereafter are based. This was followed by a discussion on the branch-and-bound method, the cutting plane method, and the branch-and-cut method for solving IP problem instances. Classifications of heuristic and metaheuristic solution methods typically used to solve VRP instances were discussed in §3.2 and §3.3, respectively, and this was followed in §3.4 by a comparison of metaheuristics in the context of solving VRPs. The working of the HGSADC algorithm, a state-of-the-art solution methodology employed in this thesis, was described in §3.5, after which a discussion followed in §3.6 on approaches that have been adopted in the literature for solving multi-objective optimisation problems approximately.

Part II of this thesis comprised two further chapters, Chapters 4 and 5, and was devoted to the proposal of two novel mathematical models which form the working basis of the proposed FVRP, in fulfilment of Objectives II–V. In Chapter 4, a mathematical model was proposed for the strategic phase of the FVRP, in fulfilment of Objective II(a). The model was derived in §4.1 after which an exact solution approach was proposed in §4.2 and implemented in the CPLEX optimisation environment *via* its Python interface. This was followed by a systematic model verification in §4.3, in partial fulfilment of Objective III. Upon an empirical analysis of the time complexity of the model in §4.4, it was found that the exact solution approach is not scalable to the size of real-world problem instances. An approximate solution approach was therefore proposed in §4.5, based on the HGSADC algorithm of Vidal *et al.* [140], in partial fulfilment of Objective IV. The proposed approximate solution approach was implemented in Python in partial fulfilment of Objective V and a systematic verification of this model solution approach followed in §4.6 during which it was found that it is acceptable and that it is capable of returning high-quality solutions to randomly generated test instances.

In Chapter 5, a mathematical model was proposed for the operational phase of the FVRP, in fulfilment of Objective II(b). The model was derived in §5.1 after which an exact model solution approach was proposed in §5.2 and implemented in the CPLEX optimisation environment *via* its Python interface. This was followed by a systematic model verification in §5.3, in final fulfilment of Objective III. Upon an empirical analysis of the time complexity of the exact solution approach for the model in §5.4, it was again found that the exact solution approach is not scalable to the size of real-world problem instances. An approximate solution approach was therefore proposed in §5.5, also based on the HGSADC algorithm of Vidal *et al.* [140] as well as on solution evaluation techniques proposed by Deb *et al.* [38], in final fulfilment of Objective IV. The proposed approximate solution approach was also implemented in Python, in final fulfilment of Objective V. A systematic verification of the approximate solution approach followed in §5.6, during which it was found that it is acceptable and that it is capable of returning high-quality sets of non-dominated solutions within an acceptable time-frame.

Part III of the thesis also comprised two chapters, Chapters 6 and 7, which were collectively dedicated to a real-world validation case study of the FVRP in the operational context of the industry partner attached to this thesis, in fulfilment of Objectives VI and VII. The purpose of Chapter 6 was to provide background information on the industry partner attached to this thesis, and to present the input data related to the case study performed in Chapter 7. The industry partner and its current logistical operations were discussed briefly in §6.1, and this was followed in §6.2 by a discussion on the real-world input data related to the case study performed.

Chapter 7 was devoted to a presentation and analysis of the results obtained during the case study. The master routes computed during the strategic phase of the FVRP were presented in §7.1 for one of the depots of the industry partner. This was followed in §7.2 by a discussion on the corresponding solutions obtained during the operational phase of the FVRP for three one-week planning horizons, as well as on trade-off solutions selected for each planning horizon. A discussion on the quality of results obtained during the case study was presented in §7.3, in partial fulfilment of Objective VII. The process followed during the case study and the results obtained were presented to an SME employed by the industry partner, in final fulfilment of VII. As discussed in §7.4, the SME validated the results of the case study.

An appraisal of the thesis contributions are presented in the remainder of this chapter, whereas suggestions for potential future follow-up work are proposed in the next chapter, in fulfilment of Objective VIII.

8.2 Appraisal of thesis contributions

The contributions of this thesis are six-fold. Each of these contributions is stated and elucidated in this section.

Contribution I *A revised taxonomy of metaheuristics for solving VRPs*

According to a taxonomic review of the VRP literature published between the beginning of 2009 and the middle of 2015, 70% of the articles reviewed employed metaheuristics as a proposed solution approach [16]. New classes of metaheuristics are still being proposed today, resulting in the introduction of new solution methods for solving VRP instances. The literature review of §3.3 contains a revised classification scheme for metaheuristics in the context of solving VRPs. This classification scheme is an amalgamation of multiple classification schemes found in the literature and includes relatively new methods employed for solving VRPs.

More specifically, the literature reviewed in §3.3 opened with a definition of the notion of a metaheuristic, as well as a brief discussion on the advantages associated with employing such a method. The taxonomy started with an explanation of the differences between trajectory-based metaheuristics and population-based metaheuristics. This was followed by a brief explanation of eight classes of trajectory-based metaheuristics. Population-based metaheuristics were further classified into evolutionary and swarm intelligence metaheuristics. Ten evolutionary metaheuristic classes and four swarm intelligence metaheuristic classes were discussed. For each of the above-mentioned classes, a brief discussion as well as a reference to an important or successful application of the metaheuristic was provided in the context of the VRP.

Contribution II *The proposal of a new FVRP for creating driver-route familiarity*

The FVRP proposed in Part II of this thesis, is aimed at improving the practical implementation of planned vehicle delivery routes in the retail sector by addressing the lack of driver-route familiarity prevalent in standard VRP formulations. First, a mathematical model was proposed in Chapter 4 for computing high-quality delivery routes, called master routes, with which delivery vehicle drivers may become familiar in the future. User inputs to this model, such as the number of overlapping master route arcs allowed without penalty and the penalty coefficient for penalising the number of overlapping arcs above this maximally allowed threshold, allows the analyst to generate master routes based on the preferences and operations of the user organisation. By adjusting these parameters, the number of master route arcs generated may be decreased (thereby yielding fewer arcs with which drivers should become familiar and therefore allowing for better familiarity with these arcs) or it may be increased (allowing for more master route arcs and therefore more freedom when computing actual delivery routes in the future, resulting in a possibly lowered cost for increasing driver-route familiarity).

Secondly, a mathematical model was proposed in Chapter 5 for the operational phase of the FVRP in the form of a bi-objective optimisation problem capable of returning possibly many high-quality trade-off solutions with increasing driver-route familiarity to choose from. The bi-objective modelling approach furthermore allows the user organisation to quantify the trade-off cost of increasing driver-route familiarity.

Finally, although the proposed FVRP only takes certain VRP attributes into consideration (such as customer time-windows and multiple planning periods), the structure of the FVRP is generic in nature and can easily be applied to VRPs exhibiting other attributes such as, for example, multiple depots and/or multiple trips. This allows for any retail organisation to apply the concept of the FVRP, regardless of their operational requirements.

Contribution III *The proposal of a novel HGSADC implementation for solving realistically sized instances of the model for the strategic phase of the FVRP*

The state-of-the art HGSADC algorithm proposed, by Vidal *et al.* [140] for solving VRPs having multiple periods, multiple depots, and customer time-windows, or any combination of these attributes, was adapted and implemented for solving instances of the model for the strategic phase of the FVRP. Substantial adaptions were made to the working of the original algorithm so as to be able to accommodate instances of the model for the FVRP strategic phase, as discussed in §4.5. These adaptions included penalising overlapping master route arcs in the objective function values associated with solutions, changing the multiple depot attribute of the original HGSADC algorithm into a heterogeneous fleet attribute, allowing customers to receive multiple visits, and including an exact solution component (in other words creating a matheuristic). The proposed approximate solution approach for the model of the FVRP strategic phase was able to return high-quality solutions within an acceptable time-frame, as discussed in §4.6.

Contribution IV *The proposal of a novel HGSADC implementation (in conjunction with a non-dominated sorting procedure) for solving realistically sized instances of the model for operational phase of the FVRP*

The proposed approximate solution approach for the model of the FVRP operational phase is also based on the HGSADC algorithm proposed by Vidal *et al.* [140], but includes the non-dominated sorting procedure proposed by Deb *et al.* [38] for multi-objective optimisation problems. As in the case of the proposed approximate solution approach for the strategic phase of the FVRP, many adaptions were made to the original HGSADC algorithm, as discussed in §5.5. In order to accommodate bi-objective optimisation problem instances of the model for the operational phase of the FVRP, the non-dominated sorting procedure and a CDD measure were employed when evaluating solutions. This included additional penalty parameters for infeasibility required when evaluating solutions, as well as alternating between objectives during the local search procedure performed when executing the HGSADC algorithm. Furthermore, a procedure for adapting the master routes obtained during the strategic phase, in order to be feasible for the corresponding instance of the operational phase being solved, was proposed. This ensures that a relatively good feasible starting solution is included in the initial population when employing the approximate solution approach. The proposed approximate solution approach for the operational phase of the FVRP was aslo able to return high-quality solutions within an acceptable time-frame, as discussed in §5.6.

Contribution V *The proposal of a systematic mechanism for generating randomly generated benchmark instances of the FVRP*

Since the FVRP proposed in this thesis is a novel VRP, benchmark instances could not be found in the literature for testing and validating the mathematical models proposed for the problem. Two random problem instance generators were therefore created in order to generate instances of the models for the strategic phase and the operational phase of the FVRP, described in §4.3 and §5.3, respectively. An arbitrary number of test instances may thus be generated for the model for the strategic phase, after which the same instances, but with varying actual demand volumes, may be generated for an arbitrary number of planning horizons during the operational phase (representing future planning horizons). A set of input parameters to the problem instance generators allows for customisation of these instances. Both problem instance generators are available on the author's Github repository so as to facilitate future research on the FVRP.

Contribution VI *Application of the FVRP and proposed solution methodologies to a real-world case study as a proof of concept*

The practical applicability of the proposed FVRP, its constituent mathematical models, and their corresponding approximate solution approaches were validated in Part III of this thesis, in respect of a case study involving real-world data provided by the industry partner attached to the thesis. First, background information on the industry partner and the input data related to the case study were provided in Chapter 6. Differences between the aforementioned randomly generate problem instances and the input data to the case study were highlighted.

An analysis of the numerical results obtained during the case study followed in Chapter 7. The master routes computed for the EPH depot of the industry partner, based on the average demand volumes associated with stores over a 13-week period, were presented. These master routes were then provided as input to the operational phase model of the FVRP during which actual delivery routes were computed for three planning horizons, each representing a duration of a week (containing five days on which deliveries could take place). Furthermore, each of these three planning horizons was associated with different demand variations resulting in low, medium, and high demand volume planning horizons. The results obtained for the operational phase of the case study demonstrated that driver-route familiarity could be achieved for each of these three planning horizons.

Finally, the results obtained during the case study were validated by an SME employed by the industry partner. The case study therefore showcased the ability of the FVRP proposed in this thesis to create driver-route familiarity based on real-world data, as well as the ability of the proposed solution methods to be scalable to the size of real-world problem instances.

CHAPTER 9

Future work

Contents

Suggestions for future work related to the work reported in this thesis are presented in this chapter, in fulfilment of Objective VIII of §1.4. These suggestions are partitioned into three sections. Suggestions for future work related to the modelling approach adopted in this thesis are presented in §9.1. This is followed in §9.2 by suggestions for future work related to the approximate model solution approaches proposed in this thesis. Finally, a suggestion for future work related to the case study performed in Chapters 6 and 7 is discussed in §9.3.

9.1 Suggestions related to the modelling approach

This section contains three suggestions for future work related to the mathematical models employed in Chapters 4 and 5.

Proposal I *Taking into account the distance along the physical road network shared between arcs of the transportation graph*

In the current modelling approach proposed for the FVRP, the distances along road segments shared between arcs of the transportation graph are not taken into account when computing actual delivery routes during the operational phase of the FVRP. An arc that does not form part of the set of master route arcs may, for example, share a significant distance along the road network with arcs that form part of the set of master route arcs. A driver might therefore travel on a part of the road network that is also part of a master route arc, but this may not contribute to the driver-route familiarity associated with the route of the driver according to the modelling approach adopted in this thesis. The route driven by a vehicle might therefore, in reality, achieve a larger familiarity objective function value than actually measured mathematically according to the current modelling approach. By including the distance along the road network shared between arcs of the transportation graph adopted in the modelling approach, more accurate (and possibly larger) familiarity objective function values may be associated with solutions.

Proposal II *Including the familiarity of delivery vehicle drivers with customers*

As mentioned by the SME employed by the industry partner attached to this thesis, an additional problem that often arises during the implementation of planned delivery routes is that delivery vehicle drivers are not familiar with the customers they visit along their routes. This leads to drivers not being familiar with the exact drop-off locations at customers or the delivery processes followed by specific customers. Furthermore, it may lead to inefficient communication if the customer store is not familiar with the driver carrying out its deliveries. A useful addition to the FVRP and its models proposed in this thesis may therefore be not only to increase driver-route familiarity, but also to increase driver-customer familiarity. Once actual delivery routes have been computed, the routes may be assigned to drivers in such a way that driver-customer familiarity is prioritised instead of or in addition to maximising driver-route familiarity, subject to operational constraints.

Proposal III *Designing a generic framework for increased driver-route familiarity*

Since the FVRP proposed in this thesis consists of two phases, with the output of the first (the strategic phase) serving as input to the second (the operational phase), the problem lends itself to incorporation into a framework which also incorporates the demand planning aspects of the problem over future planning horizons. It may therefore be beneficial to propose a generic, overarching, model- and data-driven framework for creating driver-route familiarity in planned delivery routes. The generic nature of such a framework may aid any organisation in creating driver-route familiarity in planned delivery routes, regardless of the specific operations of the organisation. Any VRP attributes required by an organisation may be included in such a framework. Furthermore, such a framework may allow for the relatively straight-forward addition of new VRP features, such as the one discussed in Proposal II.

9.2 Suggestions related to the solution approaches

This section contains three further suggestions for future work, this time related to the solution approaches proposed in Chapters 4 and 5.

Proposal IV *The addition of a dynamic variable when invoking the split algorithm*

Recall, from §7.2, that during the execution of the operational phase of the case study, when actual delivery routes were computed for the third planning horizon (associated with high volumes of demand), a feasible solution could not be uncovered during the first day of run time. This was due to the split algorithm partitioning giant tour chromosomes into routes in such a manner that the cumulative load associated with each route did not exceed twice the capacity of the corresponding delivery vehicle. Only once this factor of 2 was changed so that routes are partitioned in such a manner that the load does not exceed 1.1 times the capacity of the associated delivery vehicle, could a feasible solution be uncovered. It is therefore proposed that this factor be changed to a dynamic value, based on the portion of feasible solutions uncovered during the search. This may direct the search towards feasible solutions at the start of execution of the search algorithm, after which the value may gradually be increased, thereby diversifying the search during later stages of execution of the algorithm.

Proposal V *Performing a parallelised implementation of the proposed solution approaches*

The implementations of the approximate solution approaches proposed for the strategic and operational phases of the FVRP were not parallelised during the research carried out towards this thesis, instead only making use of a single central processing unit in the computer on which it was executed. Most computers, however, have multiple central processing units and will therefore have additional idle computational capacity available when executing the approximate solution approaches. Furthermore, distributed computing, which includes the use of cloud computing or high performance computing clusters, such as the elastic container service provided by Amazon Web Services or the Rhasatsha cluster of Stellenbosch University, allows for the use of multiple processing units. A parallel implementation of the proposed solution approaches using any of these types of computing infrastructure is therefore expected to improve the run time significantly. This parallelisation may be applied to specific operations of the approximate solution approaches in such a way that multiple operations not requiring a specific sequence of performance, are carried out in parallel. This may, for example, occur when initialising many solutions or when decomposing the problem instance into smaller subproblem instances and then solving each subproblem instance in a distributed fashion. Furthermore, additional significant run-time improvement may be achieved by implementing the parallelised solution approaches in a lower-level programming language, such as *C*, which do not require the interpretation of programming code and are therefore generally known to result in better performance than when working in programming languages such as *Python*, which require interpretation.

Proposal VI *Including historical solutions in the initial population for the operational phase*

Recall, from §5.5.5, that in the proposed approximate solution approach for the operational phase of the FVRP, the master routes computed during the strategic phase of the FVRP were adapted to be feasible for the particular corresponding instance of the operational phase and inserted as a solution in the initial population before the start of the algorithm. This addition ensured that a relatively good solution is available among the initial population, which directed the search towards good feasible solutions from the start of the algorithmic execution. In the same manner, the solutions implemented for the same depot and its assigned customers during previous planning horizons may be adapted and inserted into the initial population for subsequent planning horizons. This will allow the approximate solution approach to start with relatively good feasible solutions at its disposal and may save computational time by "recycling" the actual delivery routes computed for previous planning horizons. This is expected to decrease the run time as well as increase the quality of solutions returned by the approximate solution approach for the operational phase of the FVRP. Such a feature may easily be included in the generic framework discussed in Proposal III.

9.3 A suggestion related to the case study performed

This section contains a final suggestion for future work related to the case study performed in Chapters 6 and 7.

Proposal VII *Performing additional case studies to evaluate the effect of different transportation network characteristics*

The characteristics of a transportation network are expected to have a marked effect on the effectiveness of the FVRP. Some characteristics of the transportation network of the case study carried out in this thesis, for example, were that the depot was not centred amongst the customers and that customers were located relatively far away from the depot in some cases, resulting in many delivery routes only visiting a single customer. By performing additional case studies for transportation networks exhibiting different characteristics, insight might be gained into the effects of these characteristics on the effectiveness of the FVRP. For example, case studies may be performed in urban and exurban contexts, and with different depot locations relative to the customers. Furthermore, the effect of different VRP attributes, such as the nature of customer time-windows and site-dependencies, may be analysed.

References

[1] Abbass HA, 2001, *Marriage in honey bees optimisation: A haplometrosis polygynous swarming approach*, Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, pp. 207–214.

[2] Applegate DL, Bixby RE, Chvátal V & Cook WJ, 2011, *The traveling salesman problem: A computational study*, Princeton University Press, Princeton (NJ).

[3] Araque JR, 1989, *Solution of a 48-city vehicle routing problem by branch-and-cut*, Unpublished Manuscript, Department of Aplied Mathematics and Statistics, State University of New York at Stony Brook, Brookhaven (NY).

[4] Archetti C, Feillet D, Hertz A & Speranza MG, 2009, *The capacitated team orienteering and profitable tour problems*, Journal of the Operational Research Society, **60(6)**, pp. 831–842.

[5] Archetti C, Hertz A & Speranza MG, 2007, *Metaheuristics for the team orienteering problem*, Journal of Heuristics, **13(1)**, pp. 49–76.

[6] Archetti C, Savelsbergh MW & Speranza MG, 2006, *Worst-case analysis for split delivery vehicle routing problems*, Transportation Science, **40(2)**, pp. 226–234.

[7] Archetti C & Speranza MG, 2012, *Vehicle routing problems with split deliveries*, International Transactions in Operational Research, **19(1-2)**, pp. 3–22.

[8] Avella P, Boccia M & Sforza A, 2004, *Resource constrained shortest path problems in path planning for fleet management*, Journal of Mathematical Modelling and Algorithms, **3(1)**, pp. 1–17.

[9] Baldacci R, Christofides N & Mingozzi A, 2008, *An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts*, Mathematical Programming, **115(2)**, pp. 351–385.

[10] Bansal JC, Singh PK & Pal NR, 2019, *Evolutionary and swarm intelligence algorithms*, Springer, New York (NY).

[11] Baxter J, 1981, *Local optima avoidance in depot location*, Journal of the Operational Research Society, **32(9)**, pp. 815–819.

[12] Beasley JE, 1983, *Route first—cluster second methods for vehicle routing*, International Journal of Management Sciences, **11(4)**, pp. 403–408.

[13] Bertsimas DJ, 1992, *A vehicle routing problem with stochastic demand*, Operations Research, **40(3)**, pp. 574–585.

[14] Bodin L, Maniezzo V & Mingozzi A, 2012, *Street routing and scheduling problems*, pp. 413–450 in Hall RW (Ed), *Handbook of transportation science*, Kluwer Academic Publishers, New York (NY).

[15] Boussaïd I, Lepagnot J & Siarry P, 2013, *A survey on optimisation metaheuristics*, Information Sciences, **237**, pp. 82–117.

[16] Braekers K, Ramaekers K & Van Nieuwenhuyse I, 2016, *The vehicle routing problem: State of the art classification and review*, Computers and Industrial Engineering, **99**, pp. 300–313.

[17] Bräysy O & Gendreau M, 2005, *Vehicle routing problem with time-windows, Part I: Route construction and local search algorithms*, Transportation Science, **39(1)**, pp. 104–118.

[18] Campbell AM, Clarke LW & Savelsbergh MW, 2002, *Inventory routing in practice*, pp. 309–330 in Toth P & Vigo D (Eds), *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[19] Chao I, 2002, *A tabu search method for the truck and trailer routing problem*, Computers and Operations Research, **29(1)**, pp. 33–51.

[20] Chartered Institute of Procurement and Supply, 2020, *What is a supply chain?*, [Online], [Cited March 2021], Available from `https://www.cips.org/knowledge/procurement-topics-and-skills/supply-chain-management/what-is-a-supply-chain/`.

[21] Chen A, Yang G & Wu Z, 2006, *Hybrid discrete particle swarm optimisation algorithm for capacitated vehicle routing problem*, Journal of Zhejiang University Science A, **7(4)**, pp. 607–614.

[22] Chen P, Huang H & Dong X, 2010, *Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem*, Expert Systems with Applications, **37(2)**, pp. 1620–1627.

[23] Christofides N & Eilon S, 1969, *An algorithm for the vehicle dispatching problem*, Journal of the Operational Research Society, **20(3)**, pp. 309–318.

[24] Christofides N, Mingozzi A & Toth P, 1979, *The vehicle routing problem*, pp. 315–338 in Christofides N, Mingozzi A & Toth P (Eds), *Combinatorial optimisation*, John Wiley & Sons, Hoboken (NJ).

[25] Chu C, 2005, *A heuristic algorithm for the truckload and less-than-truckload problem*, European Journal of Operational Research, **165(3)**, pp. 657–667.

[26] Clarke G & Wright JW, 1964, *Scheduling of vehicles from a central depot to a number of delivery points*, Operations Research, **12(4)**, pp. 568–581.

[27] Cordeau J, Desaulniers G, Desrosiers M, Solomon MM & Soumis F, 2002, *VRP with time-windows*, pp. 309–330 in Toth P & Vigo D (Eds), *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[28] Cordeau JF, Gendreau M & Laporte G, 1997, *A tabu search heuristic for periodic and multi-depot vehicle routing problems*, Networks, **30(2)**, pp. 105–119.

[29] Cordeau JF, Gendreau M, Laporte G, Potvin JY & Semet F, 2002, *A guide to vehicle routing heuristics*, Journal of the Operational Research Society, **53(5)**, pp. 512–522.

[30] Cordeau JF, Laporte G & Mercier A, 2001, *A unified tabu search heuristic for vehicle routing problems with time-windows*, Journal of the Operational Research Society, **52(8)**, pp. 928–936.

[31] Cordeau JF & Maischberger M, 2012, *A parallel iterated tabu search heuristic for vehicle routing problems*, Computers and Operations Research, **39(9)**, pp. 2033–2050.

[32] Crevier B, Cordeau JF & Laporte G, 2007, *The multi-depot vehicle routing problem with inter-depot routes*, European Journal of Operational Research, **176(2)**, pp. 756–773.

[33] Dantzig GB, 1949, *Programming of interdependent activities, Part II: Mathematical model*, Journal of the Econometric Society, **17(3)**, pp. 200–211.

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

[35] Darwin C, 1859, *On the origin of species by means of natural selection or the preservation of favoured races in the struggle for life*, John Murray, London.

[36] Davis L, 1991, *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York (NY).

[37] Deb K, 2014, *Multi-objective optimisation*, pp. 403–449 in Burke EK & Kendall G (Eds), *Search methodologies*, Springer, New York (NY).

[38] Deb K, Pratap A, Agarwal S & Meyarivan T, 2002, *A fast and elitist multi-objective genetic algorithm: NSGA-II*, Transactions on Evolutionary Computation, **6(2)**, pp. 182–197.

[39] Desaulniers G, Desrosiers J, Erdmann A, Solomon MM & Soumis F, 2002, *VRP with pickup and delivery*, pp. 225–242 in Toth P & Vigo D (Eds), *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[40] Dorigo M, Maniezzo V & Colorni A, 1991, *Positive feedback as a search strategy*, (Unpublished) Technical Report TR 91-016, Department of Electronics, Politecnico di Milano, Milan.

[41] Drexl M, 2012, *Synchronization in vehicle routing — A survey of VRPs with multiple synchronization constraints*, Transportation Science, **46(3)**, pp. 297–316.

[42] Drexl M, 2013, *Applications of the vehicle routing problem with trailers and transshipments*, European Journal of Operational Research, **227(2)**, pp. 275–283.

[43] Dror M & Trudeau P, 1989, *Savings by split delivery routing*, Transportation Science, **23(2)**, pp. 141–145.

[44] Elshaer R & Awad H, 2020, *A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants*, Computers and Industrial Engineering, **140**, pp. 1–19.

[45] Emmerich M, Shir OM & Wang H, 2018, *Evolution strategies*, pp. 89–119 in Martí R, Pardalos PM & Resende MGC (Eds), *Handbook of heuristics*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[46] Ergezer M & Simon D, 2011, *Oppositional biogeography-based optimisation for combinatorial problems*, Proceedings of the 2011 IEEE Congress of Evolutionary Computation, New Orleans (LA), pp. 1496–1503.

[47] Feo TA & Resende MG, 1989, *A probabilistic heuristic for a computationally difficult set covering problem*, Operations Research Letters, **8(2)**, pp. 67–71.

[48] Fleischmann B, 1985, *A cutting plane procedure for the travelling salesman problem on road networks*, European Journal of Operational Research, **21(3)**, pp. 307–317.

[49] Fogel LJ, Owens AJ & Walsh MJ, 1966, *Artificial intelligence through simulated evolution*, John Wiley & Sons, Hoboken (NJ).

[50] 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).

[51] Fukasawa R, Longo H, Lysgaard J, De Aragão MP, Reis M, Uchoa E & Werneck RF, 2006, *Robust branch-and-cut-and-price for the capacitated vehicle routing problem*, Mathematical Programming, **106(3)**, pp. 491–511.

[52] Garaix T, Artigues C, Feillet D & Josselin D, 2010, *Vehicle routing problems with alternative paths: An application to on-demand transportation*, European Journal of Operational Research, **204(1)**, pp. 62–75.

[53] Gendreau M, Iori M, Laporte G & Martello S, 2006, *A tabu search algorithm for a routing and container loading problem*, Transportation Science, **40(3)**, pp. 342–350.

[54] Gendreau M, Potvin JY, 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).

[55] Glover F, 1977, *Heuristics for integer programming using surrogate constraints*, Decision Sciences, **8(1)**, pp. 156–166.

[56] Glover F, 1986, *Future paths for integer programming and links to artificial intelligence*, Computers and Operations Research, **13(5)**, pp. 533–549.

[57] Glover F, 1997, *A template for scatter search and path relinking*, Proceedings of the 3^{rd} Annual European Conference on Artificial Evolution, Nîmes, pp. 1–51.

[58] Glover F & Kochenberger GA, 2006, *Handbook of metaheuristics*, Springer, New York (NY).

[59] Goel A & Meisel F, 2013, *Workforce routing and scheduling for electricity network maintenance with downtime minimization*, European Journal of Operational Research, **231(1)**, pp. 210–228.

[60] Golden BL, Wasil EA, Kelly JP & Chao IM, 1998, *The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results*, pp. 33–56 in Crainic TG & Laporte G (Eds), *Fleet management and logistics*, Springer, New York (NY).

[61] Gomory R, 1958, *Essentials of an algorithm for integer solutions to linear programs*, Bulletin of the American Mathematical Society, **64(5)**, pp. 269–302.

[62] Gordon N, Wagner IA & Bruckstein AM, 2003, *Discrete bee dance algorithm for pattern formation on a grid*, Proceedings of the 2003 IEEE/WIC International Conference on Intelligent Agent Technology, Halifax, pp. 545–549.

[63] Groër C, Golden B & Wasil E, 2010, *A library of local search heuristics for the vehicle routing problem*, Mathematical Programming Computation, **2(2)**, pp. 79–101.

[64] Groër C, Golden B & Wasil E, 2011, *A parallel algorithm for the vehicle routing problem*, INFORMS Journal on Computing, **23(2)**, pp. 315–330.

[65] Gulczynski D, Golden B & Wasil E, 2010, *The split delivery vehicle routing problem with minimum delivery amounts*, Transportation Research Part E: Logistics and Transportation Review, **46(5)**, pp. 612–626.

[66] Hachicha M, Hodgson MJ, Laporte G & Semet F, 2000, *Heuristics for the multi-vehicle covering tour problem*, Computers and Operations Research, **27(1)**, pp. 29–42.

[67] Hansen N & Ostermeier A, 2001, *Completely derandomised self-adaptation in evolution strategies*, Evolutionary Computation, **9(2)**, pp. 159–195.

[68] Hillis WD, 1990, *Co-evolving parasites improve simulated evolution as an optimisation procedure*, Physica D: Nonlinear Phenomena, **42(1-3)**, pp. 228–234.

[69] Holland JH, 1975, *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor (MI).

[70] Homberger J & Gehring H, 1999, *Two evolutionary metaheuristics for the vehicle routing problem with time-windows*, Information Systems and Operational Research, **37(3)**, pp. 297–318.

[71] Jin J, Crainic TG & Løkketangen A, 2012, *A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems*, European Journal of Operational Research, **222(3)**, pp. 441–451.

[72] Jozefowiez N, Semet F & Talbi EG, 2009, *An evolutionary algorithm for the vehicle routing problem with route balancing*, European Journal of Operational Research, **195(3)**, pp. 761–769.

[73] Jung SH, 2003, *Queen-bee evolution for genetic algorithms*, Electronics Letters, **39(6)**, pp. 575–576.

[74] Karaboga D & Akay B, 2009, *A survey: Algorithms simulating bee swarm intelligence*, Artificial Intelligence Review, **31(1)**, pp. 61–85.

[75] Kawamura H, Yamamoto M, Mitamura T, Suzuki K & Ohuchi A, 1998, *Cooperative search based on pheromone communication for vehicle routing problems*, Transactions on Fundamentals of Electronics, Communications and Computer Sciences, **81(6)**, pp. 1089–1096.

[76] Kennedy J & Eberhart R, 1995, *Particle swarm optimisation*, Proceedings of the 1995 International Conference on Neural Networks, Perth, pp. 1942–1948.

[77] Kilby P, Prosser P & Shaw P, 1999, *Guided local search for the vehicle routing problem with time-windows*, pp. 473–486 in Voss S, Martello S, Osman IH & Roucairol C (Eds), *Meta-heuristics*, Springer, New York (NY).

[78] Kirkpatrick S, Gelatt CD & Vecchi MP, 1983, *Optimisation by simulated annealing*, Science, **220(4598)**, pp. 671–680.

[79] Kirlik G & Sayin S, 2015, *Computing the nadir point for multi-objective discrete optimisation problems*, Journal of Global Optimisation, **62(1)**, pp. 79–99.

[80] Knowles J & Corne D, 2003, *Properties of an adaptive archiving algorithm for storing non-dominated vectors*, IEEE Transactions on Evolutionary Computation, **7(2)**, pp. 100–116.

[81] Kontoravdis G & Bard JF, 1995, *A GRASP for the vehicle routing problem with time-windows*, ORSA Journal on Computing, **7(1)**, pp. 10–23.

[82] Koza JR, 1992, *Genetic programming: On the programming of computers by means of natural selection*, MIT Press, Cambridge (MA).

[83] Kytöjoki J, Nuortio T, Bräysy O & Gendreau M, 2007, *An efficient variable neighborhood search heuristic for very large scale vehicle routing problems*, Computers and Operations Research, **34(9)**, pp. 2743–2757.

[84] Land AH & Doig AG, 1960, *An automatic method of solving discrete programming problems*, Econometrica, **28(3)**, pp. 497–520.

[85] Laporte G & Nobert Y, 1988, *A vehicle flow model for the optimal design of a two-echelon distribution system*, pp. 158–173 in Eiselt H & Pederzoli G (Eds), *Advances in optimisation and control*, Springer, New York (NY).

[86] Li F, Golden B & Wasil E, 2007, *The open vehicle routing problem: Algorithms, large-scale test problems, and computational results*, Computers and Operations Research, **34(10)**, pp. 2918–2930.

[87] Li M & Yao X, 2019, *Quality evaluation of solution sets in multi-objective optimisation: A survey*, ACM Computing Surveys, **52(2)**, pp. 1–38.

[88] Lieberman GJ & Hillier FS, 2005, *Introduction to operations research*, McGraw-Hill, New York (NY).

[89] Lučić P & Teodorović D, 2001, *Bee system: Modeling combinatorial optimisation transportation engineering problems by swarm intelligence*, Proceedings of the 4th Annual Triennial Symposium on Transportation Analysis, Saõ Miguel, pp. 441–445.

[90] Lučić P & Teodorović D, 2003, *Vehicle routing problem with uncertain demand at nodes: The bee system and fuzzy logic approach*, pp. 67–82 in Verdegay JL (Ed), *Fuzzy sets based heuristics for optimisation*, Springer, New York (NY).

[91] Ma J, Zhang JP, Yang J & Cheng LL, 2008, *Research on cultural algorithm for solving routing problem of mobile agent*, The Journal of China Universities of Posts and Telecommunications, **15(4)**, pp. 121–125.

[92] MacArthur RH & Wilson EO, 2016, *The theory of island biogeography*, Princeton University Press, Princeton (NJ).

[93] Machado P, Tavares J, Pereira FB & Costa E, 2002, *Vehicle routing problem: Doing it the evolutionary way*, Proceedings of the 2002 Genetic and Evolutionary Computation Conference, New York (NY), pp. 690–696.

[94] Martin GE, 2001, *The art of enumerative combinatorics*, Springer, New York (NY).

[95] Mercer RE & Sampson J, 1978, *Adaptive search using a reproductive meta-plan*, Kybernetes, **7(3)**, pp. 215–228.

[96] Mester D & Bräysy O, 2007, *Active-guided evolution strategies for large-scale capacitated vehicle routing problems*, Computers and Operations Research, **34(10)**, pp. 2964–2975.

[97] Mezura-Montes E, Reyes-Sierra M & Coello Coello CA, 2008, *Multi-objective optimisation using differential evolution: A survey of the state-of-the-art*, pp. 173–196 in Chakraborty UK (Ed), *Advances in differential evolution*, Springer, New York (NY).

[98] Miliotis P, 1978, *Using cutting planes to solve the symmetric travelling salesman problem*, Mathematical Programming, **15(1)**, pp. 177–188.

[99] Min H, 1989, *The multiple vehicle routing problem with simultaneous delivery and pick-up points*, Transportation Research Part A: General, **23(5)**, pp. 377–386.

[100] Mingyong L & Erbao C, 2010, *An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time-windows*, Engineering Applications of Artificial Intelligence, **23(2)**, pp. 188–195.

[101] Mladenovic N, 1995, *A variable neighborhood algorithm — A new metaheuristic for combinatorial optimisation*, Presentation at the 2nd Optimisation Days Conference, Sydney.

[102] Mühlenbein H & Paass G, 1996, *From recombination of genes to the estimation of distributions, Part I: Binary parameters*, Proceedings of the 4th Annual International Conference on Parallel Problem Solving from Nature, Berlin, pp. 178–187.

[103] Nagata Y & Bräysy O, 2009, *Edge assembly-based memetic algorithm for the capacitated vehicle routing problem*, Networks, **54(4)**, pp. 205–215.

[104] Orloff C, 1974, *A fundamental problem in vehicle routing*, Networks, **4(1)**, pp. 35–64.

[105] 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.

[106] Pérez-Rodríguez R & Hernández-Aguirre A, 2019, *A hybrid estimation of distribution algorithm for the vehicle routing problem with time-windows*, Computers and Industrial Engineering, **130**, pp. 75–96.

[107] Pisinger D & Ropke S, 2007, *A general heuristic for vehicle routing problems*, Computers and Operations Research, **34(8)**, pp. 2403–2435.

[108] Porto VW & Fogel LJ, 1997, *Evolution of intelligently interactive behaviors for simulated forces*, Proceedings of the 6th Annual International Conference on Evolutionary Programming, Indianapolis (IN), pp. 419–429.

[109] Price K, Storn RM & Lampinen JA, 2006, *Differential evolution: A practical approach to global optimisation*, Springer, New York (NY).

[110] Prins C, 2004, *A simple and effective evolutionary algorithm for the vehicle routing problem*, Computers and Operations Research, **31(12)**, pp. 1985–2002.

[111] Prins C, 2009, *A GRASP $\times$ evolutionary local search hybrid for the vehicle routing problem*, pp. 35–53 in Pereira FB & Tavares J (Eds), *Bio-inspired algorithms for the vehicle routing problem*, Springer, New York (NY).

[112] Rechenberg I, 1965, *Cybernetic solution path of an experimental problem*, (Unpublished) Technical Report TR 1122, Royal Aircraft Establishment, Farnborough.

[113] Reimann M, Doerner K & Hartl RF, 2004, *D-Ants: Savings based ants divide and conquer the vehicle routing problem*, Computers and Operations Research, **31(4)**, pp. 563–591.

[114] Renaud J, Laporte G & Boctor FF, 1996, *A tabu search heuristic for the multi-depot vehicle routing problem*, Computers and Operations Research, **23(3)**, pp. 229–235.

[115] Reynolds RG, 1994, *An introduction to cultural algorithms*, Proceedings of the 3rd Annual Conference on Evolutionary Programming, San Diego (CA), pp. 131–139.

[116] Robust F, Daganzo CF & Souleyrette II RR, 1990, *Implementing vehicle routing models*, Transportation Research Part B: Methodological, **24(4)**, pp. 263–286.

[117] Rochat Y & Taillard ÉD, 1995, *Probabilistic diversification and intensification in local search for vehicle routing*, Journal of Heuristics, **1(1)**, pp. 147–167.

[118] Ross S, 2002, *A first course in probability*, Prentice-Hall, Upper Saddle River (NJ).

[119] Salazar-Aguilar MA, Langevin A & Laporte G, 2012, *Synchronised arc routing for snow plowing operations*, Computers and Operations Research, **39(7)**, pp. 1432–1440.

[120] Schwefel HP, 1981, *Numerical optimisation of computer models*, John Wiley & Sons, Hoboken (NJ).

[121] Shaw P, 1998, *Using constraint programming and local search methods to solve vehicle routing problems*, Proceedings of the 4th Annual International Conference on Principles and Practice of Constraint Programming, Pisa, pp. 417–431.

[122] Simon D, 2008, *Biogeography-based optimisation*, Transactions on Evolutionary Computation, **12(6)**, pp. 702–713.

[123] Smit SK & Eiben AE, 2009, *Comparing parameter tuning methods for evolutionary algorithms*, Proceedings of the 2009 IEEE Congress on Evolutionary Computation, Trondheim, pp. 399–406.

[124] Srinivas N & Deb K, 1994, *Muilti-objective optimisation using non-dominated sorting in genetic algorithms*, Evolutionary Computation, **2(3)**, pp. 221–248.

[125] Storn R & Price K, 1997, *Differential evolution — A simple and efficient heuristic for global optimisation over continuous spaces*, Journal of Global Optimisation, **11(4)**, pp. 341–359.

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

[127] Subramanian A, Uchoa E & Ochi LS, 2013, *A hybrid algorithm for a class of vehicle routing problems*, Computers and Operations Research, **40(10)**, pp. 2519–2531.

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

[129] Taillard ÉD, Badeau P, Gendreau M, Guertin F & Potvin JY, 1997, *A tabu search heuristic for the vehicle routing problem with soft time-windows*, Transportation Science, **31**, pp. 170–186.

[130] Taillard ÉD, Laporte G & Gendreau M, 1996, *Vehicle routeing with multiple use of vehicles*, Journal of the Operational Research Society, **47(8)**, pp. 1065–1070.

[131] Tang L & Wang X, 2006, *Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem*, International Journal of Advanced Manufacturing Technology, **29(11)**, pp. 1246–1258.

[132] Tarantilis CD, 2005, *Solving the vehicle routing problem with adaptive memory programming methodology*, Computers and Operations Research, **32(9)**, pp. 2309–2327.

[133] Tillman FA, 1969, *The multiple terminal delivery problem with probabilistic demands*, Transportation Science, **3(3)**, pp. 192–204.

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

[135] Toth P & Vigo D, 2002, *VRP with backhauls*, pp. 195–224, *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[136] Toth P & Vigo D, 2003, *The granular tabu search and its application to the vehicle routing problem*, INFORMS Journal on Computing, **15(4)**, pp. 333–346.

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

[138] Vidal T, 2016, *Split algorithm in O(n) for the capacitated vehicle routing problem*, Computers and Operations Research, **69**, pp. 40–47.

[139] Vidal T, Crainic TG, Gendreau M, Lahrichi N & Rei W, 2012, *A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems*, Operations Research, **60(3)**, pp. 611–624.

[140] Vidal T, Crainic TG, Gendreau M & Prins C, 2013, *A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows*, Computers and Operations Research, **40(1)**, pp. 475–489.

[141] Voudouris C, 1997, *Guided local search for combinatorial optimisation problems*, PhD Dissertation, University of Essex, Essex.

[142] Wade AC & Salhi S, 2002, *An investigation into a new class of vehicle routing problem with backhauls*, Omega, **30(6)**, pp. 479–487.

[143] Willard JAG, 1989, *Vehicle routing using r-optimal tabu search*, Master's Thesis, Management School, Imperial College, London.

[144] Winston WL, 2004, *Operations research: Applications and algorithms*, Duxbury Press, Boston (MA).

[145] Wren A & Holliday A, 1972, *Computer scheduling of vehicles from one or more depots to a number of delivery points*, Journal of the Operational Research Society, **23(3)**, pp. 333–344.

[146] Zachariadis EE & Kiranoudis CT, 2010, *A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem*, Computers and Operations Research, **37(12)**, pp. 2089–2105.

[147] Zitzler E & Thiele L, 1998, *Multi-objective optimisation using evolutionary algorithms — A comparative case study*, Proceedings of the 5th Annual International Conference on Parallel Problem Solving from Nature, Amsterdam, pp. 292–301.