Master's thesis

David Gulaker
Johannes Log Indbjo
Mathias Waage Sørensen

Heuristically Solving Large-Scale Location Routing Problems with Stochastic Customer Presence for Online Grocery Delivery

Master's thesis in Industrial Economics and Technology Management
Supervisor: Magnus Stålhane
June 2023

oda

David Gulaker
Johannes Log Indbjo
Mathias Waage Sørensen

Heuristically Solving Large-Scale Location Routing Problems with Stochastic Customer Presence for Online Grocery Delivery

Master's thesis in Industrial Economics and Technology Management
Supervisor: Magnus Stålhane
June 2023

Norwegian University of Science and Technology
Faculty of Economics and Management
Dept. of Industrial Economics and Technology Management

Preface

This thesis concludes our Master of Science in Industrial Economics and Technology Management at the Norwegian University of Science and Technology (NTNU). It is written as a part of the course TIØ4905 Managerial Economics and Operations Research, during the spring of 2023. The thesis is a continuation of the work done in the project report we wrote as part of the course TIØ4500 during the autumn of 2022.

We would like to thank our supervisor, Professor Magnus Stålhane (Department of Industrial Economics and Technology Management, NTNU), for the valuable guidance and feedback we have received throughout the process. Furthermore, we want to thank our collaboration partner Oda, and in particular Ivar Brekkå for close follow-up and for showing a huge interest in our research. We are grateful for the opportunity to work with such an exciting and relevant problem domain.

Trondheim, June 1, 2023

David Gulaker	[illegible]	[illegible]
David Gulaker	Johannes Log Indbjo	Mathias Waage Sørensen

Summary

This master's thesis studies heuristic optimization methods for solving a large and complex Location Routing Problem (LRP) with stochastic customer presence in collaboration with Oda, a Norwegian online grocery company. To deliver groceries to thousands of customers daily, Oda needs to solve a very large multi-depot capacitated Vehicle Routing Problem (VRP) with time windows, a heterogeneous fleet and tour duration constraints. When entering a new market with uncertainty related to which customers place an order, or customer presence, they must determine where the depots that vehicles drive from should be located. To find good depot locations in such situations, we model a two-stage stochastic LRP. The first stage considers locating depots, and the second stage handles routing from depots to customers for a realization of the customer presence. LRPs with stochastic customer presence of this size and complexity have not been researched before. We propose solution methods consisting of three parts: a depot location search algorithm, a scenario generation method, and a routing algorithm. These are tested on realistic problem instances with 20 possible depot locations and thousands of customers in Berlin.

For searching through first-stage solutions, a Greedy Randomized Adaptive Search Procedure (GRASP) and Constructive Adaptive Tabu Search (CATS), which first finds a promising number of open depots and then performs a local search with an adaptive neighborhood, are used. CATS performs more consistently, but both methods find solutions within 12 hours with cost gaps of less than 0.2% of the best known solutions. CATS achieves solutions within a 1% cost gap after only 2 hours. Since evaluating solutions is time-consuming, the extensions in CATS are necessary to find good solutions quickly. Stochasticity is handled by generating scenarios with different numbers of customers that are weighted from the likelihood of similar scenarios based on historical distributions of demand. Out-of-sample stability tests show that using 16 scenarios is sufficient to ensure stability, especially when it comes to ranking solutions correctly. The calculation of routing costs follows a three-phase approach. First, customers are assigned to depots, then similar customers are clustered, before a Hybrid Genetic Search solver creates routes. Compared to benchmarks from Oda's operational solver, this method demonstrates satisfactory results as it achieves a stable ranking of solutions solving the problems 20 times faster than Oda's solver.

In a case study, we use the CATS algorithm to gain insights for Oda. We find that opening depots is first profitable when exceeding an average of 3000 customers per shift, and that the fleet size and composition are important attributes in determining a depot's attractiveness.

Sammendrag

Denne oppgaven presenterer en studie av heuristiske optimeringsmetoder for å løse et stort og komplekst “location-routing”-problem (LRP) med stokastisk kundetilstedeværelse, i samarbeid med det norske nettbaserte dagligvareselskapet Oda. For å levere matvarer og andre produkter til tusenvis av kunder, løser de daglig store “vehicle-routing”-problemer (VRP) med flere depoter, kapasitets- og varighetsbegrensninger, tidsvinduer og heterogene kjøretøy. Ved utvidelse til nye markeder må de først beslutte hvor depotene som kjøretøyene starter rutene sine fra skal lokaliseres, som er utfordrende med usikkerhet knyttet til hvilke kunder som vil bestille, også kalt kundetilstedeværelse. Vi modellerer et stokastisk, to-stegs LRP for å finne gode depotlokasjoner i slike situasjoner. I første steg besluttes depotlokasjonene, mens andre steg finner kostnaden til kjøretøysrutene for ulike realiseringer av kundetilstedeværelse. LRP av denne størrelsen og kompleksiteten med stokastisk kundetilstedeværelse er ikke forsket på tidligere. De foreslåtte løsningsmetodene består av en algoritme for å søke gjennom ulike depotlokasjoner, en scenariogenereringsmetode og en rutingalgoritme. Disse testes på realistiske probleminstanser med 20 mulige depotlokasjoner og tusenvis av kunder i Berlin.

Søket gjennom førstestegsløsningene gjøres med en “Greedy Randomized Adaptive Search Procedure” (GRASP), samt “Constructive Adaptive Tabu Search” (CATS), som først finner et lovende antall åpne depoter og deretter utfører lokalsøk med et adaptivt nabolag. CATS presterer mest konsistent, men begge metodene finner løsninger i løpet av 12 timer med under 0.2% kostnadsavvik til den beste kjente løsningen. CATS finner løsninger med under 1% avvik etter bare 2 timer. Siden løsningsevalueringen er tidkrevende, er utvidelsene i CATS viktige for å finne gode løsninger fort nok. Stokastisiteten håndteres ved å generere scenarioer med ulikt antall kunder, som vektes etter sannsynligheten for lignende scenarioer basert på historiske etterspørselsfordelinger. “Out-of-sample”-tester viser at bruk av 16 scenarioer er nok for å oppnå stabilitet, spesielt når det kommer til å rangere løsningene riktig. Utregningen av rutekostnader er tredelt. Først tildeles kundene et depot, så grupperes lignende kunder sammen, før en “Hybrid Genetic Search”-løsningsmetode konstruerer rutene. Sammenlignet med løsninger fra Odas operasjonelle ruteplanlegger, viser denne metoden tilfredsstillende resultater ved å oppnå stabil rangering av løsninger selv om problemene løses 20 ganger raskere enn Odas ruteplanlegger.

Vi anvender CATS i en case-studie for å gi innsikt til Oda. Her finner vi at åpning av depoter først er lønnsomt når man har et gjennomsnitt på 3000 kunder per skift. Vi ser også at størrelsen på kjøretøysflåten og hvordan den er sammensatt er viktige faktorer når man skal vurdere lønnsomheten ved å åpne ulike depoter.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The market for online groceries is having a boom in many European cities. Revenue increased by 70% in the European online grocery market from 2019 to 2022 and now accounts for 6.1% of the total European grocery market (McKinsey & Company and Eurocommerce, 2023). Many online grocery retailers offer home delivery to the customers, and since food should be kept fresh it raises more complex distribution challenges than with many other products. Thousands of customers should be served in a short time, with very varying demand from day to day. Coordination is necessary to ensure that the customers are home to receive the food when it is delivered. With low margins and a highly competitive market, creating distribution networks that serve these customers with their demand in an effective way is important to succeed.

Which vehicle routes that are used to deliver the groceries to the customers are dependent on the location of depots where the vehicles are loaded. Still, depot location and routing decisions are rarely taken in conjunction with each other. This observation has led to research on a problem called the Location Routing Problem (LRP). The LRP is a computationally massive problem, as it involves finding the cheapest locations of depots, which are determined by the cost of routing to customers from them. Obviously, it is easier to consider locating depots and routing to customers as separate problems, but research has shown that this leads to suboptimal solutions (Salhi and Rand, 1989). Consequently, there is an incentive for online grocery retailers to model and solve the problem of locating depots as an LRP.

This thesis is motivated by a real-world problem faced by the online grocery retailer Oda. With recent expansions to Berlin, Oda is in a situation where future growth requires new depots. However, solving an LRP of the kind needed by Oda raises three challenges. First, as several depots may be opened, and with many potential locations, there are too many different combinations of open depots to evaluate all of them. Smart ways must therefore be found to search through and select which combinations of depots, or depot configurations, should be evaluated. Second, the Vehicle Routing Problem (VRP) in itself is considered a complex problem but still has to be solved quickly. Such VRPs, with thousands of customers as in Oda's case, also correspond to some of the largest problem instances solved in the literature (Arnold et al., 2019). The VRP is complex because it addresses constraints such as time windows, implying that customers can only be served within specific time slots. While the literature may use hours or even days to solve such VRPs, we want to solve it within minutes to be able to evaluate as many depot configurations as possible. Third, the demand varies from day to day in terms of the number of customers, the quantity they demand and their location. This uncertainty must somehow be handled when solving the problem.

To solve this large and complex LRP we adapt methods from recent literature and extend these with new components for our specific application. In the model, we take into account uncertainty related to which customers place an order, also called customer presence. The aim is to answer to what degree the proposed solution methods are able to efficiently find high-quality solutions for this problem, and how the stochastic customer presence can be modeled. We can then apply the most promising solution method to give insights to Oda regarding the location of depots in Berlin. Furthermore, we hope this research can lay a good foundation for how to decompose and solve complex real-world challenges as LRPs.

Through the literature review, we have found little research on solving LRPs with the exact same challenges as in this thesis. However, parts of the challenges have been researched for themselves. Mara et al. (2021) present a survey with different approaches researched to solve the LRP and shows that heuristics and metaheuristics account for 87% of the proposed solution methods in literature from 2014 to 2019. For the stochastic LRP, local search based metaheuristics seem to be the most promising methods. To deal with uncertainty related to customer demand, fuzzy logic, robust optimization and stochastic programming are commonly used (Mara et al., 2021; Prodhon and Prins, 2014). However, how to model the stochastic customer presence in problems similar to Oda's is barely discussed in the literature. When using stochastic programming it becomes increasingly important to evaluate the routing problem fast to be able to consider several scenarios. This introduces a trade-off between computational time and accuracy in the cost estimates. The project report (Gulaker et al., 2022) focuses on this specific trade-off, where a solution method is proposed in which problem decomposition and state-of-the-art routing metaheuristics are combined. This solution method is able to rank depot configurations in problems with 2000 customers consistently with Oda's routing solver, Navegante, in 95% less computational time.

In light of the research in the literature on the LRP, this thesis contributes in several ways. Based on stochastic programming, it contributes with a two-stage model and associated solution methods for the LRP with stochastic customer presence, where depot locations are decided in the first stage and the routing is performed in the second stage. To close the gap in the literature regarding stochastic customer presence, a new scenario generation method is proposed and tested. Using this method, we show that 16 scenarios are sufficient to achieve stability. For the depot location search in the first stage, we implement two different metaheuristics and compare their behavior on real-world problem instances provided by Oda. We propose a first-improvement Tabu Search, called Constructive Adaptive Tabu Search (CATS), where we explore different neighborhoods and strategies for balancing diversification and intensification in the search. Moreover, we propose an approach not tested on the LRP with stochastic customer presence before, based on a Greedy Randomized Adaptive Search Procedure (GRASP). In experiments performed in this thesis, they both demonstrate their ability to find good solutions fast, and CATS finds a 1% gap to the best known solution within 2 hours. Regarding the routing in the second stage, we use a three-phase routing algorithm, based on Gulaker et al. (2022). Here, we first convert the multi-depot problem into a set of single-depot problems by using the parallel assignment algorithm proposed by Giosa et al. (2002), then cluster similar customers into super-customers by using the algorithm by Dondo and Cerdá (2007), and finally solve the routing problems with the Hybrid Genetic Search solver proposed by Kool et al. (2022). We show that this routing algorithm ranks depot configurations consistently compared to solutions from Navegante.

We want to emphasize that the focus of this thesis is not to contribute with a routing solver that minimizes the gap to a benchmark solution. As already described, there is a trade-off between solution quality and the time used to solve the routing problem in each scenario. The longer the time spent on creating good routes, the fewer solutions can be evaluated in a given time. We assume that it is sufficient to give good enough estimates of the costs as long as the relative ranking of depot configurations is consistent with a benchmark solution. Hence, lower solution quality can be allowed as long as the ranking is preserved. This assumption is reflected in how we measure the performance of the solution methods. Rather than only comparing the final cost gap to the benchmark solution, we also assess the ability to reach good solutions fast. Rather than using only the final cost to assess the impact different drawings of scenarios have, we also consider how the ranking is affected. To increase confidence in the applicability of our approach, we use solutions from Navegante to benchmark the solution methods.

The rest of the thesis is structured the following way. Chapter 2 describes details about Oda's market situation, operations and considerations during expansion. In Chapter 3, we review literature that is relevant to this problem, with an emphasis on how stochastic customer presence is modeled. The two-stage stochastic model for the LRP is defined mathematically in Chapter 4. In Chapter 5, we describe the proposed solution methods, including the details of the used algorithms. Different methods of searching through possible depot configurations are presented and we describe the three-phase routing algorithm. In Chapter 6, we explain the distribution that is used to represent the stochastic customer presence and how scenarios are generated. Chapter 7 contains an overview of the experiments performed and their results. It provides a test of stability related to scenarios, an evaluation of the routing algorithm with a comparison to Gulaker et al. (2022), an evaluation of how CATS and GRASP perform, and an analysis of solution attributes influencing costs. It finishes with a case study for Oda where we look into questions that provide insights related to locating depots in Berlin. Finally, Chapter 8 concludes the thesis by extracting the main findings, and Chapter 9 suggests where new research could focus its attention to expand the literature on the field.

Chapter 2

Background

In 2013, Oda was started by a group of entrepreneurs with one goal in mind: making everyday grocery shopping easier and more accessible to people. They wanted to achieve this by opening an online grocery store with a highly optimized value chain. In 2021, Oda received the status of a unicorn company after being valued at more than one billion dollars (Finansavisen, 2021), confirming both their success and that the online grocery market has a lot of potential. They have now expanded to Finland and Germany, while still retaining the position as the market leader in Norway (E24, 2022).

This chapter presents background information that is relevant to understand Oda's business and the market they work in. A brief introduction to the online grocery market with recent developments is first given in Section 2.1. Section 2.2 provides an overview of Oda's operations. Relevant strategic decisions related to Oda's expansion are described in Section 2.3.

2.1 The Online Grocery Market

In Simmons et al. (2022), online grocery is framed as the next S-curve of growth. They explain that technology is in the process of disrupting several parts of the value chain, from user experience to order preparation and last-mile delivery. Furthermore, they expect that technological advancements could enable grocers to have lower prices online, which can create a boom for the online grocery market. Because of these trends, the online grocery market has been flooded with new companies in the last few years.

This thesis focuses on Oda's expansion to Germany, and hence the German online grocery market. Germany's grocery retail sector has annual sales volumes between 220 and 300 billion euros, which makes it the largest grocery retail market in Europe (Thedens and Hachibiti, 2022). However, Germany is just catching up on the online grocery market trend compared to leading European countries such as UK and France (Simmons et al., 2022), meaning that there is still a great potential for innovation in this market. Because of this, several brands have made their entry in

Berlin and Munich to test their concepts. At the same time, some of the largest supermarkets in Germany, like Rewe and Edeka, have already opened online platforms with home delivery. Oda might therefore experience considerably more competition in Germany compared to Norway, making value chain optimization perhaps more important than ever.

2.2 Operations in Oda

A deeper look into Oda's operations is required to understand what processes are a part of an online grocery store, and how optimization is embedded in this system. Oda's customers can choose from thousands of products. Most of the products are groceries, but they also offer an increasing selection of non-grocery products such as books, toys and appliances. Customers place their orders on a digital platform. In addition to the option to buy different products, Oda provides a wide range of recipes so that customers easily can set up a menu for the week and order all the products needed. When customers check out, they are offered several time slots during the day to get the products delivered, with different transportation costs for each of them. In general, it is cheaper to choose time slots with a wide time range than a narrow time range. Factors such as order size, transportation distance and time slot popularity are also considered when Oda chooses which time slots to offer, and what they cost. Delivering orders happens in two shifts each day, one in the morning and one in the afternoon. The morning shift lasts from 06:00 to 14:00, and the afternoon shift lasts from 14:00 to 22:00. Orders delivered in the morning must be placed by 20:00 on the day before, whereas orders delivered in the afternoon can be placed until midnight the day before. Delivery on the same day is also possible if the order is placed in the morning but at a greater cost.

The logistics for distributing the products are complex. An overview of the processes that takes place on a daily basis is presented in Figure 2.1. At the fulfillment center (FC), it is made sure that the products placed by the customers are available, and that products are picked and stacked in boxes according to the orders. Oda's route planner Navegante decides the routes to drive. When the route planner is terminated and the boxes are prepared, line-haul trucks transport them to the distribution points (DIPs), which are small buildings or temporary structures strategically located close to customers.

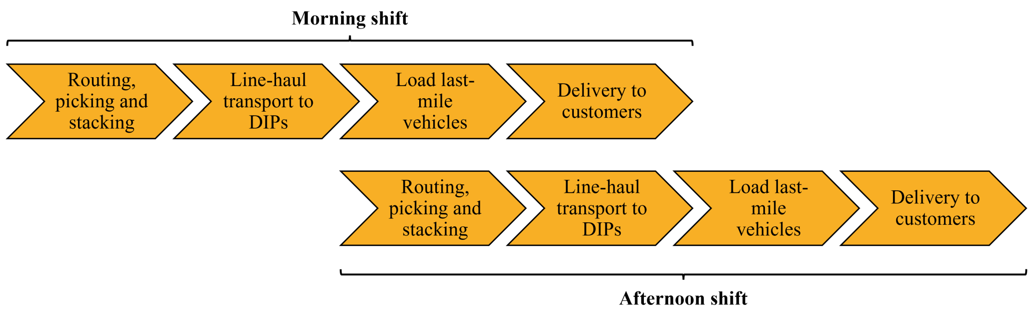

Figure 2.1: Daily operations in Oda.

At DIPs, the boxes are unloaded and then loaded again into smaller last-mile vehicles. Figure 2.2 shows the difference between a last-mile vehicle and a line-haul truck. A last-mile vehicle can typically hold 70-100 boxes, depending on the boxes' weight and volume, and can have different cost structures, while a line-haul truck can hold around 600 boxes. The number of loading ramps at the DIPs is typically lower than the number of last-mile vehicles, so they must be loaded in different time slots. Additionally, there is only a limited number of vehicles located at each DIP, and the same last-mile vehicles are used for both shifts. The last-mile vehicles can start their route to the customers when they are finished loading.

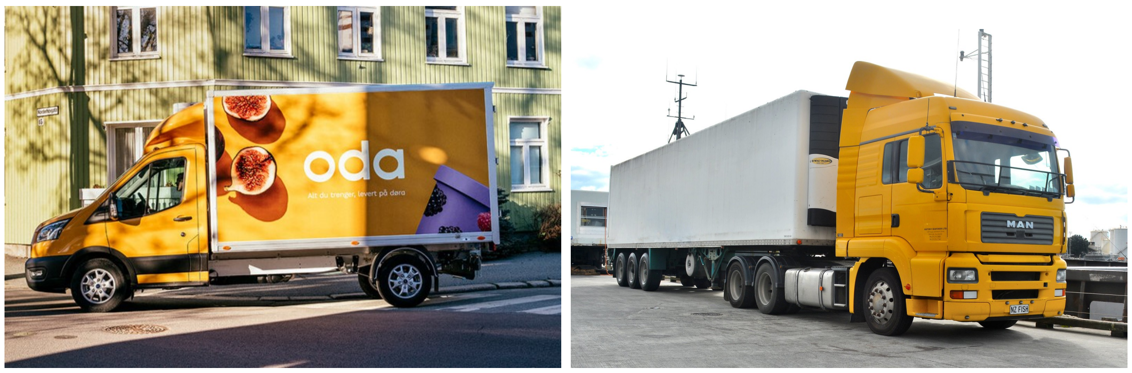

Figure 2.2: Example of a last-mile vehicle (left) and a line-haul truck (right). Photos by Oda (2022) and GPS 56 (2017).

The creation of the routes used for delivery is of special interest to this thesis. Route planning consists of creating routes from the FC to the DIPs, in addition to the routes to the customers. This routing process is illustrated in Figure 2.3. Line-haul trucks transport the packed boxes from the FC to the DIPs. Here, the boxes are stored temporarily until they are picked up by the last-mile vehicles and transported to the customers' doorsteps. Boxes to each customer are always transported together so that each customer is visited by only one vehicle. As can be seen in Figure 2.3, there are some last-mile vehicles that drive directly from the FC to the customers. Thus, the FC in practice serves the role of both a central warehouse and a DIP, and is therefore referred to as the warehouse in this thesis.

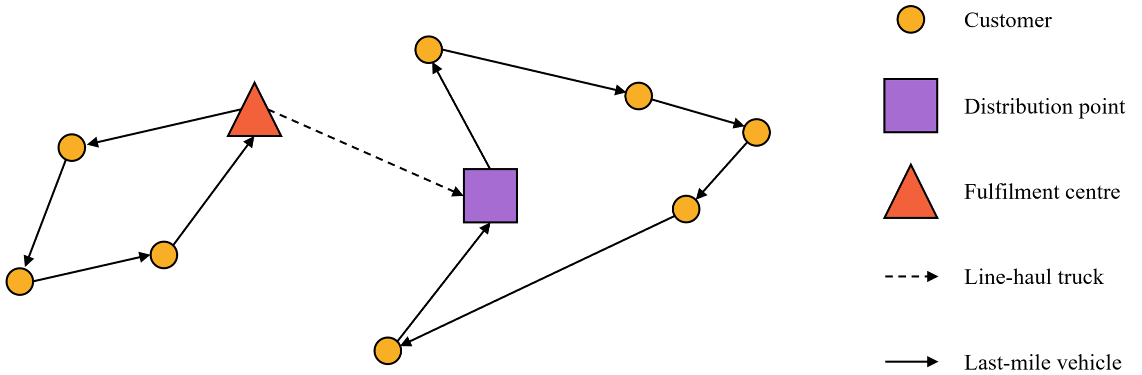

Figure 2.3: Routing in Oda.

2.3 Strategic Planning during Expansion

Several strategic decisions must be taken when expanding to new countries, some of which have an impact on the daily operations in Oda. Because of limited resources, for instance in terms of fleet size, Oda must decide where they should deliver in order to maximize their profit. This is a central part of what is called revenue management. Not only must they decide in which city to open for delivery, but it could also be necessary to decide what areas in the city to serve. All these decisions affect where Oda should open the FC and potentially new DIPs. The locations of the FC and DIPs are furthermore connected to the vehicle routes, as the locations determine the start and end point of routes for the line-haul trucks and the last-mile vehicles. This chain of decisions shows how daily operations can be affected by strategic decisions, and are therefore interrelated.

As decisions on a strategic and operational level affect each other, this should be an incentive for making them simultaneously. This way, the information about the consequences becomes more accurate, which can be used to increase the efficiency of the value chain even more. With an expansion to new countries, and in particular Germany, this information might therefore be valuable for Oda. They have already decided to start delivery in Berlin, where they also have opened an FC, but have yet to open any DIP. To enable future growth, deciding where to open these DIPs must be taken in the near future. Consequently, this is a great opportunity to combine the decision of locating DIPs in Berlin with routing decisions.

Chapter 3

Related Literature

In this chapter, we present the literature related to our work. First, the literature search strategy is described in Section 3.1. Then, a brief overview of the Facility Location Problem (FLP) and the Vehicle Routing Problem (VRP) is given in Section 3.2 and 3.3, respectively. In Section 3.4, we define the Location Routing Problem (LRP) and explain the most prominent solution methods. Section 3.5 goes more into detail on the modeling and solution methods for LRPs with stochastic customer presence. Finally, a description of our contribution to the literature follows in Section 3.6.

3.1 Literature Search Strategy

For the literature search, Google Scholar, Elsevier and NTNU Oria are used as the literature search engines, complemented by surveys on the LRP. Table 3.1 displays the surveys we use and what time period they cover. The preceding surveys are referred to as previous works in later surveys. Together, the surveys cover the literature on the LRP from its beginning in 1964 until 2019. For describing the FLP and VRP, we use the survey Klose and Drexl (2005) and the literature review from the project report (Gulaker et al., 2022).

Table 3.1: Surveys used in the literature review for the Location Routing Problem.

Period	Survey
- 2006	Nagy and Salhi (2007)
2007 - 2013	Prodhon and Prins (2014)
2014 - 2019	Mara et al. (2021)

In order to identify relevant literature, the search in the search engines and the survey references follows a three-stage search strategy. In the first stage, general keywords referring to the LRP are used to guide the search. All papers which contain at least one of the first-stage keywords are included. In the second stage, more specific keywords are used to search through papers that passed the first stage.

To capture the uncertainty that is present in our problem, the problem in the papers must either be stochastic or two-stage. Additionally, as we are dealing with a large problem, either the terms "heuristic" or "large" must be used for the problem. All keywords used in the literature search are listed in Table 3.2.

Table 3.2: Keywords for the literature search. Included papers must contain at least one term from each column.

First-stage keywords	Second-stage keywords	
Problem type	Problem variant	Additional terms
Location Routing Problem	Stochastic	Heuristic
Location-Routing Problem	Two-stage	Large
LRP		

In the third stage, a more detailed reading of the papers that passed the first two stages is conducted. As a part of this reading, the papers are evaluated based on a set of inclusion criteria, shown in Table 3.3. They define the key features of the target papers. Inclusion criteria IC1 more specifically determines that we are only interested in LRPs that optimize distribution in a single distribution echelon. This means we are only interested in the routing of last-mile vehicles to customers, and not the routing of line-haul trucks to depots. Many types of uncertainty might occur in an LRP, and hence we use IC2 to state that we include papers that consider uncertainty related to the customers. We have also chosen to narrow down solution methods to those who use a stochastic programming approach. Therefore, IC3 excludes papers that handle uncertainty with either fuzzy or robust optimization. The inclusion criteria are applied to sort out only the relevant papers, ultimately leading to 7 papers. These papers are presented in Section 3.5 of the literature review and used to define our contribution in Section 3.6.

Table 3.3: Inclusion criteria used on candidate papers.

IC#	Inclusion criteria
IC1	The Location Routing Problem is only single-echelon
IC2	Uncertain customer demand is considered
IC3	Not using fuzzy or robust optimization

3.2 The Facility Location Problem

The FLP origins from the Weber problem, which is defined in 1909 (Klose and Drexl, 2005). This problem formulates a model in which a single facility is to be located so that the sum of distances to all the demand points is minimized. When including multiple potential facilities, possible questions to be answered in the FLP are which facilities to open and which facilities each customer should be served from. A facility is in this setting a general term that can take different forms based on the specific application. In the context of Chapter 2, we use the term depot to refer to what Oda calls distribution points and what is called a facility in the literature. Hence, a depot in this thesis refers to a place where goods are stored temporarily before they are packed into vehicles and transported to the customers.

Because of its applicability for the construction of distribution systems and supply chain management, the FLP has become a well-known problem in the operations research domain, and several extensions have been defined (Klose and Drexl, 2005). Relevant extensions for our problem are listed in Table 3.4. The standard FLP is represented with a discrete location space. This is related to the fact that decision-makers in real-world location problems rarely have the freedom to build facilities wherever they want. Furthermore, a common extension of the FLP is to make the facility capacitated, meaning that they have a constraint on the maximum demand they are able to cover. Since the late 1900s and until today, the effort has been on including more complex real-world aspects in the models. One of these branches is research on the FLP with stochastic data, often in the form of uncertain customer demand.

Table 3.4: Relevant extensions of the standard Facility Location Problem.

Characteristic	Standard	Extension
Location space	Discrete	Continuous
Depot capacity	Uncapacitated	Capacitated
Data	Deterministic	Stochastic

3.3 The Vehicle Routing Problem

Dantzig and Ramser (1959) introduce the problem that today is known as the VRP. In this formulation, which they call the Truck Dispatching Problem, they try to make cost-effective routes between customers in order to fulfill their demand. Vehicles start and end their routes at a depot. The total capacity of each vehicle is normally restricted by using some measure of quantity like volume, weight or both. If the vehicles have restricted capacity they are commonly called capacitated, and otherwise uncapacitated. The canonical form of the VRP is the capacitated VRP (CVRP). A range of different extensions can be added to this formulation to close the gap to the logistic problems as they appear in the real world. The VRP extensions that are most relevant to our problem are listed in Table 3.5. First, the vehicles do not necessarily need to be equal to each other, or homogeneous. This might be due to different capacities or cost structures. A fleet that consists of different types of vehicles is called heterogeneous. Second, it is possible to extend the VRP to take into account more than one depot, called multi-depot. Third, several time constraints are introduced in the literature, the most relevant being time windows for customers and depots, and duration constraints for how long each vehicle can be used before it must return to the depot. Time windows give an interval of time in which customers can be serviced by vehicles or vehicles are available to use.

Table 3.5: Relevant extensions of the standard Vehicle Routing Problem.

Characteristic	Standard	Extension(s)
Vehicle fleet	Homogeneous	Heterogeneous
Number of depots	Single-depot	Multi-depot
Time constraints	None	Time windows, tour duration

In order to cope with the scale of real-world problems and the complexity introduced by the extensions in Table 3.5, Gulaker et al. (2022) identify that much of the literature has focused on combining problem decomposition with fast solution methods. Research on decomposition distinguishes between clustering and districting. Clustering groups customers based on similarities whereas districting partitions the geographical space into smaller zones, or districts, that contain only one depot. Moreover, they find that nearest neighbor algorithms are popular for clustering, which in its simplest form just groups customers with their closest neighbors. Dondo and Cerdá (2007) propose a nearest neighbor algorithm that respects constraints on vehicle capacity, heterogeneous fleet and time windows. Regarding districting, the literature is more deficient. Gulaker et al. (2022) find, however, that the parallel assignment algorithm proposed by Giosa et al. (2002) handles large problem instances with up to 1000 customers in 2 seconds. In comparison with 6 other algorithms for districting, parallel assignment seems to be the best trade-off between computational time and performance. Based on these findings, Gulaker et al. (2022) propose to first use the parallel assignment algorithm by Giosa et al. (2002) for districting and then the clustering algorithm by Dondo and Cerdá (2007) on the customers in each district to create super-customers.

To find the routes in large and complex VRPs, several algorithms are worth mentioning. We adopt the terms from Arnold et al. (2019) to describe the VRP scale, where instances with between 100 and 1000 customers are considered large, and instances with more than 1000 customers are considered very large. To quickly find good routes in very large VRPs, Arnold et al. (2019) develop a knowledge-guided local search (KGLS) that can efficiently solve VRP instances of up to 30,000 customers. It uses both the heuristic by Lin and Kernighan (1973) and the savings heuristic by Clarke and Wright (1964) to construct initial solutions, before running a local search that perturbs solutions and penalizes bad routes based on knowledge of bad routes found in Arnold and Sörensen (2019). KGLS is benchmarked against the Lin-Kernighan-Helsgaun version 3 (LKH-3) heuristic, which is a much-used heuristic capable of solving VRPs, including VRPs with time windows (VRPTW) (Helsgaun, 2017). It works by transforming a VRPTW into many multi-traveling salesperson problems and minimizes both the routing time and the penalty for violating constraints. According to Arnold et al. (2019), LKH-3 targets high solution quality instead of fast solution times, and is therefore used to benchmark solution quality.

However, for solving CVRPs, the Hybrid Genetic Search for capacitated vehicle routing problems (HGS-CVRP) proposed by Vidal (2022) is considered state-of-the-art in terms of solution quality and convergence on CVRP instances of up to 1000 customers. HGS-CVRP outperforms among others the aforementioned KGLS and

LKH-3 algorithms (Vidal, 2022). It is a specialized implementation for the CVRP based on the HGS algorithm by Vidal et al. (2012). The HGS algorithm found in Vidal et al. (2012) is made to solve the periodic VRPs, the multi-depot VRP and the multi-depot periodic VRP. The principle of a hybrid genetic search is to combine a genetic algorithm with a local search algorithm. It works by keeping two pools of solutions, one feasible and one infeasible. In each iteration of the algorithm, parents are selected by a binary tournament. An ordered crossover operator is used to create offspring, which is further improved (educated) by a local search algorithm. The offspring is placed according to its feasibility in the feasible or infeasible pool of solutions. When the size of a solution pool reaches its maximum size, the best solutions and the solutions that contribute most to diversity are selected for survival. There are also measures to diversify if the best solution does not improve for a given number of iterations. The algorithm runs for a specified time or until reaching a specified number of non-improving iterations.

HGS has been extended to solve the VRPTW, where it also has made its claim as one of the best-performing solution methods. One such implementation is proposed by Vidal et al. (2013), and is called HGS with Adaptive Diversity Control (HGSADC). Another HGS for VRPTW implementation is the solver created by Kool et al. (2022), which won the VRPTW track of the 12th DIMACS implementation challenge (DIMACS, 2022). This solver is called Router, but to indicate its connection to the HGS algorithm we refer to it as HGS Router (HGSR) in this thesis. HGSR includes additional construction heuristics, a new crossover operator called Selective Route Exchange and an intensified local search procedure. Contrary to HGSADC, an open-source routing software for HGSR exists (PyVRP, 2023).

HGSADC, HGSR and LKH-3 have all been tested in the literature on the VRPTW benchmark instances by Gehring and Homberger (1999) with 1000 customers (see Appendix A). Table 3.6 compares the average, maximum and minimum percentage gap to the best known solution on these benchmarks. The runtime limit used in the test for HGSADC and HGSR is 120 minutes, whereas LKH-3 is not specified. HGSR finds a much lower gap compared to the other solvers. Additionally, the gaps for HGSADC and LKH-3 have larger ranges between maximum and minimum gaps, indicating that they are less consistent than HGSR.

Table 3.6: Comparison between Hybrid Genetic Search with Adaptive Diversity Control (HGSADC), Lin-Kernighan-Helsgaun 3 (LKH-3) and Hybrid Genetic Search Router (HGSR) by using average, maximum and minimum gap (%) to best known solution values on the large Gehring and Homberger (1999) Vehicle Routing Problem with time windows instances and 1000 customers.

	HGSADC	LKH-3	HGSR
Average	2.51	2.23	0.32
Maximum	14.74	14.39	1.02
Minimum	0.08	0.08	0.00

3.4 The Location Routing Problem

Maranzana (1964) describes an algorithm that takes into account transport costs from routes when deciding the location of distribution points. This paper is by many considered the first publication on the LRP, which more generally can be defined as location planning with tour planning aspects taken into account (Nagy and Salhi, 2007). With a hierarchical view, the LRP is a compound problem consisting of the FLP as the master problem and the VRP as the subproblem. As shown in Figure 3.1, while the FLP only needs to answer what facilities to open and the assignment of customers to facilities, the LRP must both decide facility locations and the routes to serve customers from these facilities. The most common version of the LRP is the capacitated LRP (Prodhon and Prins, 2014), meaning that facilities and vehicles have a maximum capacity. This assumption is implicit when we refer to the LRP throughout this thesis.

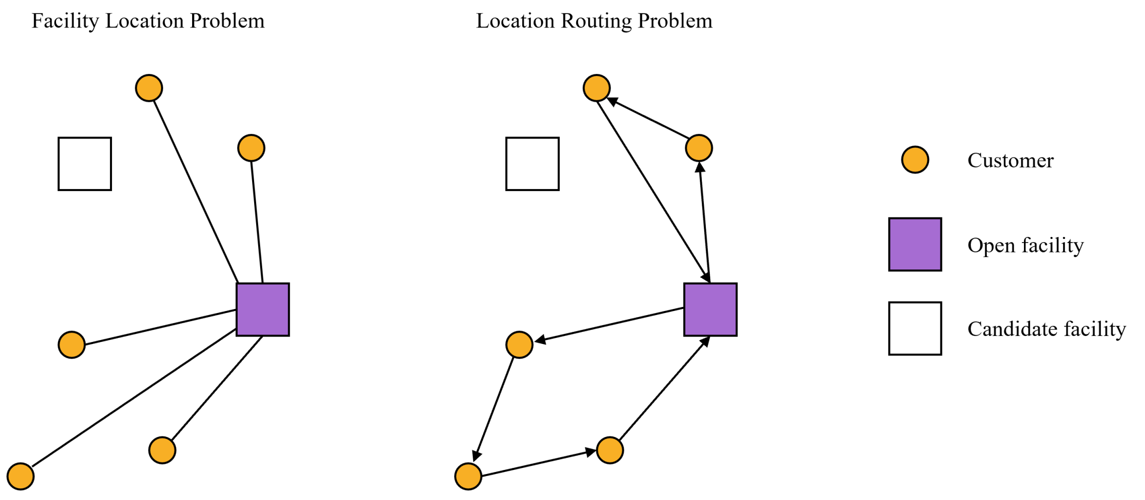

Figure 3.1: Connection between the Facility Location Problem and the Location Routing Problem.

Salhi and Rand (1989) show that the classical strategy of solving the FLP and the VRP separately often leads to suboptimal solutions. One can therefore ask why these two problems are not always solved simultaneously as an LRP. They elaborate on a few possible reasons for this. First, an obvious reason is that not all location problems have a routing aspect. Second, many researchers have claimed that it is inappropriate to combine location and route planning because the former is typically a strategic decision whereas the latter is a tactical or operational decision. Third, the LRP is NP-hard as it can be reduced to the FLP and the VRP, both of which are NP-hard. Hence, the LRP is a complex and computationally demanding problem to solve. These reasons could explain why the LRP has received less attention than the FLP and VRP since Maranzana (1964), especially early on when the technology did not satisfy the computational requirements for the LRP. In a survey examining publications from 2007 to 2019, Mara et al. (2021) identify a shift regarding the research on LRP. Particularly, they show that the LRP has received increasingly more attention towards the end of this time period.

To solve LRPs, many solution methods have been used. Exact solution methods, such as branch-and-cut (Belenguer et al., 2011) and branch-and-price with column generation (Baldacci et al., 2011), are possible on relatively small problems. The largest problem solved with exact methods, according to Prodhon and Prins (2014), is up to 200 customers and 14 depots. For larger problems, it is more common to solve the LRP by using heuristics and metaheuristics. There are numerous variants, but the most common metaheuristics for solving single-objective LRPs are Simulated Annealing or Genetic Algorithm (Mara et al., 2021). Simulated Annealing is a local search based metaheuristic that accepts non-improving solutions with a probability that is regulated during the search, whereas genetic algorithms evolve a population (a set of solutions) by using concepts inspired by biology.

The popularity of Simulated Annealing and Genetic Algorithms is likely connected to their ability to balance diversification and intensification. In a Simulated Annealing algorithm proposed by Ferreira and de Queiroz (2018), a diversification procedure is used that opens and closes a depot, both randomly chosen. The component analysis shows that this particular procedure has a significant impact on the average gap when tested on benchmark problems. Yu et al. (2019) utilize the ability Genetic Algorithm has to intensify the search on promising parts of the solution space. They keep two populations, one with infeasible solutions and another with feasible solutions, and use crossover operators to mix them when producing new solutions, called reproduction. The motivation for this procedure is that the best solution often is found to be close to the boundary between feasible and infeasible solutions. Diversifying the search has a great impact in Ferreira and de Queiroz (2018), while Yu et al. (2019) show how intensification can be utilized based on the knowledge we have about the solution space. Nevertheless, the right balance between the two seems to be important when solving the LRP.

There are also solution methods that combine Simulated Annealing and Genetic Algorithms, among other components. Voigt et al. (2022) propose a hybrid between Adaptive Large Neighborhood Search and Genetic Algorithm with a range of other components such as Tabu Search and Simulated Annealing. The Genetic Algorithm is the main algorithm, whereas the Adaptive Large Neighborhood Search is a subroutine used to locally improve child solutions, a process referred to as education. In the neighborhood search, they adaptively change the probability of selecting a neighborhood operator based on observed performance, which is a method originally proposed by Pisinger and Ropke (2007). They perform detailed component analysis that shows how adaptively deciding the neighborhood helps to find high-quality solutions in less computational time, and that Simulated Annealing has a high impact on the overall performance.

Another metaheuristic that is mentioned in the literature is the Greedy Randomized Adaptive Search Procedure (GRASP). This metaheuristic consists of an initial phase where a solution is found by using a greedy algorithm with some randomness, and a second phase that improves this solution. Hence, the initial phase can be viewed as a diversification phase, whereas the second phase can be viewed as an intensification phase. GRASP has a wide variety of applications. It can be used together with different metaheuristics, such as Evolutionary Local Search (Duhamel et al., 2010). It can also be used as an initial phase to construct good starting solutions (Contardo et al., 2014). Prodhon and Prins (2014) compare the performance of metaheuristics on LRP benchmark problems in the period from 2007 to 2013, where GRASP is used in one of the best-performing metaheuristics on LRP benchmark problems in the period from 2007 to 2013. Mara et al. (2021) show that GRASP has been used in the literature from 2014 to 2019, but not to the same extent as before.

3.5 The Location Routing Problem with Stochastic Customer Presence

Of particular interest to this thesis are papers that solve problems where the presence of customers is stochastic. By stochastic customer presence, we mean that potential customer locations are known, but it is uncertain whether they demand service or not. Therefore, depots must be located to minimize the cost of serving many different realizations of customer presence, which we call different customer sets. This resembles real-life situations where not every potential customer orders each day. However, this should not be confused with multi-period problems in which demand can accumulate between periods. Below follow descriptions of how problems with stochastic customer presence are modeled as a stochastic program in the literature and what solution methods are used.

3.5.1 Modeling

Some papers in the literature split the routing part of the LRP with stochastic customer presence into a priori and a posteriori routes. The a priori routes are created before the customer presence is revealed but can be altered afterward in an a posteriori route in exchange for a penalty or additional cost. In this thesis, we refer to this modeling approach as corrective routing. Albareda-Sambola et al. (2007) handle stochastic customer presence by using this approach, where customer presence is modeled by a Bernoulli random variable. In the first stage, a set of open depots is chosen to satisfy all customer service requests with a specified probability. Then, customers are assigned to depots and a priori routes connecting all customers are designed. In the second stage, customer presence is revealed, and the routes must be altered to only visit present customers. It is also possible that there are not enough open depots to satisfy the observed customer demand, in which case a penalty is incurred. The objective is to minimize the cost of opening depots, the expected penalty costs and a posteriori routing costs. Tordecilla et al. (2020) take into account safety stock and different depot sizes when creating the a priori routes. Safety stock corresponds to empty space in the vehicles when creating a priori routes in case the demand in the assigned route is revealed to be higher than expected. The a posteriori routes are then constructed by making corrective changes to the a priori routes where there is excess demand, either by forcing vehicles to make detours back to the depot or extending routes of vehicles that are not full.

Another modeling approach is to take only decisions connected to the locating of depots in the first stage, and then routing in the second stage. This modeling approach is referred to as post routing in this thesis. Corredor-Montenegro et al. (2021) follow this approach to handle the uncertainty connected to customer presence, or active clients in their terms. By understanding the statistical properties of the historical demand, they generate demand observations that replicate the historical properties. Further, a mixed integer programming model is used to select a small subset of scenarios that represent the variability in demand. They emphasize the need for simulating demand instead of basing their model only on historical observations to handle cases where future demand is greater than previously observed demand. Post routing is also used by Klibi et al. (2010), but they consider a scenario as the daily demand over a time period of several days. Monte Carlo simulation is used to generate the scenarios. In this respect, it is a multi-period problem. However, after the scenarios have been generated, they solve routing problems for each day in the scenario independently of other days. They report that they need 6 scenarios, each spanning over 200 days, to achieve sufficient solution accuracy without increasing solution times too much. De Maio et al. (2022) present another example of a two-stage stochastic program with post routing, but their model differs from the other papers in how they represent the uncertainty. They argue against deterministic models using sample means for demand as this fails to provide good solutions in the face of variability. However, they do not model future uncertainty with an explicit probability distribution. Instead, they use only historic realizations of demand and solve for these realizations, contrary to Corredor-Montenegro et al. (2021).

3.5.2 Methods

Nagy and Salhi (2007) propose a classification of solution methods to the LRP. They propose four different categories: sequential, cluster-based, iterative and hierarchical. The sequential methods solve the LRP by first locating the depots and then computing the routing costs. Due to the lack of an integrated view, these methods are not considered to solve an LRP. In the cluster-based methods, clusters of customers are created for each potential depot or route, which reduces the complexity of the problem. Iterative methods solve the location and routing problem many times in succession, each time passing feedback between the two phases. Finally, hierarchical methods view the LRP as different decisions at different levels, where the location decisions are considered the main task and the routing is a subordinate problem used to calculate the costs of the location decisions. We use the same categories to describe the following methods.

The most common heuristic solution framework in our review is an iterative framework. Javid and Azad (2010), Albareda-Sambola et al. (2007), Zhang et al. (2019) and Tordecilla et al. (2020) use local search based methods in the iterative framework to improve an initial solution. The operators that are used, both for routing and depot locations, are dynamically updated throughout the search in order to guide the search toward more promising parts of the solution space. For depot locations, operators that are commonly used in the papers are to open a depot, close a depot and both open and close simultaneously. Javid and Azad (2010) use a combination of Tabu Search and Simulated Annealing, whereas Albareda-Sambola et al. (2007) use a Variable Neighborhood Search that sequentially applies a local search for each operator. Zhang et al. (2019) also use Variable Neighborhood Search, but only to improve the best particle (solution) in Particle Swarm Optimization. The heuristics proposed by Tordecilla et al. (2020) use Iterated Local Search, which restarts the local search from a different initial solution whenever a local optimum is reached. In contrast to the other papers in the literature review, they use simulations to estimate the costs of the location decisions.

Klibi et al. (2010) use a metaheuristic solution method with Tabu Search. It comments on the temporal hierarchical nature of their problem, a stochastic multi-period location transportation problem, where location decisions occur before and with a longer time horizon than the daily routing decisions. Therefore, a hierarchical framework is used. At the upper level, the distribution network that should serve the customers is designed, while the daily routes to serve customers are created at the lower level. The routing is solved by a modification of the Clarke and Wright (1964) Savings algorithm and the network design problem is solved using Tabu Search.

There are also papers that attempt to solve the LRP with stochastic customer presence using exact methods. Both Corredor-Montenegro et al. (2021) and De Maio et al. (2022) implement a two-stage stochastic model where the expected routing cost is found by using scenarios. Of particular interest is how these papers facilitate the usage of exact methods in such a complex problem as the LRP. Corredor-Montenegro et al. (2021) use valid inequalities to reduce the size of the solution space, and they reuse routes from previous solutions to construct high-quality initial solutions in later iterations. De Maio et al. (2022) use a ranking and selection method in which they assume that the decision maker is indifferent to selecting an alternative to the current best solution if the difference in solution values is smaller than an indifference zone. They use a confidence interval to decide how many scenarios they need to solve before being indifferent between two solutions, which greatly reduces the computational time.

3.6 Our Contribution

The contribution of this thesis is summarized in Table 3.7 by comparing our thesis with the papers presented in Section 3.5. Now that the FLP and VRP are defined, we classify Oda's situation by using the literature. Recall from Chapter 2 that the last-mile trucks can hold 70-100 boxes, transportation happens in two shifts, there are several depots, and the customers can make orders within specific time slots. The depots also have a limited number of vehicles available, and the customer presence is uncertain. Depots cannot be located anywhere in a city either, there are some potential locations to consider. This translates to an FLP with discrete location space and capacitated depots as the facilities. Stochastic customer presence means that there is stochastic data in the problem. The vehicle delivery routines translate to a VRP with capacitated vehicles, a heterogeneous fleet, multiple depots, time windows and tour duration constraints. Together, the FLP and VRP form a stochastic LRP because decisions in the FLP affect the VRP, and vice versa. On top of this, the problem at hand is very large because the number of possible customers in Berlin is in the magnitude of thousands.

Table 3.7: Contribution of our thesis compared to included papers for the Location Routing Problem with stochastic customer presence. ILS, VNS, GRASP and PSO are abbreviations for Iterated Local Search, Variable Neighborhood Search, Greedy Randomized Adaptive Search Procedure and Particle Swarm Optimization, respectively. Very large indicates problem sizes with at least 1000 customers.

Paper	Classification														
	Characteristics						Model		Method						
	Discrete	Capacitated	Heterogeneous vehicles	Time windows	Tour duration	Very large	Post routing	Corrective routing	Exact	ILS	VNS	GRASP	Tabu Search	PSO	Simulated Annealing
De Maio et al. (2022)	x	x					x		x						
Corredor-Montenegro et al. (2021)	x	x		x	x		x		x						
Tordecilla et al. (2020)	x	x						x		x					
Zhang et al. (2019)	x	x					x				x			x	
Javid and Azad (2010)	x	x					x						x		x
Klibi et al. (2010)	x	x				x	x						x		
Albareda-Sambola et al. (2007)	x	x						x			x				
Sum	7	7	0	1	1	1	5	2	2	1	2	0	2	1	1
This thesis	x	x	x	x	x	x	x					x	x		

To solve the LRP with stochastic customer presence we propose a two-stage stochastic solution method. In our view, corrective routing is best suited for problems where the customer locations are known or relatively certain before the demand is revealed, thus being able to make a priori routes that only need small corrections in the second stage. In contrast, the problem in this thesis needs to handle large variations in where customers are located in each scenario. Therefore, we choose the post routing modeling approach. Additionally, our thesis proposes a new scenario generation method. Few of the papers in the literature review mention how the scenarios are generated and, in general, it seems that there is little agreement on how stochastic customer presence is best represented. We fill this research gap by putting more emphasis on how the scenarios are generated and on testing the solution quality on realistic problems.

We implement and compare two different location search algorithms. These weigh diversification and intensification differently, which seems to be an important feature in the methods that are presented in the literature review. Comparing two different algorithms might also make us more aware of potential weaknesses and strengths, helping us to decide which one is the most promising for our application. The location search algorithms that are implemented are an extended Tabu Search and GRASP. This comparison of algorithms for our application is in itself an interesting contribution to the literature. In addition, the extensions proposed to the Tabu Search are important contributions of this thesis. Compared to most of the literature, our algorithms have time to evaluate relatively few solutions in total, meaning that they need to intensify the search more, especially in the beginning of the search. Hence, we need to find good solutions fast. Inspired by Klibi et al. (2010), we add tabu lists based on which moves have been made earlier. We use three different neighborhoods, which are explored differently in two phases of the search. First, in the constructive phase, we quickly find a solution with a reasonable number of open depots. This solution is the starting point for the adaptive phase, where the neighborhoods are explored with different probabilities that are adjusted throughout the search, inspired by Voigt et al. (2022). These constructive and adaptive phases are why the algorithm is referred to as Constructive Adaptive Tabu Search (CATS). GRASP, the other solution method, is implemented as a much simpler method with only the initial diversification phase. Hence, by design, GRASP represents a different approach to solving the problem.

The routing algorithm for the second-stage problem is based on the implementation in the project report (Gulaker et al., 2022). In particular, we reuse the decomposition method where the customers are assigned to districts and then clustered. Gulaker et al. (2022) use LKH-3 for constructing routes. However, recent research results and the comparison of the VRP solvers in Section 3.3 shows that HGSR produces lower gaps and also is more consistent than LKH-3. Therefore, we incorporate the open-source version of HGSR for routing. This way, the goal is to make the routing algorithm from the project report more scalable and accurate. As stated in the literature review, the VRP that is solved by Gulaker et al. (2022) takes into account the same extensions as in this thesis. To the best of our knowledge, this VRP is more complex than any of the VRPs solved as a part of a stochastic LRP in the literature.

Chapter 4

Problem Description and Mathematical Model

This chapter describes the LRP this thesis aims to solve. Gulaker et al. (2022) formulate the VRP that describes the real-world routing problem Oda's solver Navegante solves for each shift. In this thesis, an extension of this model to solve the LRP is proposed: a two-stage stochastic mixed integer program, with the depot location decision in the first stage and the vehicle routing decision in the second stage. The stochastic aspect regards the stochastic customer presence, where the second stage is solved for a realization of the stochastic parameters in the first stage. The two stages of the problem are described in Section 4.1 and 4.2, respectively. A mathematical model of the problem is then presented in Section 4.3.

4.1 Description of the First-Stage Problem

Food boxes should be delivered from one of several depots given in the set $\mathcal{N}^D$. The first depot in $\mathcal{N}^D$, with index 0, corresponds to the warehouse. As mentioned in Chapter 2, the warehouse takes on two different roles: delivery of food boxes to the depots with line-haul trucks and delivery of food boxes to the customers with last-mile vehicles. However, in this thesis, we are not interested in the routing of line-haul trucks, meaning that the warehouse can be viewed as a larger depot that must be open.

The binary variable δ_i takes the value 1 if depot $i \in \mathcal{N}^D$ is open, and 0 otherwise. Each depot can either be open or closed, except for the warehouse which is always open. Thus, the vector δ contains all binary variables δ_i that specify which depots in $\mathcal{N}^D$ that are open or closed, which is referred to as a depot configuration. Furthermore, several alternative depots at the same location can be considered. These are mutually exclusive options, meaning that alternative depots at the same location cannot be opened together. To enforce this decision, the parameter $\mathcal{K}_{ij}$ must be 1 for all pairs of depots $i \in \mathcal{N}^D$ and $j \in \mathcal{N}^D \setminus \{i\}$ that can be opened at the same time, and 0 if they cannot. A fleet size γ, being the total number of last-mile vehicles across all open depots, is also decided.

The objective of the first-stage problem is to open the depots that minimize the cost of the depot configuration, which consists of the costs of opening the depots, the fleet costs and the expected operational costs of the routes to serve the customers. For each depot $i \in \mathcal{N}^D$, the opening cost G_i consists of depot leasing costs and line-haul costs. The depot leasing costs are the costs to lease the depot, and the line-haul costs are the costs for transporting the food boxes with line-haul trucks to the depot. Ongoing expenses incurred from opening the depot, like electricity and cleaning, are also included in the depot leasing costs. The fleet cost G^F is the cost that is incurred for having the vehicles available in the fleet, such as vehicle leasing and services. Finally, the expected operational costs are the expected routing costs from the second-stage problem for the depot configuration δ, the fleet size γ and the uncertain set of customers ξ. All these costs are scaled to a cost per shift in order to make them comparable to each other.

4.2 Description of the Second-Stage Problem

In this section, the second-stage VRP is described, and all the notation that is relevant is defined.

4.2.1 Customers

The set of all possible customers is denoted $\mathcal{N}^C$. However, for a given shift only a subset of these customers actually place an order. Thus, the set of customers that have ordered food boxes for delivery is uncertain. To describe this uncertainty, we define F_i to be a stochastic binary parameter that is 1 if customer $i \in \mathcal{N}^C$ places an order, and otherwise 0. Let ξ be the collection of the stochastic parameters, that is $\xi = (F_0, ..., F_{|\mathcal{N}^C|})$. We assume that the probability distributions of the parameters in ξ are known. A specific realization $\tilde{\xi}$ of ξ contains realizations of all the stochastic parameters, each denoted as $\tilde{F}_i$. Note that all attributes connected to each customer, being location, demand and time window, are deterministic. All possible locations (both depots and customers) are given in the set $\mathcal{N}$, whereas individual locations are denoted by indices i, j and k.

4.2.2 Vehicles and Depot Loading

Each depot $i \in \mathcal{N}^D$ has a set of vehicles available, denoted by $\mathcal{V}_i$. We assume that the number of vehicles available at the warehouse is large enough to cover the demand for all possible realizations $\tilde{\xi}$ of ξ, and hence that we never need to decide whether a customer should not be served. The set of all vehicles is denoted by $\mathcal{V}$, and is equivalent to $\bigcup_{i \in \mathcal{N}^D} \mathcal{V}_i$. Since a vehicle $v \in \mathcal{V}_i$ belonging to a depot $i \in \mathcal{N}^D$ cannot visit other depots, we also introduce the set $\mathcal{N}_v$ containing all locations reachable by the vehicle, which includes the set of customers and only the depot i. More formally, it can be defined as $\mathcal{N}_v = \mathcal{N}^C \cup \{i\}$ for all $i \in \mathcal{N}^D$ and $v \in \mathcal{V}_i$. Each vehicle must serve the whole demand of each customer it visits. The demand of customer $i \in \mathcal{N}^C$ is given by D_i. At the same time, a vehicle $v \in \mathcal{V}$ has a maximum capacity H_v^V.

Each depot also has a given number of loading docks, limiting how many vehicles that can be loaded simultaneously. This maximum number of vehicles is denoted by L_i for depot $i \in \mathcal{N}^D$. To better represent time and simultaneity, we introduce a set of time intervals $\mathcal{T}$ where each interval $\tau \in \mathcal{T}$ has a start time $\underline{T}_\tau^I$ and an end time $\overline{T}_\tau^I$. There could for instance be a time interval for every five minutes. We introduce the three variables $k_{v\tau}$, $e_{v\tau}$ and $d_{v\tau}$ to indicate whether vehicle $v \in \mathcal{V}$ starts loading during timeslot $\tau \in \mathcal{T}$, ends loading during the timeslot, or is loading during the whole timeslot, respectively.

4.2.3 Time, Availability and Overtime

The time it takes to drive between location $i \in \mathcal{N}$ and location $j \in \mathcal{N}$ is denoted by T_{ij}. Each location $i \in \mathcal{N}$ has a service time given as T_i^S. This is the time it takes for the vehicle to serve the customer, or to load goods at the depot. When the customers place orders, they also choose a time window in which their order shall be delivered. The lower bound of the time window for customer $j \in \mathcal{N}^C$ is denoted as $\underline{T}_j^N$, and the upper bound as $\overline{T}_j^N$. Time windows are also set for vehicles, as they have start and end times for when they are available. These are denoted by $\underline{T}_v^V$ and $\overline{T}_v^V$ respectively for vehicle $v \in \mathcal{V}$. In addition to the vehicle time windows, a working shift has a maximum duration for $v \in \mathcal{V}$, given by T_v^D. It also has a maximum duration for ordinary work time denoted by T_v^O, which works as a threshold for when overtime is counted. All time in shifts that surpasses this overtime threshold is counted as overtime.

4.2.4 Break Times

If a shift lasts longer than a given time, drivers need to be given a break. This break threshold time is given by T_v^B. Breaks are represented by the variable b_{ijv}, which should be set to 1 if vehicle $v \in \mathcal{V}$ has a break between visiting location $i \in \mathcal{N}_v$ and $j \in \mathcal{N}_v$. Furthermore, a break should have a duration of B^D, and be placed in the middle part of the shift. A break center coefficient B_v^C is given for each vehicle $v \in \mathcal{V}$ to indicate the length of the time interval that the break is allowed to start in. For instance, if B_v^C is equal to 0.2, the break must start in the middle 20% of the shift.

4.2.5 Routes

Every customer should be visited by a vehicle that brings the demanded food boxes. To do this, each vehicle in use at a depot should be scheduled a route to drive, originating in the depot for loading, visiting some customers, and ending in the depot at the end of the shift. We use the variable x_{ijv} to indicate that vehicle $v \in \mathcal{V}$

drives directly from location $i \in \mathcal{N}_v$ to location $j \in \mathcal{N}_v$. To keep track of whether a vehicle is in use, the binary variable y_v is used. Some vehicles are mandatory to use because of contracts, which is indicated by R_v that is 1 if vehicle $v \in \mathcal{V}$ is mandatory. To denote when a vehicle $v \in \mathcal{V}$ should arrive at a location $i \in \mathcal{N}_v$, we use the variable t_{iv}. Note that for depots, which are visited both at the start and end of a shift, this variable denotes arrival on the first visit, and thus the start time of the shift. To also keep track of the end time of the shift for vehicle $v \in \mathcal{V}$ we use the variable t_v^E.

4.2.6 Costs

For the vehicle routes, the costs consist of fixed costs of using vehicles, costs per distance unit driven, costs per time unit a vehicle is in use and overtime costs. Fixed costs are paid when using a vehicle during a shift, and include costs given in contracts with drivers and costs for loading the vehicle at the beginning of the shift. These fixed costs are denoted by C_v^F for vehicle $v \in \mathcal{V}$. There are also costs related to the distance that is driven by vehicles, which can include fuel costs and cost of maintenance. Costs per distance unit are denoted by C_v^D for each vehicle $v \in \mathcal{V}$. The distance between two locations $i \in \mathcal{N}$ and $j \in \mathcal{N}$ is given by the parameter A_{ij}. The driver's wages and other time-based costs for driving a vehicle $v \in \mathcal{V}$ is given in C_v^T. Finally, a driver who works overtime should be compensated for the overtime. Extra hourly costs for overtime for the driver of vehicle $v \in \mathcal{V}$ is given by C_v^O. The total overtime for vehicle $v \in \mathcal{V}$ in the shift is represented by the variable a_v.

4.3 Mathematical Model

In this section, the mathematical model for the problem is presented, using the mathematical notation presented in Section 4.1 and 4.2. The structure of a stochastic program with recourse is used. First, we introduce the first-stage problem with respective constraints. Next, we present the second-stage problem that takes into account the first-stage decisions and a realization of the stochastic customer presence. An overview of the mathematical model can also be found in Appendix B.

4.3.1 First-Stage Problem

$$\min z = \sum_{i \in \mathcal{N}^D \setminus \{0\}} G_i \delta_i \tag{4.1a}$$
$$+ G^F \gamma \tag{4.1b}$$
$$+ \mathbb{E}_\xi \left[Q(\delta, \gamma, \xi) \right] \tag{4.1c}$$

The objective function of the first-stage problem is defined by three terms. The costs for each depot is defined by term (4.1a), and term (4.1b) defines the costs of the vehicle fleet size for the depot configuration. Term (4.1c) models the expected operational costs described by the expected value of the recourse function Q. The recourse function Q is the objective value of the solution of the second-stage problem given the depot configuration δ, the fleet size γ and a customer set specified by a realization $\tilde{\xi}$ of ξ.

$$\delta_i + \delta_j \leq 1 + \mathcal{K}_{ij} \qquad i \in \mathcal{N}^D, j \in \mathcal{N}^D \setminus \{i\} \tag{4.2}$$

$$\delta_0 = 1 \tag{4.3}$$

$$\delta_i \in \{0, 1\} \qquad i \in \mathcal{N}^D \tag{4.4}$$

$$\gamma \in \mathbb{Z}_{\geq 0} \tag{4.5}$$

To assure that two alternative depots at the same location cannot be opened simultaneously, constraints (4.2) are used. By defining these constraints for all pairs of different depots, the number of alternative depots at the same location is not restricted. Constraint (4.3) makes sure that the warehouse is always open. Finally, constraints (4.4) restrict the depot opening decision to be binary, and constraint (4.5) forces the fleet size γ to be a non-negative integer.

4.3.2 Second-Stage Problem

The second-stage problem is called by the recourse function in the first-stage problem. It is defined for a given depot configuration δ, a given fleet size γ and a specific realization $\tilde{\xi}$ of ξ. The second-stage problem is similar to the mathematical model from the project report (Gulaker et al., 2022), extended with compatibility for δ, γ and $\tilde{\xi}$. Note that some constraints are formulated non-linearly. This is done to enhance readability and does not affect the implementation of the model as it is not implemented with a commercial solver. All these constraints are possible to linearize using big-M notation, however.

Objective Function

$$Q(\delta, \gamma, \tilde{\xi}) = \min \sum_{v \in \mathcal{V}} C_v^F y_v \tag{4.6a}$$

$$+ \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N}_v} \sum_{j \in \mathcal{N}_v \setminus \{i\}} C_v^D A_{ij} x_{ijv} \tag{4.6b}$$

$$+ \sum_{i \in \mathcal{N}^D} \sum_{v \in \mathcal{V}_i} C_v^T (t_v^E - t_{iv}) \tag{4.6c}$$

$$+ \sum_{v \in \mathcal{V}} C_v^O a_v \tag{4.6d}$$

Term (4.6a) models the fixed costs of using a vehicle. Term (4.6b) is the cost per distance unit driven. Time-based costs are handled by term (4.6c), where the duration of a shift for vehicle $v \in \mathcal{V}_i$ belonging to depot $i \in \mathcal{N}^D$ is found by subtracting the start time of the shift t_{iv} from the end time t_v^E. Finally, term (4.6d) gives the overtime costs.

Flow Constraints

$$\sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N}_v \setminus \{j\}} x_{ijv} = \tilde{F}_j \qquad j \in \mathcal{N}^C \tag{4.7}$$

$$\sum_{j \in \mathcal{N}^C} x_{ijv} - \delta_i y_v = 0 \qquad i \in \mathcal{N}^D, v \in \mathcal{V}_i \tag{4.8}$$

$$\sum_{j \in \mathcal{N}_v \setminus \{i\}} x_{jiv} - \sum_{j \in \mathcal{N}_v \setminus \{i\}} x_{ijv} = 0 \qquad v \in \mathcal{V},\ i \in \mathcal{N}_v \tag{4.9}$$

Constraints (4.7) ensure that all customers that should be visited are visited once. By using $\tilde{F}_j$, no vehicle is allowed to travel to customer j if this customer is not contained in the customer set specified by the realization $\tilde{\xi}$ of ξ. If a vehicle is used, constraints (4.8) make sure it drives out from its depot. The first-stage decision δ_i is used in this constraint to prevent any vehicle from a closed depot to be used. Constraints (4.9) handle that when a vehicle drives to a location, it also drives away from it. Since this includes depot locations, it also makes sure a vehicle returns to the depot if it has driven out from it.

Vehicle Capacity and Usage

$$\sum_{i \in \mathcal{N}^C} \sum_{j \in \mathcal{N}_v \setminus \{i\}} D_i x_{ijv} \leq H_v^V \qquad v \in \mathcal{V} \tag{4.10}$$

$$\sum_{v \in \mathcal{V}} y_v \leq \gamma \tag{4.11}$$

$$y_v \geq R_v \delta_i \qquad i \in \mathcal{N}^D, v \in \mathcal{V}_i \tag{4.12}$$

Constraints (4.10) ensure that the total demand of the customers that a vehicle visits does not exceed the capacity of the vehicle. Constraint (4.11) enforces that the total vehicle fleet size is not exceeded. To make sure that mandatory vehicles are used, constraints (4.12) is in place.

Break Time Constraints

$$(t_v^E - t_{kv})(1 - \sum_{i \in \mathcal{N}_v} \sum_{j \in \mathcal{N}_v \setminus \{i\}} b_{ijv}) \leq T_v^B \qquad k \in \mathcal{N}^D,\ v \in \mathcal{V}_k \tag{4.13}$$

$$b_{ijv} - x_{ijv} \leq 0 \qquad v \in \mathcal{V}, i \in \mathcal{N}_v, j \in \mathcal{N}_v \setminus \{i\} \tag{4.14}$$

$$\frac{1 - B_v^C}{2}(t_v^E - t_{kv}) b_{ijv} \leq t_{iv} \qquad k \in \mathcal{N}^D, v \in \mathcal{V}_k, i \in \mathcal{N}_v, j \in \mathcal{N}_v \setminus \{i\} \tag{4.15}$$

$$\frac{1 + B_v^C}{2}(t_v^E - t_{kv}) b_{ijv} \geq t_{jv} \qquad k \in \mathcal{N}^D, v \in \mathcal{V}_k, i \in \mathcal{N}_v, j \in \mathcal{N}_v \setminus \{i\} \tag{4.16}$$

Constraints (4.13) impose a break if the shift duration exceeds the break threshold and constraints (4.14) ensure that a break is only allowed to be set at a distance that the vehicle drives. Constraints (4.15) enforce that any break starts after the lower limit of the break time interval, whereas constraints (4.16) enforce the upper limit of the break time interval.

Time Constraints

$$(t_{iv} + T_i^S + B^D b_{ijv} + T_{ij} - t_{jv}) x_{ijv} \leq 0 \qquad v \in \mathcal{V}, i \in \mathcal{N}_v, j \in \mathcal{N}^C \setminus \{i\} \tag{4.17}$$

$$(t_{iv} + T_i^S + B^D b_{ijv} + T_{ij} - t_v^E) x_{ijv} \leq 0 \qquad i \in \mathcal{N}^C, j \in \mathcal{N}^D \setminus \{i\}, v \in \mathcal{V}_j \tag{4.18}$$

Constraints (4.17) make sure that the start time of successive customers is set with enough time in between for servicing the customer, driving to the next customer and taking a break if this should be done. Constraints (4.18) do the same for setting the end time of the shift.

$$\underline{T}_j^N \leq t_{jv} \leq \overline{T}_j^N \qquad i \in \mathcal{N}^D, v \in \mathcal{V}_i,\ j \in \mathcal{N}_v \setminus \{i\} \tag{4.19}$$

$$t_{iv} \geq \underline{T}_v^V \qquad i \in \mathcal{N}^D, v \in \mathcal{V}_i \tag{4.20}$$

$$t_v^E \leq \overline{T}_v^V \qquad v \in \mathcal{V} \tag{4.21}$$

Constraints (4.19) ensure all location time windows are respected. Vehicle availability time windows are handled by constraints (4.20) and constraints (4.21).

$$t_v^E - t_{iv} \leq T_v^D \qquad i \in \mathcal{N}^D,\ v \in \mathcal{V}_i \tag{4.22}$$

$$t_v^E - t_{iv} - a_v \leq T_v^O \qquad i \in \mathcal{N}^D,\ v \in \mathcal{V}_i \tag{4.23}$$

The maximum working shift duration is enforced by constraints (4.22). To count overtime correctly, constraints (4.23) are in place.

Dispatch Constraints

$$\sum_{\tau \in \mathcal{T}} \underline{T}^I_\tau k_{v\tau} - t_{iv} \leq 0 \qquad i \in \mathcal{N}^D,\ v \in \mathcal{V}_i \tag{4.24}$$

$$(t_{iv} + T^S_{iv} - \sum_{\tau \in \mathcal{T}} \overline{T}^I_\tau e_{v\tau}) y_v \leq 0 \qquad i \in \mathcal{N}^D,\ v \in \mathcal{V}_i \tag{4.25}$$

$$\sum_{\tau \in \mathcal{T}} k_{v\tau} - y_v = 0 \qquad v \in \mathcal{V} \tag{4.26}$$

$$\sum_{\tau \in \mathcal{T}} e_{v\tau} - y_v = 0 \qquad v \in \mathcal{V} \tag{4.27}$$

$$k_{v\tau} + d_{v\tau} - d_{v(\tau+1)} - e_{v(\tau+1)} = 0 \qquad v \in \mathcal{V},\ \tau \in \mathcal{T} \backslash \{|\mathcal{T}|\} \tag{4.28}$$

$$\sum_{v \in \mathcal{V}_i} (k_{v\tau} + d_{v\tau} + e_{v\tau}) \leq L_i \qquad i \in \mathcal{N}^D,\ \tau \in \mathcal{T} \tag{4.29}$$

Constraints (4.24) set $k_{v\tau}$ to 1 if loading in depot $i \in \mathcal{N}^D$ of vehicle $v \in \mathcal{V}_i$ starts within time interval $\tau \in \mathcal{T}$. Constraints (4.25) set $e_{v\tau}$ to 1 if loading on depot $i \in \mathcal{N}^D$ of vehicle $v \in \mathcal{V}_i$ ends within time interval $\tau \in \mathcal{T}$ and the vehicle is used. Constraints (4.26) ensure that only one start timeslot is set if vehicle $v \in \mathcal{V}$ is in use, while constraints (4.27) ensure that only one end timeslot is set if vehicle $v \in \mathcal{V}$ is in use. If loading of vehicle $v \in \mathcal{V}_i$ happens during the whole interval, constraints (4.28) set $d_{v\tau}$ to 1. By summing all $k_{v\tau}$, $d_{v\tau}$ and $e_{v\tau}$ for each timeslot $\tau \in \mathcal{T}$ on a depot $i \in \mathcal{N}^D$, constraints (4.29) restrict the number of vehicles that can be loaded simultaneously.

Binary and Non-Negativity Constraints

$$x_{ijv} \in \{0,1\} \qquad v \in \mathcal{V},\ i \in \mathcal{N}_v,\ j \in \mathcal{N}_v \backslash \{i\} \tag{4.30}$$

$$y_v \in \{0,1\} \qquad v \in \mathcal{V} \tag{4.31}$$

$$b_{ijv} \in \{0,1\} \qquad v \in \mathcal{V},\ i \in \mathcal{N}_v,\ j \in \mathcal{N}_v \backslash \{i\} \tag{4.32}$$

$$t_{iv} \geq 0 \qquad v \in \mathcal{V},\ i \in \mathcal{N}_v \tag{4.33}$$

$$a_v \geq 0 \qquad v \in \mathcal{V} \tag{4.34}$$

$$k_{v\tau} \in \{0,1\} \qquad v \in \mathcal{V},\ \tau \in \mathcal{T} \tag{4.35}$$

$$e_{v\tau} \in \{0,1\} \qquad v \in \mathcal{V},\ \tau \in \mathcal{T} \tag{4.36}$$

$$d_{v\tau} \in \{0,1\} \qquad v \in \mathcal{V},\ \tau \in \mathcal{T} \tag{4.37}$$

At last, we ensure the binary and non-negative properties of variables with constraints (4.30) to (4.37).

Chapter 5

Solution Methods

This chapter explains the solution methods we suggest to solve the LRP. We first give an overview of the solution methods, including how the stochastic parameters are handled. The explanation of the solution methods is divided into two sections, one for each stage of the problem. The depot location search in the first stage is explained in Section 5.1, whereas the routing algorithm in the second stage is explained in Section 5.2. For the second-stage problem, we start by defining simplifications to the mathematical problem that are used by the routing algorithm in Section 5.2.1.

We discretize the probability distribution of the stochastic customer presence ξ using scenarios. For each scenario $s \in \mathcal{S}$, we define the realization $\tilde{\xi}_s$ of ξ, with probability p_s. The expectation in the first-stage objective function can then be written as in equation (5.1). The stochastic parameters contained in $\tilde{\xi}_s$ are reformulated to $\tilde{F}_{is}$ for all customers $i \in \mathcal{N}^C$ and scenarios $s \in \mathcal{S}$. More details on how the scenarios are generated and tested are presented in Chapter 6 and Section 7.1, respectively.

$$\mathbb{E}_\xi \left[Q(\delta, \gamma, \xi) \right] = \sum_{s \in \mathcal{S}} p_s Q(\delta, \gamma, \tilde{\xi}_s) \tag{5.1}$$

The solution methods use a combination of heuristics and decomposition to find solutions to the problem. A solution to the LRP means finding values for variables in the first-stage and second-stage problems. However, the depot configuration δ and fleet size γ in the first-stage problem are the most interesting parts of the solution as they answer the strategic questions. Solutions to the second-stage problem found in this location are not used in actual operational planning. Figure 5.1 gives an overview of the solution methods. To start the search, an initial depot configuration must be provided. The selected depot configuration and the customer realizations from the scenarios are used to create a set of routing problems. Each of these routing problems is solved by first decomposing the problem into single-depot problems by creating districts for each depot, then clustering the customers in each district to super-customers, and finally using HGSR to construct the routes. After solving each single-depot problem separately, the results are merged into a solution to the original problem. The routing cost of the depot configuration is computed according to equation (5.1). The state of the search and the solution is updated based on the total cost of the depot configuration. The search continues until the stopping criterion is met.

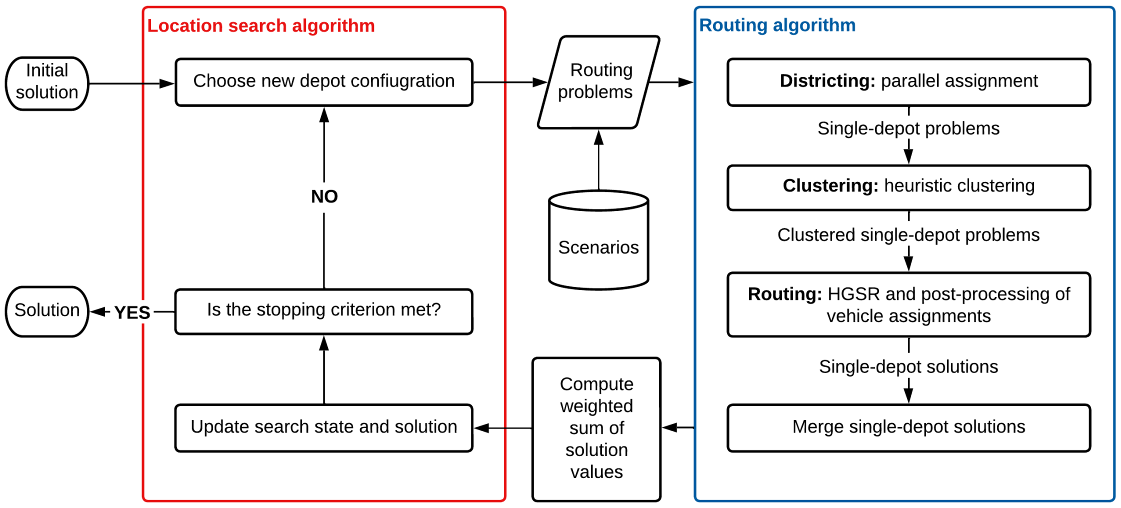
Location search algorithm
Initial solution
Choose new depot confiugration
NO
Solution
YES
Is the stopping criterion met?
Update search state and solution
Routing problems
Scenarios
Compute weighted sum of solution values
Routing algorithm
Districting: parallel assignment
Single-depot problems
Clustering: heuristic clustering
Clustered single-depot problems
Routing: HGSR and post-processing of vehicle assignments
Single-depot solutions
Merge single-depot solutions

Figure 5.1: Overview of the solution methods.

5.1 The Depot Location Search

Because the possible combinations of depots grow $O(2^{|\delta|})$, testing all possible depot combinations quickly becomes intractable. Therefore, a heuristic search through depot configurations is necessary. We start by explaining how candidate configurations are made through neighborhood generation in Section 5.1.1. Then, we define the search algorithms CATS and GRASP in Section 5.1.2 and Section 5.1.3, respectively.

5.1.1 Neighborhood

The process of searching through depot configurations relies on the concept of neighborhood operators. When a neighborhood operator is applied to a selected depot configuration, a set of candidate configurations is generated. This set is called a neighborhood of the selected depot configuration, and each candidate configuration is called a neighbor. Going from one selected depot configuration to selecting one of its neighbors instead is referred to as a move. We say that a configuration is "explored" when it is evaluated in order to decide whether to move to it or not. An explored configuration that the search decides to move to is called a "visited" configuration. In our problem, the search space for the depot location search is all possible depot configurations.

Recall from Section 4.1 that a depot configuration is a set of open and closed depots described by the vector of binary variables δ. For the depot location search, we consider the neighborhood operators in Table 5.1, which are inspired by Klibi et al. (2010). The Open operator opens a closed depot. Likewise, the Close operator closes an open depot. The Open and Close operators are commonly known as flip moves, and assure that the search space is connected as all possible depot configurations can be reached by using these two neighborhood operators. We also use a Swap operator, where one depot closes and another opens simultaneously. This allows better exploration at the same number of depots, as it reduces the number of local optima. Figure 5.2 shows the effect of each neighborhood operator on a depot configuration. With only Open and Close operators, there might be situations where it is neither improving to use the Open nor the Close operator, indicating that the search has found a local optimum using these operators. However, there could be improving configurations with the same number of depots open found in the Swap neighborhood. By introducing the Swap operator, the previous depot configuration would therefore no longer be a local optimum, allowing the search algorithm to continue finding improving solutions. However, introducing the Swap operator comes at the cost of increasing the size of the neighborhood from $O(|\delta|)$ to $O(|\delta|^2)$. To limit the neighborhood size, no additional operators that change two or more depots simultaneously are used.

Table 5.1: Neighborhood operators for the depot location search.

Operator name	Description
Open	Open a depot
Close	Close a depot
Swap	Both open a depot and close a depot

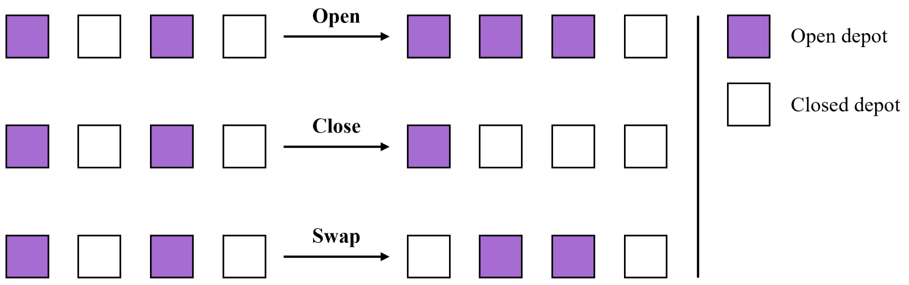

Figure 5.2: Possible effects of the neighborhood operators on a depot configuration.

We define a neighborhood generator to be a function $N(\delta)$ that returns a neighborhood of δ, and which can use either of the neighborhood operators depending on the underlying strategy of the generator. A naive way of generating neighborhoods is to apply all three operators and randomly choose a neighbor from the total neighborhood, which we refer to as the random neighborhood generator $N^R(\delta)$. This generator has some obvious weaknesses. First, we might know that some operators are less likely to make an improvement dependent on which operators have been tried before and how many depots are open. Second, the probability of picking a neighbor that has been generated by using Swap is much larger than that of Open and Close because the number of combinations of changing two depots grows much faster than the number of combinations from changing only one depot. Therefore, to improve the efficiency and effectiveness of the search through depot configurations we propose an alternative that consists of three alternative neighborhood generators: the constructive $N^C(\delta)$, the adaptive $N^A(\delta)$, and finally the combination of the constructive and adaptive $N^{CA}(\delta)$.

The constructive neighborhood generator $N^C(\delta)$ has three consecutive phases, each one using only one neighborhood operator. In each phase, neighbors in the neighborhood generated by the operator are explored, moving to the neighbor if it is improving. The parameter φ^C limits the number of neighbors that can be explored without any improvement before going to the next phase. Whenever an improving neighbor is found before reaching the limit, the counter is reset and φ^C new neighbors can be explored using the same operator as before. In the first phase, neighbors are explored by opening depots using the Open operator. When φ^C non-improving neighbors have been explored, the generator uses the Close operator to try to close depots. Again, when the limit is reached, the generator enters its final phase where only the Swap operator is used to generate new neighbors. The intuition is that the constructive neighborhood generator first finds the correct number of open depots in phases 1 and 2, after which the task is only to find which depots at this number should be open.

The adaptive neighborhood generator $N^A(\delta)$ is based on Pisinger and Ropke (2007). They implement a scoring system that gives each operator a reward of ω_h based on the result h of applying the operator. Operators receive the reward of ω_1 when they find a global best solution and ω_2 when they find a locally improved solution. Otherwise, they do not receive a reward. $\omega_0 = 0$ is used for these cases. We use a decay rate θ that reduces the score of the operator each time it is used. The score of operator M for usage number i, denoted s_i^M, is thus updated according to equation (5.2) when it receives reward ω_h. The probability of choosing an operator is calculated by dividing each operator's score by the sum of the scores. When using an operator does not lead to a move, decaying its score effectively increases the probability of using the other two operators. All operators start with a score s_0. Whenever $N^A(\delta)$ is used, an operator is drawn randomly with this probability of being chosen. If all neighbors in an operator neighborhood have been explored, the probability of choosing this operator is set to 0. Neighbors are then drawn from neighborhoods generated by the other operators until all neighborhoods have been explored or a move is made to another solution. If a move is made, the probabilities are again based on each operator's score.

$$s_i^M = \theta(s_{i-1}^M + \omega_h) \tag{5.2}$$

Both the constructive and the adaptive neighborhood generators make some assumptions about the search space. $N^C(\delta)$ assumes that there exists a path of improving moves from the starting depot configuration through the search space to a configuration with the correct number of open depots by using only Open and Close for a limited number of tries. This is a strong assumption, as local optima might exist which traps the search at the wrong number of open depots, and the number of tries might be too few. $N^A(\delta)$ assumes that successful and unsuccessful operators in the past can be used to determine which operators to use in the future. However, if the starting configuration has too few or too many open depots, the search could be heavily biased towards trying the Open and Close operators when reaching the number of open depots that is optimal, although Swap should optimally be used. To use the strengths of $N^A(\delta)$ and mitigate this bias, we combine $N^C(\delta)$ and $N^A(\delta)$ into the combined neighborhood generator $N^{CA}(\delta)$, which is the neighborhood generator used by the proposed CATS algorithm. $N^{CA}(\delta)$ starts by using the Open phase of $N^{C}(\delta)$ and then switches to $N^{A}(\delta)$ after the Open phase has finished. This means the adaptive generator can start to search around a promising number of open depots without being influenced by the path through the solution space to get there. $N^{CA}(\delta)$ is more flexible and can potentially correct situations such as if $N^{C}(\delta)$ has found four depots to be the correct number, but only because it did not discover the best depot configuration with three depots. In the adaptive phase, we bias towards Swap by adding a bias s^{B} to the start score of Swap. A possible path between the depot configurations found in the search space for $N^{CA}(\delta)$ is illustrated in Figure 5.3.

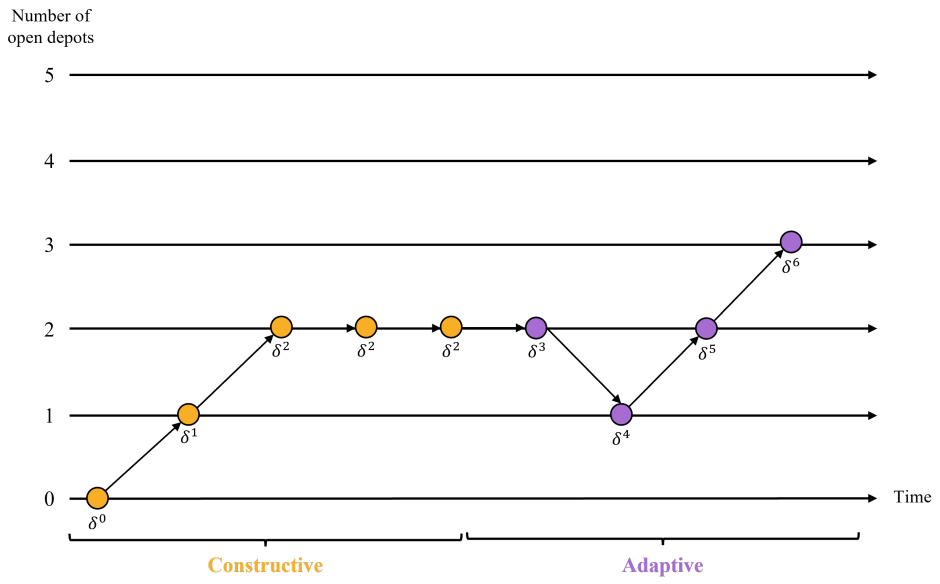

Figure 5.3: A possible path between the best solutions found with the combined neighborhood generator $N^{CA}(\delta)$ and maximum opening failures equal to 2.

It is beneficial that the most promising neighbor solutions are investigated first. This increases the chance for the constructive neighborhood generator to successfully reach a good level of open depots with a lower number of tries φ^{C}, and it might speed up the learning process of the adaptive neighborhood generator. We assume that depot configurations with open depots in close proximity to the customers are more promising. For each depot configuration δ in the neighborhood we therefore calculate the mean depot distance $A^{D}_{s}(\delta)$ for each scenario $s \in \mathcal{S}$, which is the average distance from the customers in the scenario to their closest open depot in δ. When using any of the neighborhood generators except for the random approach, the neighbor depot configurations are then sorted by the lowest expected mean depot distance $\mathbb{E}(A^{D}(\delta))$, as defined in equation (5.3). This is the weighted average of $A^{D}_{s}(\delta)$ for all scenarios $s \in \mathcal{S}$, weighted by the scenario weight p_s.

$$\mathbb{E}(A^{D}(\delta)) = \sum_{s \in S} p_s A^{D}_{s}(\delta) \tag{5.3}$$

5.1.2 Tabu Search

As described in Chapter 3, Tabu Search is a much-used metaheuristic based on local search for solving LRPs. Tabu Search is introduced in Glover (1986) and uses the concept of a tabu list to guide the search when a local optimum has been found. The tabu list describes forbidden solutions that cannot be visited, either because they have been visited before or because a component of the solution has been used before. Usually, Tabu Search is a best-improvement search, where moves are made to the best improving neighbor (Gendreau and Potvin, 2019). However, when Tabu Search finds a local optimum, a non-improving move to a neighboring solution that is not tabu is allowed. To prevent immediately returning to the previous solution, the tabu list is required. A tabu tenure decides how long a solution should be in the tabu list before it is removed, for example after a number of search iterations. The search can be stopped when reaching a time limit or a predetermined number of iterations without improvement.

To curb computational time, a first-improvement Tabu Search is implemented. In a first-improvement search, the first non-tabu neighbor explored that is better than the current solution is moved to immediately. This is contrary to the best-improvement search, where all neighbors are evaluated and the best one is picked, which usually leads to longer computational times. There are several reasons why we use the first-improvement version of the tabu search. First, the search space in our problem is very large as it grows exponentially with the number of depots and linearly with the number of scenarios. Second, given the complexity of the problem, the evaluation of each solution takes some time. Evaluating all neighbors would therefore be very time-consuming. To ensure that the search terminates, a time limit is used to stop the search. Because the algorithm should preferably explore many different neighborhoods before hitting the time limit, it is beneficial to move to the first improving neighbor.

Algorithm 1 contains pseudocode for the first-improvement tabu search metaheuristic. A set of tabu solutions X, a starting solution and a neighborhood generator are required by the algorithm. The objective function z, defined in Section 4.3.1, is used to evaluate the quality of a solution. In line 7, a counter is defined for the number of solutions δ^w that are explored. The search itself starts with the loop in line 8. For each iteration, a new neighbor solution in $N(\delta')$ is explored until either an improving solution is found or all neighbors are evaluated. The expected routing cost is computed in line 10. The $|S|$ routing problems that must be solved are independent of each other and are therefore solved in parallel. In line 11, the fleet size γ^w is set to the maximum number of vehicles required by the depot configuration for all scenarios. The total depot cost is computed in line 12. Lines 13 and 14 check whether the solution is non-tabu and improves the current best solution. If this is the case, the current solution is updated in line 15. The best solution found so far is updated in lines 16 to 17. When the search reaches a local optimum because no neighboring solution improves the current solution, and there is still time left, a jump to a random non-tabu solution with the same number of open depots is performed. This jump is done to allow the search to reach an unseen configuration and remains at the same number of open depots because no improving move to more or fewer depots has been found. Before the next iteration, the set of tabu solutions must be updated in line 20. When the stopping criterion is met, the best solution found, (δ^*, γ^*), is returned.

Algorithm 1 First-Improvement Tabu Search

```
1: $X \leftarrow$ set of tabu solutions, initially containing the starting solution
2: $(\delta', \gamma') \leftarrow$ current solution with objective value $z'$, initially starting solution
3: $(\delta^*, \gamma^*) \leftarrow$ best solution found with objective value $z^*$, initially starting solution
4: $N \leftarrow$ neighborhood generator
5:
6: function TabuSearch
7:     $w \leftarrow 0$
8:     while stopping criterion not met do
9:         $\delta^w \leftarrow$ neighbor solution from $N(\delta')$ that has not been explored
10:        $E^w \leftarrow \sum_{s \in \mathcal{S}} p_s \cdot$ Routing$(\delta^w, \tilde{\xi}_s)$
11:        $\gamma^w \leftarrow$ maximum number of vehicles required by $\delta^w$ over all scenarios
12:        $z^w \leftarrow \sum_{i \in \mathcal{N}^D \setminus \{0\}} G_i \delta_i^w + G^F \gamma^w + E^w$
13:        if $\delta^w \notin X$ then
14:            if $z^w \leq z'$ then
15:                update current solution, setting $(\delta', \gamma') = (\delta^w, \gamma^w)$ and $z' = z^w$
16:                if $z^w < z^*$ then
17:                    update best solution, setting $(\delta^*, \gamma^*) = (\delta^w, \gamma^w)$ and $z^* = z^w$
18:        if explored all neighbors in $N(\delta')$ without any improving move then
19:            $(\delta', \gamma') \leftarrow$ random $\delta \notin X$ with same number of open depots as $\delta'$
20:        update tabu solutions in X
21:        $w \leftarrow w + 1$
22:    return $(\delta^*, \gamma^*)$ with $z^*$
```

We implement three tabu lists based on Klibi et al. (2010). The set of tabu solutions X in Algorithm 1 contains all solutions that are present in any of the tabu lists. One of the tabu lists contains all previously visited depot configurations. When a move is made to a depot configuration, the search prohibits further visits to this configuration by having no tabu tenure for the visit tabu list. The two other lists are short-term as they both have their own tabu tenures. These lists are based on the moves that recently have been used to explore a new neighbor, and we therefore call them the flip and swap tabu list. When δ^w is a neighbor in $N(\delta')$ that has been generated using a flip operator (Open or Close), the depot δ_i^w that has been flipped cannot be flipped again in α iterations. Hence, all depot configurations δ with δ_i not equal to δ_i^w are added to the flip tabu list and removed in iteration $w + \alpha$. On the other hand, when Swap is used, the pair of depots i and j that has been swapped (δ_i^w, δ_j^w) is prohibited to be swapped again in β iterations. Note that swaps do not add any configurations to the flip tabu list. We set α and β by using the number of depots that are open in the depot configuration the neighbor was generated from, as described in equations (5.4), which is based on Nagy and Salhi (1996) and Klibi et al. (2010).

$$\alpha = \lfloor \frac{1}{2} \sum_{i \in \mathcal{N}^D} \delta'_i \rfloor \quad \text{and} \quad \beta = 2 \sum_{i \in \mathcal{N}^D} \delta'_i \tag{5.4}$$

We define CATS to be the first-improvement Tabu Search with the combined neighborhood generator $N^{CA}(\delta)$, neighborhood sorting and all the proposed tabu lists. We expect that the combined neighborhood generator together with neighborhood sorting enables good solutions with a promising number of open depots to be found fast in the constructive phase, and that the adaptive phase contributes to a balance between diversification and intensification further on in the search. The visit tabu list prevents the same depot configuration from being visited again later, thus avoiding parts of the solution space that have already been explored. If the flip of a specific depot recently has been tried, we do not want to flip this depot again for some time. This is because depots that recently resulted in an improvement are probably also good in nearby solutions, and the opposite is also probable. The same reasoning holds for swaps. Using the flip and swap tabu lists assures this. To assess the impact of the different components of CATS in Section 7.4, we add the option to disable each of them. As shown in Table 5.2, the first-improvement Tabu Search with all components disabled except the visit tabu list is referred to as Basic Tabu Search (BTS), which is used as a baseline algorithm for comparison.

Table 5.2: Differences between the Basic Tabu Search (BTS) and Constructive Adaptive Tabu Search (CATS). $N^R(\delta)$ and $N^{CA}(\delta)$ are the random and combined neighborhood generator, respectively.

Component	BTS	CATS
Neighborhood generator	$N^R(\delta)$	$N^{CA}(\delta)$
Neighborhood sorting	✗	✓
Visit tabu list	✓	✓
Flip and swap tabu lists	✗	✓

5.1.3 Greedy Randomized Adaptive Search Procedure

Our implementation of GRASP, based on Feo and Resende (1995), is shown in Algorithm 2. In line 1, the scenario that is used for the single-scenario evaluation is chosen. This could be any of the scenarios, but for the procedure to be most efficient, it should be chosen so that the costs are a consistent proxy for the full evaluation in lines 24 to 26. Therefore, we use the one with the number of customers closest to what is expected. In line 2, we define (δ^*, γ^*) with objective value z^* as the best solution found. For simplicity, we define a helper function SINGLEEVAL(δ) in lines 4 to 8, which returns the single-scenario objective value $\hat{z}$ of a depot configuration

δ. The function for GRASP is defined in lines 10 to 29, which starts by the outer loop in line 11 that is controlled by a stopping criterion, either in terms of a number of iterations or a maximum running time. In each iteration of this outer loop, the GRASP restarts its search. In lines 12 to 14, a counter w is initialized and is used when referring to the candidate depot configuration δ^w with single-scenario evaluation $\hat{z}^w$. The current depot configuration δ' with corresponding single-scenario evaluation $\hat{z}'$ are initialized in line 15. $\hat{z}'$ is set to infinity to start the loop in line 16.

In the inner loop in lines 16 to 23, GRASP searches for a new candidate depot configuration δ^w that improves the current depot configuration δ'. The inner loop is repeated until the new candidate depot configuration is not an improvement of the current depot configuration. In line 17, δ' and $\hat{z}'$ are updated because δ^w is better. Then, in lines 18 to 21, the next candidate depot configuration δ^{w+1} is determined. At this point, GRASP is first greedy by generating a candidate list containing the open neighbors of δ' in line 18, all of which are evaluated in line 19 with the single scenario, and reduced to a restricted candidate list R in line 20 based on some criteria. We use the criterion that it should include the configurations with a cost no higher than $\alpha\%$ of the best evaluation in L^{eval}, also called a value restriction (Feo and Resende, 1995). Afterward, GRASP is random in line 21 by choosing δ^{w+1} to be a random depot configuration of R. In line 22, the single-scenario evaluation $\hat{z}^{w+1}$ of δ^{w+1} is retrieved from L^{eval}. The counter is incremented in line 23.

When the candidate depot configuration no longer is an improvement of the current depot configuration, δ' must be the best configuration in this iteration. In lines 24 to 26, δ' is evaluated with respect to all the scenarios. This includes finding the expected routing cost E', determining the vehicle fleet γ' based on the routing solutions and finally computing the solution value z'. The solution value in this iteration z' is compared with the best solution value over all previous iterations z^* in line 27, and the best solution is updated in line 28. When the stopping criterion in line 11 is met, the best solution found, (δ^*, γ^*), is returned in line 29.

Recall from Section 3.4 that GRASP usually involves two phases. The initial phase creates diverse promising solutions and the second phase, or intensification phase, is used to improve this solution. We do not include the intensification phase because we do not use a local search around configuration δ' in each iteration. This local search takes time, which would have reduced the number of new solutions explored. In our implementation of GRASP, we prioritize diversification to explore more of the solution space, thus distinguishing it further from the Tabu Search. Furthermore, using only one scenario enables GRASP to efficiently consider many different depot configurations as the new candidate δ^{w+1} because the evaluation in line 19 can be run in parallel.

Algorithm 2 GRASP

```
1: $\hat{s} \leftarrow$ scenario to use for single-scenario evaluation
2: $(\delta^*, \gamma^*) \leftarrow$ best solution found with objective value $z^*$, initially none with $z^* = \infty$
3:
4: function SingleEval($\delta$)
5:     $\hat{E} \leftarrow$ Routing($\delta, \tilde{\xi}_{\hat{s}}$)
6:     $\hat{\gamma} \leftarrow$ maximum number of vehicles required by $\delta$ in scenario $\hat{s}$
7:     $\hat{z} \leftarrow \sum_{i \in \mathcal{N}^D \setminus \{0\}} G_i \delta_i + G^F \hat{\gamma} + \hat{E}$
8:     return $\hat{z}$
9:
10: function GRASP
11:     while stopping criterion not met do
12:         $w \leftarrow 0$
13:         $\delta^w \leftarrow$ candidate depot configuration, initially with no open depots
14:         $\hat{z}^w \leftarrow$ SingleEval($\delta^w$)
15:         $\delta', \hat{z}' \leftarrow \delta^w, \infty$
16:         while $\hat{z}^w < \hat{z}'$ do
17:             update current depot configuration, setting $\delta' = \delta^w$ and $\hat{z}' = \hat{z}^w$
18:             $L \leftarrow$ candidate list containing all open neighbors of $\delta'$
19:             $L^{eval} \leftarrow \{$SingleEval($\delta$) $\mid \delta \in L\}$
20:             $R \leftarrow$ restricted candidate list of $L$ based on criterion and $L^{eval}$
21:             $\delta^{w+1} \leftarrow$ randomly chosen depot configuration of $R$
22:             $\hat{z}^{w+1} \leftarrow$ single-scenario evaluation from $L^{eval}$ corresponding to $\delta^{w+1}$
23:             $w \leftarrow w + 1$
24:         $E' \leftarrow \sum_{s \in \mathcal{S}} p_s \cdot$ Routing($\delta', \tilde{\xi}_s$)
25:         $\gamma' \leftarrow$ maximum number of vehicles required by $\delta'$ over all scenarios
26:         $z' \leftarrow \sum_{i \in \mathcal{N}^D \setminus \{0\}} G_i \delta'_i + G^F \gamma' + E'$
27:         if $z' < z^*$ then
28:             update best solution, setting $(\delta^*, \gamma^*) = (\delta', \gamma')$ and $z^* = z'$
29:     return $(\delta^*, \gamma^*)$ with $z^*$
```

5.2 The Routing Algorithm

To estimate the cost of a depot location, we are solving the second-stage routing problem for a set of scenarios. For doing this efficiently, we use a routing algorithm based on the algorithm proposed by Gulaker et al. (2022) to find estimates for the routing costs. This algorithm divides the routing process into three phases: districting, clustering and routing. An overview of these phases is shown in Figure 5.4.

By districting, each customer is assigned to a depot, thus splitting the multi-depot VRP into a set of single-depot VRPs called districts. Similar customers in each district are then clustered into super-customers. Finally, routes are created between these super-customers. The districting, clustering and routing are explained in more detail in Section 5.2.2, 5.2.3 and 5.2.4, respectively.

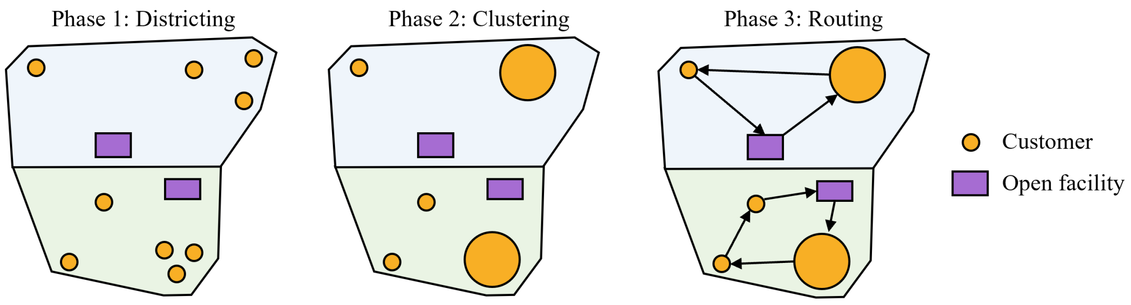

Figure 5.4: Overview of the phases in the routing algorithm.

Algorithm 3 gives a high-level pseudocode of the routing algorithm. Phase-specific parameters are given in lines 1 to 3. The routing algorithm takes as input a depot configuration δ^w and a scenario $\tilde{\xi}_s$. To simplify the notation when explaining the routing algorithm in this section, $\hat{\mathcal{N}}^C$ is the set of customers that are present in the realization $\tilde{\xi}_s$, and $\hat{\mathcal{N}}^D$ is the set of open depots in configuration δ^w. These are defined in lines 8 and 9. Customers are first partitioned into districts by using Algorithm 4 in line 10. Then, for each district, customers are clustered by using Algorithm 5 in line 12. In line 13, the routing cost for each single-depot problem j is assigned to Q_j. The total cost is calculated in line 14 and then returned in line 15.

Algorithm 3 The Routing Algorithm

```
1: $\rho \leftarrow$ minimum rest capacity
2: $l^{max} \leftarrow$ max distance
3: $t^{max} \leftarrow$ max waiting time
4:
5: function ROUTING($\delta^w, \tilde{\xi}_s$)
6:     input: $\delta \leftarrow$ depot configuration $w$
7:     input: $\tilde{\xi}_s \leftarrow$ scenario containing stochastic parameters $\tilde{F}_{is}$
8:     $\hat{\mathcal{N}}^C \leftarrow \{i \in \mathcal{N}^C \mid \tilde{F}_{is} = 1\}$
9:     $\hat{\mathcal{N}}^D \leftarrow \{j \in \mathcal{N}^D \mid \delta^w_j = 1\}$
10:    $W_j \leftarrow$ customers in district $j \in \hat{\mathcal{N}}^D$ returned from DISTRICTING($\rho, \hat{\mathcal{N}}^C, \hat{\mathcal{N}}^D$)
11:    for district $j \in \hat{\mathcal{N}}^D$ do
12:        $W'_j \leftarrow$ clustered $W_j$ returned from CLUSTERING($l^{max}, t^{max}, W_j$)
13:        $Q_j \leftarrow$ routing cost for single-depot problem $j$ with customers $W'_j$
14:    $Q \leftarrow \sum_{j \in \hat{\mathcal{N}}^D} Q_j$
15:    return $Q$
```

5.2.1 Simplifications of the Vehicle Routing Problem

The VRP that is defined as the second-stage problem in Chapter 4 represents operational decisions made on a daily basis. In contrast, the depot location decision in the first-stage problem is a strategic decision. Therefore, there are certain aspects of the VRP that are not that relevant when the decision we are interested in is strategic. Furthermore, each constraint that the operational VRP takes into account adds a new layer of complexity. This leads to a trade-off between computational time and accuracy, which is of particular interest in this thesis as we need to solve the second-stage problem fast while still retaining reliable estimates of the routing costs. This section outlines some simplifications of the mathematical model from Chapter 4. Based on the simplifications, a transition from the operational problem to the strategic problem solved by the solution methods is defined. The simplifications are summarized in Table 5.3. In addition to the simplifications shown in the table, we also add a penalty to avoid infeasibility.

Table 5.3: Simplifications of the operational mathematical model.

Constraints	Description	Operational	Strategic
(4.7), (4.8), (4.9)	Flow constraints	✓	✓
(4.10)	Vehicle capacity	Heterogeneous	Homogeneous
(4.11)	Fleet size	✓	✓
(4.12)	Mandatory vehicles	✓	✗
(4.13), (4.14), (4.15), (4.16)	Break time constraints	✓	✗
(4.17), (4.18), (4.19)	Time windows	✓	✓
(4.20), (4.21)	Vehicle availability time	✓	✗
(4.22), (4.23)	Tour duration and overtime	✓	✓
(4.24), (4.25), (4.26), (4.27), (4.28), (4.29)	Dispatch constraints	✓	✗

Because the vehicle fleet that is available in a given depot configuration is limited, it can cause the problem to be infeasible when there are not enough vehicles to serve all customers. This is inconvenient for several reasons. First, local search based methods do not handle infeasible solutions well. For instance, we could represent infeasible solutions with an infinite cost, but then the search cannot tell by how much a solution is infeasible. Second, a hard limit of vehicles is not necessarily a good representation of reality. In the long run, it would be possible to acquire more vehicles. In the short term, an external car could be hired at a higher cost, or a storage worker could drive a car if the number of drivers was limited. Third, if the vehicle fleet was in fact limited, one could cancel some customer orders or deliver them late. Taking into account all these possibilities would make our problem too complex to solve quickly enough. Consequently, in order to avoid infeasibility in a simpler way, we add the ability to extend the vehicle fleet at an extra cost. This is done by extending the warehouse fleet and adding a penalty cost for using more vehicles than it has available in the original problem. Specifically, we define a parameter $\overline{\mathcal{V}_0}$ as the number of vehicles originally available at the warehouse, and extend the list $\mathcal{V}_0$ of vehicles available there until the warehouse can serve all customers in every scenario. Hence, all solutions will be feasible. A fixed penalty of C^P is added to the objective function of the routing problem for the number of vehicles used that exceeds $\overline{\mathcal{V}_0}$. This penalty represents the additional costs of extending the warehouse fleet with one vehicle in a specific shift.

According to the mathematical model in Chapter 4, the vehicles are heterogeneous with respect to vehicle capacity, availability time windows and cost structures. It is assumed that a fleet with homogeneous capacity is a sufficient representation when considering decisions in the long term. One can also argue that this is beneficial because a fleet with different vehicle models might cause higher maintenance costs and less flexibility for redistribution between depots. With this simplification, the vehicle capacity H_v^V is considered as a constant H^V for all vehicles $v \in \mathcal{V}$. Furthermore, on a daily basis, some vehicles are set mandatory to assure that the employed drivers are assigned to a route. There are also non-mandatory vehicles that can be assigned to a route on short notice if necessary. Setting this specific threshold between the amount of mandatory and non-mandatory vehicles is harder in the long run. Also, this is a decision that is yet to be made, meaning that this threshold is more flexible in the long run. We therefore make all vehicles in the fleet non-mandatory by setting the right-hand side of constraints (4.12) to zero.

To reduce complexity, we assume that the constraints on vehicle availability can be relaxed. In daily operations at Oda, the vehicle availability time windows are set with some differences to make the shifts predictable for drivers. However, these variations are so small and of practically no significance in the long term, therefore we disregard them. Consequently, constraints (4.20) and (4.21) are made non-binding by changing $\underline{T}_v^V$ and $\overline{T}_v^V$ to the start and end of the time window for the depot in which vehicle v belongs.

The operational routing problem includes constraints to ensure that the drivers receive their breaks during the shifts. We interpret these regulations as more significant for the day-to-day operation, and not that important when estimating the costs at a strategic level. Therefore, the break time constraints (4.13), (4.14), (4.15) and (4.16) are not considered by the solution methods. This means that the vehicles have some more time to utilize within the shift and that the payment during the breaks is not apparent in the estimated routing costs.

The dispatch constraints control how much time each vehicle needs at the depot, either in a queue to be loaded or at the loading dock, before the route can start. The exact prioritization for dispatch is interconnected to several other features of the problem, making this a complex decision in itself. Furthermore, as with the vehicle availability constraints, we assume that the time added to the routes from the dispatch constraints is small compared to the total duration of the routes. These small adjustments in loading time for vehicles should therefore have little impact on the total costs. To disregard the dispatch constraints, the number of loading docks L_i is set equal to $|\mathcal{V}_i|$ for all depots $i \in \mathcal{N}^D$, meaning that all vehicles can be loaded simultaneously.

5.2.2 Districting

The parallel assignment algorithm proposed by Giosa et al. (2002) is used for districting. Customers are assigned to the closest available depot, in the order of a measure called urgency. Urgency defines a precedence relationship between the customers, where customers with higher urgency are assigned to a depot first. The urgency is calculated as shown in equations (5.5). An assignment cost C_{ij} from each customer i to each depot j is calculated, which is the cost of traveling directly from the customer to the depot including both time and distance costs. From these, the urgency μ_i for each customer i is found by comparing the cost of assigning the customer to its closest depot with the cost of assigning it to any other depot.

$$\mu_i = \sum_{j \in \hat{\mathcal{N}}^D} C_{ij} - \min_{j \in \hat{\mathcal{N}}^D} (C_{ij}) \qquad i \in \hat{\mathcal{N}}^C \tag{5.5}$$

A pseudocode for the districting algorithm is shown in Algorithm 4. The parameter ρ in line 2 defines the minimum share of capacity that each depot should remain after districting. This is to ensure that the routing solver can create feasible routes to serve all customer demand in a district. The list of customers L^C in line 3 corresponds to all present customers, while the list of depots L^D in line 4 is all open depots. In line 5, an empty list for each open depot is created to keep track of the customers that are assigned to the districts. Line 6 calculates the maximum demand that the district can serve by summing all vehicle capacities and multiplying with $(1-\rho)$. The loop starting in line 7 assigns the customers to the districts in the order of most urgent customer first. The most urgent customer i' and its preferred depot j' that still has remaining capacity to cover the demand are found in lines 8 and 9, respectively. In line 10, i' is assigned to the district $W_{j'}$ and removed from the list of customers L^C. After assignment, the rest capacity of depot j' is decremented by the customer demand in line 11. Whenever a depot gets full, the urgency is recalculated using only the depots that have capacity left, which is assured in lines 12 to 13. Finally, all the districts with assigned customers is returned in line 14.

Algorithm 4 Parallel Assignment (Giosa et al., 2002)

```
1: function DISTRICTING($\rho$, $L^C$, $L^D$)
2:     input: $\rho \leftarrow$ minimum rest capacity
3:     input: $L^C \leftarrow$ list of customers
4:     input: $L^D \leftarrow$ list of depots
5:     $W_j \leftarrow$ list for each depot $j \in L^D$, initially empty
6:     $H^W_j \leftarrow (1-\rho) \cdot |\mathcal{V}_j| \cdot H^V$
7:     while $|L^C| > 0$ do
8:         $i' \leftarrow \operatorname{argmax}_{i \in L^C}(\mu_i)$
9:         $j' \leftarrow \operatorname{argmin}_{j \in L^D}(C_{i'j} \mid H^W_j - D_{i'} >= 0)$
10:        append $i'$ to district $W_{j'}$ and remove it from $L^C$
11:        $H^W_{j'} \leftarrow H^W_{j'} - D_{i'}$
12:        if depot $j'$ has no capacity left then
13:            recalculate $\mu_i$ using depots $\{j \in L^D \mid j$ has capacity$\}$
14:    return $W_j$ for each depot $j \in L^D$
```

5.2.3 Clustering

To effectively reduce the number of customers that the routing algorithm needs to route between, we use a clustering approach that clusters similar customers into super-customers with an aggregated demand and service time. We use the nearest neighbor algorithm proposed by Dondo and Cerdá (2007) to create the super-customers. Algorithm 5 shows the pseudocode for our implementation of the clustering algorithm. The input parameters l^{max} and t^{max} in lines 2 and 3 specify the maximum allowed distance and waiting time that is allowed in order for a customer to be accepted in a cluster. L^C in line 4 is the list of all customers to be clustered, sorted by the earliest start of their time windows. $\mathcal{P}_i^C$ in line 5 contains the time window and demand attributes for each of these customers. The loop that starts in line 7 is the clustering process. For each iteration, the sets containing information for cluster n are first defined in lines 8 to 11. K_n contains the customers, while $\mathcal{P}_n^K$, A_n^K and T_n^K hold the aggregated time window and demand, distance and service time, respectively. A new customer is picked in line 12 and then used to initialize cluster n in line 13. Recall from Chapter 4 that T_i^S is the service time of customer i. The customers that are yet to be clustered are copied to a new list L' in line 14. Iteratively, in priority of the closest customer first, all the customers in L' are tested for the inclusion criteria in lines 15 to 20, being time windows, l^{max}, t^{max} and the vehicle capacity H^V. Customers that satisfy all these criteria are added to cluster n in lines 18 to 20. When all customers in L' are tested, but there are still customers in L^C that are not clustered, the loop starts over with cluster $n+1$.

In line 22, after all customers in L^C are clustered, the coordinates of the clusters are set to the mean of all coordinates in the super-customer. Additionally, the time and distance from the new super-customers to any other customer are updated by using the customer that is closest to this center. This is different from Gulaker et al. (2022), where the time and distance to all customers outside the cluster are updated by using the closest customer in the cluster. As this closest customer strategy represents the lower bound on time and distance, it is expected to underestimate the actual time and distance. On the other hand, we expect that our new strategy overestimates the actual time and distance because the route to and from the cluster originates in the center. One would assume that an optimal route would drive to customers in the cluster that is closer to the preceding customer first, but as we do not expect the clusters to contain many customers, this should not impact the total solution quality too much. However, using this center customer strategy reduces computing time substantially compared to the closest customer approach, as it avoids computing all nearest neighbors each time. This makes the new strategy more attractive. Finally, the set of clusters, or super-customers, and the corresponding aggregated attributes are generated in line 23 and returned in line 24.

5.2.4 Routing

This thesis uses the HGSR solver (Kool et al., 2022) to create routes in the final phase, in contrast to Gulaker et al. (2022) where LKH-3 was used. As discussed in Section 3.3, the reason for changing from LKH-3 is that HGSR seems to perform

Algorithm 5 Heuristic Clustering (Dondo and Cerdá, 2007)

```
function CLUSTERING($l^{max}$, $t^{max}$, $L^C$)
    input: $l^{max} \leftarrow$ max distance between new and closest customer in cluster
    input: $t^{max} \leftarrow$ max waiting time for the earliest possible arrival
    input: $L^C \leftarrow$ list of customers sorted by earliest time window first
    $\mathcal{P}_i^C \leftarrow$ attributes (time window and demand) for each customer $i \in L^C$
    $n \leftarrow 0$
    while $|L^C| > 0$ do
        $K_n \leftarrow$ list of customers in cluster $n$, initially empty
        $\mathcal{P}_n^K \leftarrow$ attributes (time window and demand) for cluster $n$
        $A_n^K \leftarrow$ internal distance for cluster $n$
        $T_n^K \leftarrow$ internal service time for cluster $n$
        remove the first customer $i$ from $L^C$ and append $i$ to $K_n$
        initialize $\mathcal{P}_n^K = \mathcal{P}_i^C$, $A_n^K = 0$ and $T_n^K = T_i^S$
        $L' \leftarrow$ copy of list $L^C$
        for customer $j$ in $L'$ do
            $\hat{\imath} \leftarrow$ the closest customer to $j$ in $K_n$
            if visiting $j$ after $\hat{\imath}$ satisfies time windows, $l^{max}$, $t^{max}$ and $H^V$ then
                append customer $j$ to $K_n$ and remove it from $L^C$
                update cluster attributes $\mathcal{P}_n^K$ according to $\mathcal{P}_j^C$
                update $A_n^K = A_n^K + A_{\hat{\imath}j}$ and $T_n^K = T_n^K + T_{\hat{\imath}j} + T_j^S$
        $n \leftarrow n + 1$
    Update cluster locations
    $W \leftarrow$ set with all $n$ clusters corresponding to $K_n$ with $\mathcal{P}_n^K$, $A_n^K$ and $T_n^K$
    return $W$
```

better than LKH-3 in the literature. HGSR is available at GitHub (PyVRP, 2023) and is capable of solving single-depot VRPs with homogeneous vehicles and time windows. It minimizes the total duration that the vehicles use, similar to LKH-3, and therefore equation (5.6) is the objective function that is used for the routing. By adjusting the vehicle availability time windows, the tour duration constraints (4.22) are also respected. We terminate the solver by using an adaptive time limit. The time limit is based on how much time is used on clustering; more time on clustering means less time on routing, and vice versa. It is a design decision that the entire routing process should be quick to enable fast evaluation of depot configurations. After termination, we calculate the actual routing costs according to the objective function defined in Section 4.3.2. The aim of this approach is not to find close to optimal costs, but rather that the cost of solutions rank similarly to a long evaluation of the routing problem.

$$z = \min \sum_{i \in \mathcal{N}^D} \sum_{v \in \mathcal{V}_i} (t_v^E - t_{iv}) \tag{5.6}$$

In essence, the constraints considered by HGSR are the same as in the strategic column of Table 5.3, except for the overtime constraints which are handled by the post-processing step. The objective function, equation (5.6), used by HGSR is another significant simplification of the real routing costs described in Section 4.3.2, which includes fixed costs for using vehicles, the distance costs and the overtime costs. HGSR does not account for heterogeneous costs, either. To ensure that the overtime constraints are satisfied, and allow for heterogeneous costs, we add a post-processing step after HGSR has terminated. This post-processing step follows a regret measure, where we compare the differences in costs of assigning a route to the two cheapest alternative vehicles. Vehicles are given routes in order of the largest cost difference between the two alternatives, as the larger the difference, the more important it is that the cheapest vehicle gets the route.

Chapter 6

Scenario Generation

As described at the beginning of Chapter 5, the distribution of the stochastic customer presence is estimated by using scenarios. Because the scenarios put into the model is not a full description of the true distribution, there is a need to assess whether the set of scenarios, or scenario tree, represents the underlying distribution well. In this chapter, we first explain how the customer distributions for each shift are created in Section 6.1, and then describe the scenario generation method in Section 6.2. The scenario generation method is tested in Section 7.2 of the computational study.

6.1 Distribution of the Stochastic Parameters

The stochastic part of the problem is customer presence. In Section 4.2, this is defined as a collection of stochastic parameters $\xi = (F_0, ..., F_{|\mathcal{N}^C|})$ where F_i is equal to 1 if customer $i \in \mathcal{N}^C$ places an order, and 0 otherwise. We assume that F_i is independent and identically distributed with a Bernoulli distribution, and that the probability of being present is equal for each customer. Because of these properties, and since the Bernoulli distribution is a special case of the binomial distribution with a single trial (Walpole et al., 2012), we argue that ξ can be constructed by first drawing the number of customers from a binomial distribution with $|\mathcal{N}^C|$ trials and then draw the customers with corresponding attributes. Furthermore, according to the central limit theorem, the binomial distribution is approximately equal to the normal distribution when the number of trials is large (Walpole et al., 2012), which is the case in this problem.

When modeling the customer presence, we want to take into account the variation in customer presence for different shifts during the week. We define μ to be the average shift demand. For each shift in a week, a normal distribution is constructed to draw the number of customers. The mean demand for each of these distributions is expressed as percentages of μ, which are retrieved from historical data. For instance, Monday afternoon could be 140% of μ while Sunday morning could be 70%. Over one week, these percentages must average to 100%. All distributions use the same standard deviation σ. To avoid the endless tails of the normal distributions, the extreme 1% values are removed at each end of the distributions. For instance, the lower 1% contains a negative number of customers, and the upper 1% contains billions of customers, none of which are relevant cases when considering depot locations.

After the number of customers for a scenario s has been determined based on these distributions, the present customers i are sampled from the set of possible customers, $\mathcal{N}^C$. For each scenario $s \in \mathcal{S}$, this decides the value of $\tilde{F}_{is}$ in realization $\tilde{\xi}_s$. Each customer is connected to a set of attributes, being location, demand, service time and time window. It is important to notice that, for modeling reasons, the attributes for each customer are deterministic, meaning that if a specific customer is sampled in two different scenarios they always have the same set of attributes. However, by sampling from a big enough pool of customers, variation in these attributes can also be represented. For instance, different sampled customers with different demands in the same location may represent one customer with varying demands or customers living in the same building.

6.2 Scenario Generation Method

Having one distribution for each shift is impractical in terms of sampling. First, being able to represent several distributions in a good way requires more scenarios than just drawing from one distribution. Second, shift distributions that are overlapping can lead to redundant (similar) scenarios in the scenario tree, which could rather be modeled as one scenario with a higher probability in order to save computational time. Figure 6.1 shows the shift distributions used in our experiments, with μ equal to 2000 and σ equal to 1.5%. Note that since the demand for the morning and afternoon shifts are highly correlated in Oda's case, we assume the same distribution for the morning and afternoon shifts of a weekday. This allows for using only one distribution per weekday. Furthermore, it can be observed from Figure 6.1 that shifts on certain days are highly overlapping. We therefore propose that the shift distributions can be merged into a single distribution.

This merged distribution is constructed by doing a very large sampling from each distribution and adding the samples together to a big sample of customer numbers. In our experiments, we draw 200,000 customer numbers from each of the 7 distributions, which results in one large set of 1,400,000 numbers. This sample is then split up into N ranges, and each range is given a weight based on the share of numbers that are in that range. A scenario tree can then be generated by drawing x customer numbers from each of the N ranges, resulting in $Nx = |\mathcal{S}|$ different scenarios. Each of these scenarios $s \in \mathcal{S}$ is given the weight p_s of the range they are drawn from, with all weights scaled so that they sum to 1. These weights are later used to calculate the expected costs of the routing problems based on the calculated routing cost of each scenario, according to equation (5.1).

We use the value $N = 8$ as this seems like a small enough increment when testing different numbers of scenarios in Section 7.2, while being large enough for dividing the distribution in ranges of a reasonable size. The size of the ranges is set manually,

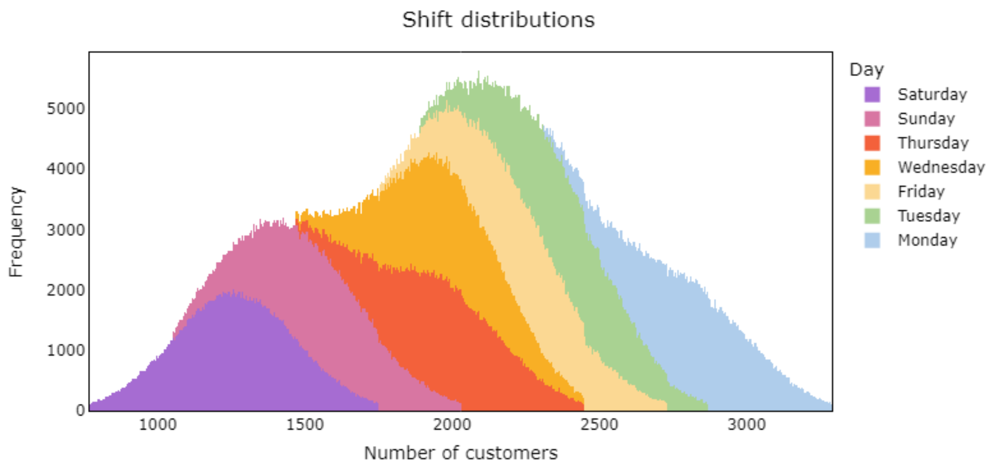

Figure 6.1: Stacked shift distributions for each day with $\mu = 2000$, $\sigma = 1.5\%$, and 200,000 samples.

with a focus on that the ranges on the outer edges of the distribution are not very wide. This ensures that the extreme scenarios are represented when drawing from each range. The merged distribution with the used ranges and their weights is shown in Figure 6.2.

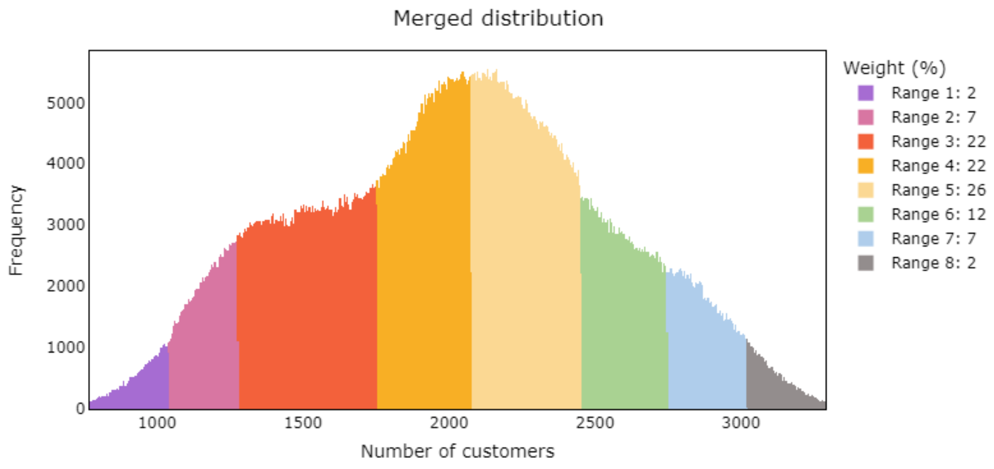

Figure 6.2: Merged distribution of the shift distributions with $\mu = 2000$, $\sigma = 1.5\%$, and 200,000 samples. The ranges used and their percentage weights are shown to the right in the legend.

Chapter 7

Computational Study

In order to evaluate the proposed solution methods, we analyze the results of our computational study in this chapter. Several experiments are designed to investigate how the solution methods behave when tested on different problem instances and to provide insights into business-related questions.

This chapter begins by describing the experimental setup in Section 7.1. The stability of the scenario generation method is then tested in Section 7.2. In Section 7.3, the routing algorithm is compared with Gulaker et al. (2022) to evaluate the impact of using HGSR instead of LKH-3 and the new way of updating time and distance to the customer clusters. Afterward, we move on to the evaluation of the location search algorithms in Section 7.4, where we focus on describing how BTS, CATS and GRASP perform. To get more knowledge about the solutions, we analyze the costs in more detail and how they are affected by solution attributes in Section 7.5. Finally, we provide some insights specific to Oda's case in Section 7.6.

7.1 Experimental Setup

In this section, an overview of the data that is provided by Oda and the problem instances that are used in the experiments are first given. Next, we present the parameters and corresponding values, followed by an explanation of the experiments and measures used in these. Experiments are run on computing servers with two Intel Xeon Gold 5115 CPUs with a total of 20 cores running at a nominal speed of 2.4 GHz, and 96 GB of RAM. The servers are running the Linux distribution CentOS 7. Evaluating scenarios is done in parallel when evaluating a depot configuration in CATS and BTS. When evaluating only one scenario per depot configuration in GRASP, the evaluation of depot configurations is run in parallel instead.

7.1.1 Overview of the Data

The customer data is sampled from different sources. Customer locations are sampled from 8618 unique locations in the Berlin area in Germany, which have been generated by Oda's customer simulator that takes into account the demography in the area and the ordering probability for different customer segments. As other attributes are sampled independently, we allow several customers to have the same location, as this can be seen as customers living in the same building. Hence, the number of customers is not limited by the number of unique locations. The unique customer locations are distributed according to Figure 7.1, where the color scale indicates how many are located within a given hexagonal area. The colors indicate higher density towards the center of Berlin. There are also some smaller concentrations of customers on the outskirts of the city, like in Potsdam. According to the data, the driving distance between a customer in the middle of Potsdam and the center of Berlin is 39 km, and it takes approximately 39 minutes to drive this distance during the morning shift. This shows that the customers are distributed over large areas, both in terms of spatial distance and driving time.

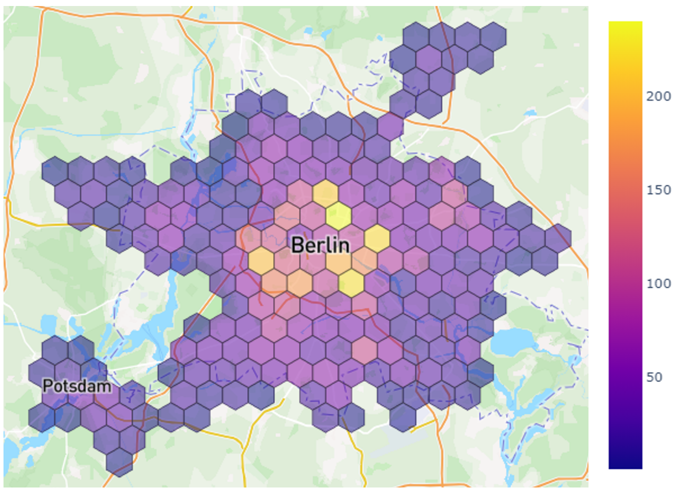

Figure 7.1: Heatmap of possible customer locations.

The customers' demand is sampled from a uniform distribution between 10 and 50 kilograms. A vehicle, which has a maximum capacity of 1000 kilograms, can therefore serve 33 customers with an average demand. A customer's service time is set equal to $180 \cdot (1 + D_i/\overline{D}_i) + r$, where D_i is the demand of customer i, $\overline{D}_i$ is the average demand and r is random noise sampled from a uniform distribution between -30 and 30. This results in service times in the range between 3.5 and 8.5 minutes, with 6.0 minutes on average. The time windows of the customers are sampled from an ordinary weekday in Oda, and either have a short duration of 2 hours or a long duration of 5 hours. For each problem instance we either sample all time windows from the morning shift or the afternoon shift, which shift is chosen randomly with 0.5 probability for each. In Figure 7.2, the number of customers with open time windows at different times of the day is shown to get an overview of which periods are most busy during the day. The peak of open customer time windows occurs during the middle of the shifts. The morning shift is not as busy in the beginning compared to the afternoon shift. Furthermore, there are few time windows open in the last hours of the afternoon shift.

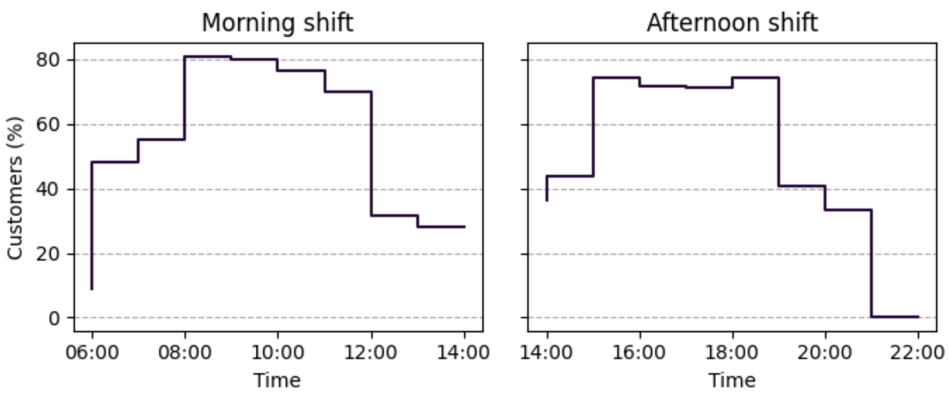

Figure 7.2: Percentage of all customers whose time windows are open for delivery throughout each shift. Shifts are separate, with no time windows overlapping between the shifts.

There are 20 unique depot locations provided in the data. These are shown in Figure 7.3, in addition to the warehouse location. The depot locations are evenly spread across Berlin, and there are also some depots located in Potsdam. When compared to the customer distribution in Figure 7.1, some depots are located close to the city center where most customers are, whereas others are located on the outskirts of Berlin. The warehouse, which also functions as a large depot, is located far south of the customers and hence has a longer driving distance to customers on average compared to the depots.

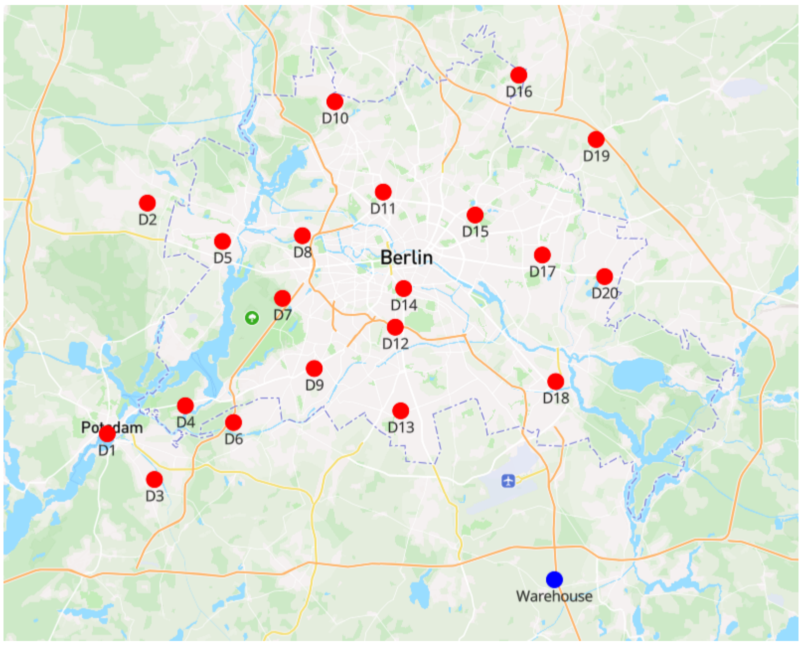

Figure 7.3: Potential depot locations.

Recall from Chapter 4 that the cost for a solution is the sum of the opening cost G_i for each open depot i, the fleet cost G^F for each vehicle in the fleet, and the expected routing costs. The opening cost G_i consists of line-haul cost and leasing cost. The line-haul cost is calculated with an estimated time cost of 800 per hour, using the time of a round-trip between the warehouse and the depot. Each depot requires enough line hauls to fill the capacity of all available last-mile vehicles at the depot, and in our case, a line haul can fill the capacity of 6 last-mile vehicles. The leasing cost is the estimated cost to open, rent, maintain and operate a depot, and is derived from a cost between 50,000 and 100,000 per month dependent on the centrality of the location and number of vehicles that are available at the depot. The used line-haul cost and leasing cost for each depot can be found in Table C.2 in Appendix C. Additionally, the fleet cost G^F is set to 333 per vehicle for all depots and is found by dividing the monthly vehicle leasing and service costs, estimated to 20,000 per vehicle, with the number of shifts each month, which is 60.

Regarding the routing costs, we consider the fixed cost C_v^F, distance cost C_v^D, time cost C_v^T and overtime cost C_v^O, shown in Table 7.1. The vehicle fleet consists of two types of vehicles. The vehicle type with a high fixed cost and no time cost is referred to as fixed-cost vehicles, whereas the other type is referred to as time-cost vehicles. A fixed-cost vehicle must be used for a minimum of 6.9 hours before it is cheaper than a time-cost vehicle. Both types have the same capacity of 1000 kg. The mix of vehicle types that are available at each depot can be found in Table C.2 in Appendix C.

Table 7.1: The capacity H^V, fixed cost C_v^F, time cost C_v^T (per hour), distance cost C_v^D (per km) and overtime cost C_v^O (per hour) for the vehicle types in the fleet.

Vehicle type	H^V	C_v^F	C_v^T	C_v^D	C_v^O
Fixed-cost vehicle	1000	3240	0	3.5	180
Time-cost vehicle	1000	760	360	3.5	180

7.1.2 Description of the Problem Instances

For our experiments, 7 problem instances are generated. The number of depot locations, alternative depot sizes at each location and the scenario tree that is used in each problem instance are given in Table 7.2. Each scenario tree is given a name to indicate when the same realizations are reused, and we also show the average shift demand μ and standard deviation σ used in the scenario generation method described in Chapter 6, in addition to the minimum (min) and maximum (max) number of customers in the scenarios. We generate three problem instances that use the same scenario tree, S1. These are generated by using a μ of 2000 and σ of 1.5%. Problem instances A, B and C contain different numbers of depot locations and alternative depot sizes at each location, with specific depots that are included in each of these problem instance as shown in Table C.2 in Appendix C. This enables us to observe how well the solution methods scale to larger solution spaces and also to compare the difference when having alternative depot sizes.

In Section 7.6, we perform a case study with problem instances that are more closely related to the problem that Oda faces in Berlin. For this case study, we define problem instances O2, O3, O4 and O5 with depots and scenario trees as shown in Table 7.2. The depot locations that are used in the O problem instances are depot D17, D10, D12 and D6. As shown in Figure 7.3, depot D17 is located in the eastern part of Berlin, D10 in the north, D12 in the center, and finally D6 on the border between Berlin and Potsdam. The 4 alternative depot sizes at each location have 6, 12, 18 and 24 vehicles available. For example, D17-6 indicates depot D17 with 6 available vehicles. Various scenario trees are used for the O problem instances because we want to analyze how the solution is affected when the average shift demand μ is changed. More specific details regarding the depots in the O problem instances can be found in Table C.1 in Appendix C.

Table 7.2: Problem instances with number of depot locations, number of alternative depot sizes at each location and the scenario tree that is used, including the average shift demand μ, standard deviation σ (%), minimum (min) number of customers and maximum (max).

Instance	Depots		Scenario tree				
	Locations	Sizes	Name	μ	σ	min	max
A	20	1	S1	2000	1.5	950	3090
B	10	2	S1	2000	1.5	950	3090
C	10	1	S1	2000	1.5	950	3090
O2	4	4	S2	2000	1.5	967	3111
O3	4	4	S3	3000	1.5	1334	4772
O4	4	4	S4	4000	1.5	2006	6133
O5	4	4	S5	5000	1.5	2222	7561

As discussed in Section 5.2.1, a penalty is incurred per excessive vehicle that must be added to the warehouse fleet to ensure that all customers are served. What magnitude of the penalty cost represents reality the best would vary much from case to case. Based on discussions with Oda, a penalty cost of $C^P = 500$ when exceeding $\overline{\mathcal{V}_0} = 120$ vehicles at the warehouse is found to be realistic in their case. These are the values we use in the O problem instances run in the case study for Oda in Section 7.6. In order to test the performance of our location search algorithms, we want to ensure that opening several depots is profitable. This way we can observe how the algorithms search through different numbers of depots and the larger neighborhoods of bigger depot configurations. We know that Oda has a relatively large warehouse and flexible vehicle agreements, and preliminary testing shows that a higher C^P and lower $\overline{\mathcal{V}_0}$ than realistic for Oda is needed to achieve this. For problem instances A, B and C we therefore consider a case with less warehouse capacity and a higher cost of acquiring extra vehicles. Here we use a penalty cost of $C^P = 1000$ and an initial capacity of $\overline{\mathcal{V}_0} = 30$ vehicles at the warehouse.

7.1.3 Presentation of the Algorithmic Parameters

The parameters that are used in the computational study are presented in Table 7.3. The number of consecutive non-improving explorations φ^C in the constructive neighborhood generator is set to 5. This value is based on a trade-off between time used for diversification and intensification, and since neighbors are already sorted by their expected mean depot distance it is assumed that the value of 5 is sufficient to reach the correct number of depots. Parameters for the adaptive neighborhood generator are based on the parameters that are used by Pisinger and Ropke (2007). As stated in that paper, these parameters should be quite forgiving as the weight given to each operator is adjusted automatically throughout the search. The bias towards one particular neighborhood in the combined neighborhood generator is a difference from Pisinger and Ropke (2007). We assume, however, that swap is more important in the combined neighborhood generator as the adaptive phase starts with an already promising number of open depots. The parameters used for districting and clustering are set to be the same as in Gulaker et al. (2022), where these are tuned for the same VRP as solved in this thesis. For HGSR, we use the default parameters in the solver.

Table 7.3: Parameters used in the computational study.

Parameter	Description	Value
Constructive neighborhood generator		
φ^C	Limit on non-improving explorations	5
Adaptive neighborhood generator		
θ	Decay rate of neighborhood operator scores	0.9
ω_1	Reward for finding new best solution	10
ω_2	Reward for finding an improving solution	5
s_0	Start score of the operators	10
s^B	Swap bias in the combined neighborhood generator	20
Districting		
ρ	Minimum rest capacity at depots	0.01
Clustering		
l^{max}	Maximum distance (meters) between customers	2000
t^{max}	Maximum waiting time (seconds) at visit	0

The starting solution for all solution methods is chosen to be zero open depots. This represents a real-world situation, where no depots are initially open. The task is therefore to find how many and which depots should be open. Although starting at zero open depots is not the only possibility, both GRASP and CATS are designed to start at the lowest feasible number of open depots. With the extended warehouse capacity, the solution with no open depots is always feasible. Moreover, the idea is to have many more depot options than what should eventually be opened. For instance, it could be reasonable to consider 20 options, even though at most 5 of these should actually open. Having a starting solution with all possible depots open would therefore be more inefficient as finding a realistic number of open depots would take longer. Starting somewhere in the middle is also possible, but that requires domain knowledge to determine a good starting solution. Although it is possible to use domain knowledge to find a good starting solution, we have instead opted for algorithm designs that can quickly move to a good number of open depots.

7.1.4 Stability Testing

A common way to assess the scenario generation method is to test stability. Wallace and King (2012) introduce in-sample stability and out-of-sample stability as evaluation criteria to measure stability. In-sample stability is achieved when the objective values corresponding to the optimal solutions for different scenario trees are approximately the same. Hence, with in-sample stability, the objective value is not affected by the scenario generation method in itself. When testing for out-of-sample stability, the first-stage solutions are fixed and then evaluated using the true distribution ξ. This reveals whether the generated scenario trees really are good representations of the true distribution.

One of the problems with the stability tests defined by Wallace and King (2012) is that they assume the optimal solution can be found. Using heuristics we can not guarantee that the solution we find is optimal. Furthermore, there are random components in the routing solver that can create noise. Because it is impossible to determine exactly whether the noise comes from the scenario generation method or the heuristics, in-sample stability cannot be tested. However, we can still test out-of-sample stability and identify the required number of scenarios to assure the noise from the scenario generation method is lower than the noise from the heuristics. At this point, we know that it does not help to increase the number of scenarios further, which is valuable insight as we want to keep the number of scenarios low in order to reduce the computing time from solving a VRP in each scenario.

We use a weak out-of-sample test for heuristics as proposed by Guo et al. (2019) to find the required number of scenarios. First, we generate 5 scenario trees, each containing the same number of scenarios with a μ of 2000 and σ of 1.5%. We pick 5 first-stage solutions that seem promising from preliminary testing to be evaluated, as described in Table D.1 of Appendix D. Next, with each of these first-stage solutions fixed, we find the objective values for all the scenario trees and compute the relative difference between the largest and smallest objective value. This results in 5 relative difference values, one for each first-stage solution. The stability for the chosen number of scenarios is defined to be the largest relative difference between objective values, and the aim is to minimize this measure. As stated by Wallace and King (2012), unless the scenario generation method is deterministic, increasing the number of scenarios results in better stability. If the opposite happens, it can be assumed that the noise from the heuristic dominates the noise from the scenario generation. By repeatedly running the weak out-of-sample test for 8, 16, 24 and 32 scenarios we can identify this best stability.

Furthermore, we want to assess how the ranking of depot configurations is affected by stability and whether an increase in the number of scenarios leads to a more consistent ranking. If a new scenario tree is used and this leads to exactly the same percentage increase or decrease in costs for all depot configurations, then the ranking must also be the same, and we can assure that the ranking is independent of which scenario tree is used. To do this, we generate 5 scenario trees with the same number of scenarios and then create all possible pairs of these scenario trees, resulting in 10 pairs. For each scenario tree in each pair, we evaluate 10 depot configurations, as described in Table D.2 in Appendix D, and rank them by ascending costs. Kendall's τ correlation coefficient, proposed by Kendall (1938), is used to evaluate the correlation between the rankings in each pair. This coefficient ranges between -1 and 1, where -1 indicates total disagreement by having the opposite ranking and 1 indicates the exact same ranking of depot configurations. This test is repeated for 8, 16, 24 and 32 scenarios, and we compare the minimum, maximum and average τ to say how the ranking is affected.

7.1.5 Evaluation of the Routing Algorithm

To evaluate the routing algorithm, and particularly the changes from Gulaker et al. (2022), we compare the solution costs with benchmark costs from Oda's routing solver, Navegante. The first change is how distance and time in the clustering algorithm are updated. The second change is implementing HGSR as the routing solver instead of LKH-3. The differences between the solution costs and the benchmark costs can be viewed as an error in our estimates. Therefore, for this type of comparison, we use the mean absolute percentage error (MAPE) to evaluate accuracy. To assess the ability to rank depot configurations correctly, we compute Kendall's τ correlation coefficient, between the ranking implied by the routing algorithm and the ranking according to the solutions from Navegante. We also investigate how the clustering algorithm is affected by comparing the computational time. 23 routing problems of varying sizes are used as benchmarks. These are named T1 to T23 and are described in Table C.3 of Appendix C.

In Section 7.6 we consider a case study for Oda, and we therefore wish to make the estimated routing costs more similar to Oda's actual routing costs. We do this by defining an adjustment constant that is multiplied by the estimated routing cost. To find the best adjustment constant, we compare constants found by three different methods and use the one that yields the lowest average gap using cross-validation techniques with 5 folds. Cross-validation is performed to examine the methods' bias toward the data available. The three methods are a mean squared error (MSE) optimization, finding the mean gap and finding the median gap. For each fold, the constants are found by using 4/5 of the data, which afterward is evaluated on the remaining 1/5 of the data. Hence, the former is the training data, and the latter is the test data. With the MSE method, the adjustment constant C is found by minimizing $\sum_{t=1}^{T_i}(z_t^{ODA} - C \times z_t)^2$, where z_t is the cost found by our algorithm and z_t^{ODA} is the cost found by Navegante for all routing problems t in training set T_i for fold $i \in \{1,2,3,4,5\}$. With the median and mean methods, the adjustment constant is 1 minus the percentage median and mean gap respectively. MAPE is used to compare the error of the estimated costs before and after adjustment.

7.1.6 Evaluation of the Location Search Algorithms

When evaluating the location search algorithms, we record the best found objective value $\tilde{z}(t)$ at time t, and define z^{BKS} as the best known solution value for the problem instance. From these two values, we define the gap to best known solution value $g^{BKS}(t)$ at time t as in equation (7.1). That is, the relative gap between $\tilde{z}(t)$ and

z^{BKS}. The best known solution and associated objective value for problem instances A, B and C are found during testing.

$$g^{BKS}(t) = \frac{\tilde{z}(t) - z^{BKS}}{z^{BKS}} \tag{7.1}$$

The primal integral, defined in Berthold (2013) as the integral of the primal gap, is also used as an evaluation metric. In equation (7.2), the primal gap $g^{PRIM}(t)$ is defined, and will be a number between 0 and 1 describing the size of the cost gap between $\tilde{z}(t)$ and z^{BKS} when a solution is found at time t, and 1 if a solution is not found. When having discrete measurements, as in our case, the primal integral $g^{PI}(t)$ is equal to the sum of the primal gaps multiplied by the share of the time they stay in the primal gaps. Thus, given the time points t_i as fractions between 0 and 1 where a new best solution is found for $i \in 1, ..., I-1$, the primal integral is defined as in equation (7.3).

$$g^{PRIM}(t) = \begin{cases} \frac{\tilde{z}(t)-z^{BKS}}{\tilde{z}(t)}, & \text{if a solution is found} \\ 1, & \text{otherwise} \end{cases} \tag{7.2}$$

$$g^{PI}(t) = \int_{t=0}^{1} g^{PRIM}(t)\, dt = \sum_{i=1}^{I} g^{PRIM}(t_{i-1}) \cdot (t_i - t_{i-1}) \tag{7.3}$$

Berthold (2013) argues that certain metrics do not reflect a correct assessment of the trade-off between speed and solution quality. For instance, the time needed to find the best known solution underestimates algorithms that can find solutions that are close to the best known solution in a short amount of time. This is more evident in the primal integral, where finding good solutions early is rewarded. Therefore, we use both the primal integral $g^{PI}(t)$ and the gap to the best known solution $g^{BKS}(t)$ to evaluate the performance of the solution methods.

7.2 Stability Testing

We use the weak out-of-sample test as defined in Section 7.1.4 to find the minimal number of scenarios that achieves acceptable stability. We use the average objective value (AOV) and the relative difference (RD) between the largest and smallest objective value to evaluate the results. The largest RD for each number of scenarios is called the stability level. The results are shown in Table 7.4.

The stability is affected by the noise from the heuristic, but an increase in the number of scenarios helps to some degree. With an exact solution method, RD is expected to decrease as the number of scenarios increases. When using 8 scenarios, the stability becomes 8.43%. Therefore, we could expect that the objective value changes by approximately 8.43% whenever a new scenario tree is generated from the scenario generation method. Increasing the number of scenarios to 16 improves the

Table 7.4: Average objective value (AOV) and percentage relative difference (RD %) for the weak out-of-sample stability test with different numbers of scenarios.

Number of Scenarios		8	16	24	32
Solution 1	AOV	242,562	245,674	245,768	246,603
	RD %	3.21	2.91	3.43	2.73
Solution 2	AOV	243,665	243,996	242,158	244,287
	RD %	3.36	3.46	2.72	3.27
Solution 3	AOV	239,541	241,724	242,844	242,772
	RD %	6.77	2.91	3.17	3.11
Solution 4	AOV	245,622	248,544	248,934	248,364
	RD %	8.43	3.31	4.49	3.35
Solution 5	AOV	238,911	238,962	240,155	242,159
	RD %	5.86	0.99	3.51	1.62
Stability level %		8.43	3.46	4.49	3.35

stability to 3.46%, which is a significant improvement. However, when increasing the number of scenarios further to 24 and 32, the stability level is more fluctuating. This is contradictory to what we would expect from an exact solution method, because increasing the number of scenarios should lead to lower RD and better stability. What this out-of-sample test shows is that after this point, the noise from the heuristic is larger than the noise from the scenario generation method. Therefore, increasing the number of scenarios further from 16 is not necessary, as the best stability we can achieve with this solution method has been reached.

Kendall's τ correlation coefficient is used to evaluate how the ranking of 10 depot configurations is affected by the noise from the scenario generation method and the heuristic. The results in Table 7.5 show that in the best case, the τ is equal to 0.96, which means that only two consecutive depot configurations switch their ranking when a new scenario tree is used. Since a τ of 1.00 is not found for any number of scenarios and scenario trees, it seems like this misplacement is a consequence of errors in the estimated costs rather than the scenario generation method. Scenario trees with 8 scenarios have a lower minimum τ compared to a higher number of scenarios. This also holds on average, even though the values are closer to each other. There is no indication that increasing the number of scenarios above 16 results in a more consistent ranking. It might even look like the noise from the heuristics exceeds the noise from the scenario generation method at a point because using 32 scenarios has a lower minimum and average τ compared to 16 and 24. By using 16 scenarios we can expect a τ between 0.82 and 0.96. Results for one pair of scenario trees with 16 scenarios and τ equal to 0.82 is shown in Table 7.6. Depot configurations 1, 2 and 4 are ranked differently between scenario tree 1 and scenario tree 2, while the remaining depot configurations obtain the same rankings. However, the absolute differences in cost from depot configuration 1 for the misplaced depot configurations in scenario tree 2 is under 0.5%, which is much lower than the rest of the cost differences. In fact, the relative difference between depot configurations 1 to 4, where the differences in rankings occur, is on average 0.32% over the scenario trees used for the ranking test. The relative differences between depot configurations 5 to 10, which are always consistently ranked, is on average 5.12%. This indicates that the noise from the scenario generation method and the heuristic with 16 scenarios can lead to inconsistent ranking when the differences in costs are very small. Furthermore, with a relative difference this small, it also means that the expected impact of this inconsistency is insignificant.

Table 7.5: Kendall's τ correlation coefficient between the rankings of 10 depot configurations according to 5 different scenario trees.

	8	16	24	32
Maximum	0.96	0.96	0.96	0.96
Minimum	0.69	0.82	0.82	0.73
Average	0.85	0.89	0.88	0.86

Table 7.6: Ranking of depot configurations δ^i by two different scenario trees with 16 scenarios and Kendall's τ correlation coefficient of 0.82. The difference (diff) in cost from δ^1 is shown.

	Scenario tree 1			Scenario tree 2		
i	Rank	Diff	Diff (%)	Rank	Diff	Diff (%)
1	1	0	0.00	4	0	0.00
2	2	146	0.06	1	-615	-0.26
3	3	434	0.18	3	-76	-0.03
4	4	995	0.42	2	-390	-0.16
5	5	5,542	2.32	5	3,957	1.67
6	6	6,487	2.71	6	4,886	2.07
7	7	9,214	3.85	7	7,201	3.05
8	8	10,840	4.53	8	8,975	3.80
9	9	11,815	4.94	9	9,748	4.12
10	10	19,798	8.28	10	17,757	7.51

7.3 Evaluation of the Routing Algorithm

Without the routing costs being calculated well and quickly enough, the whole solution method collapses. The quick calculation is ensured by putting the same time limit of 90 seconds as Gulaker et al. (2022) on the routing algorithm, but we still need to evaluate the solution quality. The goal is that the costs found by our routing algorithm are close to the cost of a benchmark solution. In particular, the most important aspect is that our routing algorithm ranks the solutions the same way as the benchmark. This section therefore goes more into detail on the performance of the routing algorithm.

Compared to Gulaker et al. (2022), who experienced clustering times of up to 80 seconds on 1808 customers, we have improved the algorithm so that up to 2991 customers are clustered in under 41 seconds. Figure 7.4 shows the computational time spent on districting, clustering and routing. The bars are ordered by the largest maximum district size. T3 has the largest share spent on clustering, but it is also the biggest problem with 2991 customers in one district. After clustering T3, only 586 super-customers remain, which makes the routing itself more manageable.

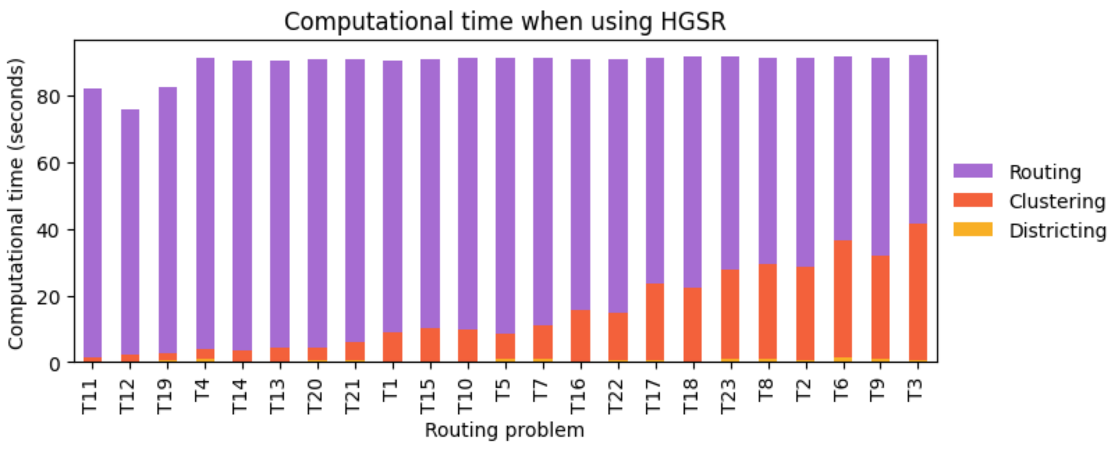

Figure 7.4: Computational time spent on districting, clustering and routing with Hybrid Genetic Search Router (HGSR) for routing problems T ordered by increasing maximum district size.

To assess the effect of using HGSR instead of LKH-3, we compare their MAPE and Kendall's τ correlation coefficient on routing problems T. In Figure 7.5, we show the routing costs found by our routing algorithm using both LKH-3 and HGSR compared to the costs found by Navegante. The results are ordered by increasing routing costs for Oda. First, we observe that HGSR's costs generally lie closer to Navegante's costs than LKH-3's costs do, although some of the costs found by our algorithms are so close that only one data point is visible, such as for T14 and T21. The MAPE is 11.26% for LKH-3 and 9.07% for HGSR. LKH-3 obtains a Kendall's τ correlation coefficient of 0.921, and HGSR obtains a τ of 0.937 when using Navegante's solutions as the correct ranking.

Another interesting observation is that when the cost of LKH-3 and HGSR are most similar, the number of customers per open depot is the lowest. Because LKH-3 and HGSR only work on single-depot VRPs after clustering and districting, it is the number of super-customers after this process that should be investigated. When there are few customers to route between, LKH-3 and HGSR perform relatively similarly. When the number of customers is large, however, HGSR performs better. This is illustrated in Figure 7.6 by showing the routing cost difference between LKH-3 and HGSR as a function of average problem size in the single-depot VRPs. The difference is positive when LKH-3 has a higher gap than HGSR. To the left in the plot, the average number of customers to route between is relatively small, as low as 41, due to a large number of districts. With a low number of customers, there are only small differences between the solvers. As the number of average customers in districts increases because the number of districts decreases, the cost difference between HGSR and LKH-3 increases. This means that HGSR finds at least as good or better solutions than LKH-3. When costs are similar, it is because the districting

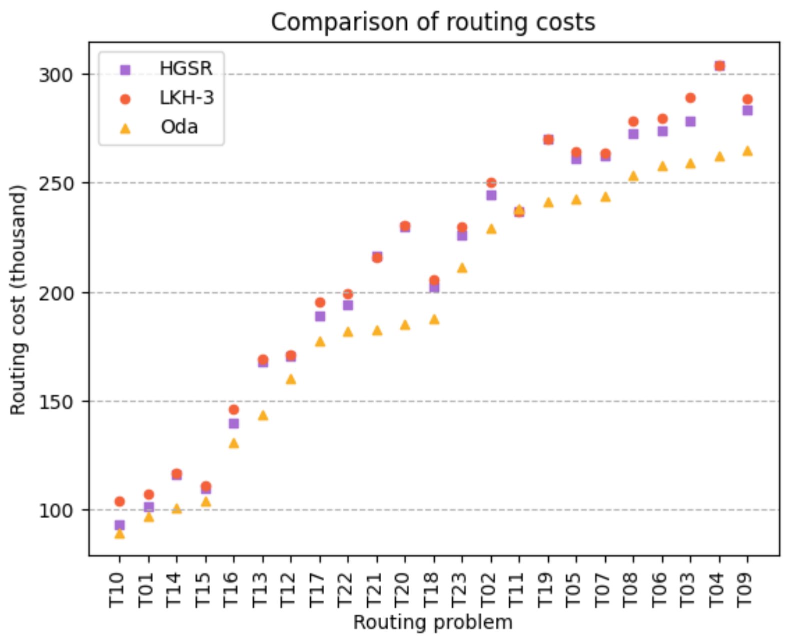

Figure 7.5: Comparison of the routing costs found by Oda's operational routing solver, Navegante, and our routing algorithm using Hybrid Genetic Search Router (HGSR) and Lin-Kernighan-Helsgaun 3 (LKH-3) in the routing phase.

and clustering algorithms have created small routing problems that are relatively easy for both solvers to create good routes in.

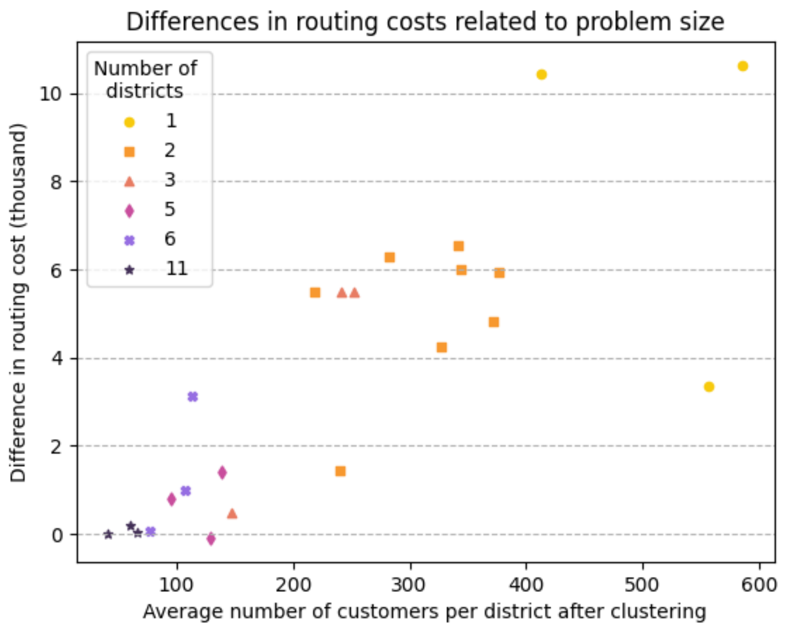

Figure 7.6: Cost difference between solutions found by Hybrid Genetic Search Router (HGSR) and Lin-Kernighan-Helsgaun 3 (LKH-3) plotted against average number of customers after districting and clustering. A positive difference is caused by LKH-3 having a higher gap than HGSR. The number of districts is indicated by the marker.

Furthermore, we observe that our routing algorithms find routing costs above Navegante's routing costs except for T11. Consistently overestimating the cost is advantageous, if it means the ranking of solutions remain the same. However, the routing costs found by our algorithms are not monotonically increasing with Navegante's routing costs. For example, Navegante finds the routing cost of T15 to be higher than that of T14, but our algorithms find the opposite. Nevertheless, as explored in Gulaker et al. (2022), individual differences between two scenarios do not have as big of an impact when averaging over many scenarios. When ranking depot configurations by the average over scenarios, both routing solvers rank depot configurations consistently with the ranking from using Navegante.

7.4 Evaluation of the Location Search Algorithms

In this section, we evaluate the solution methods using different measures in order to identify the most promising method. Additionally, we discuss the strengths and weaknesses of the different methods. The three location search algorithms are assessed. First, the performance of these are evaluated. Then, the differences in performance are discussed by analyzing diversification and intensification. Finally, CATS is analyzed further by assessing the performance impact of its components.

As explained in Section 7.1.6, we measure the performance relative to the best known solution (BKS) for each problem. Since our problem instances have not been researched earlier, the BKS is set to the best found solution in the experiments performed in this thesis. These solutions are presented in Table 7.7.

Table 7.7: The best known solution (BKS) and associated objective value (z^{BKS}) during the experiments for problem instance A, B and C.

Instance	Open depots in BKS	z^{BKS}
A	D9, D10, D18	239,039
B	D5, D13, D17	240,110
C	D5, D13, D17	240,104

7.4.1 Evaluation of Performance

Figure 7.7 shows the gap from the best found solution to the best known solution during the runs for the three location search algorithms, for problem instances A, B and C defined in Section 7.1.2. The curves in the figures indicate the convergence profile for BTS, CATS and GRASP. When looking at problem instance A, which is the largest problem instance in terms of possible depot configurations, the results indicate that BTS and GRASP perform better than CATS. CATS eventually manages to explore a solution close to the best known solution, but is stuck at a gap of approximately 0.5% for a long time. On the other hand, CATS seems to perform better on problem instances B and C. From the convergence profile, we can see that

CATS manages to find a good solution within 3 hours on problem instance B, which is only beaten by GRASP after approximately 11 hours of runtime. On problem instance C, CATS is the best-performing algorithm within 2 hours, but GRASP finds the best known solution approximately 1 hour before CATS finds a gap of 0.013%. Thus, it seems like CATS is very efficient in the beginning of the search, especially on problem instances B and C, but that GRASP eventually finds the lowest gap if given enough time.

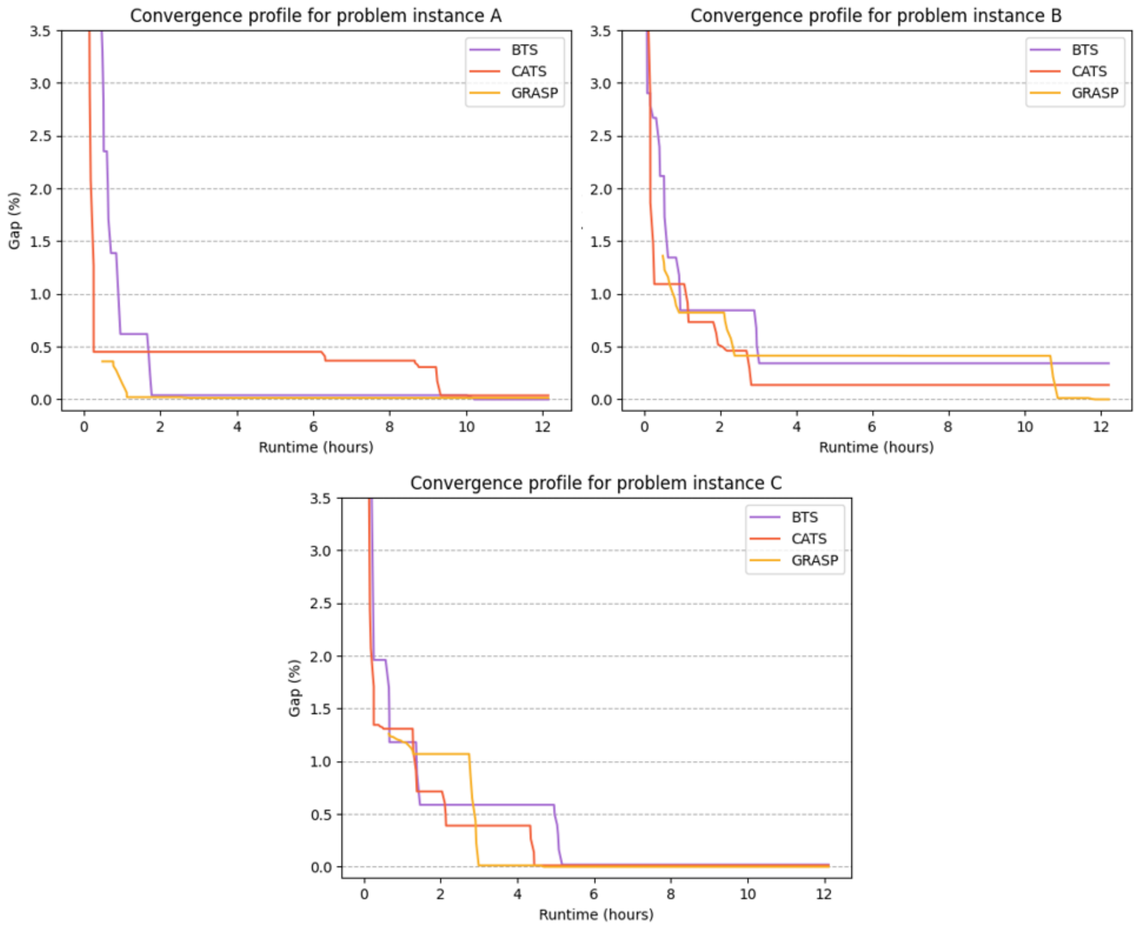

Figure 7.7: Convergence profiles for Basic Tabu Search (BTS), Constructive Adaptive Tabu Search (CATS) and Greedy Randomized Adaptive Serach Procedure (GRASP) on problem instance A, B and C, showing percentage gap to the best known solution over time.

What is also interesting is that all three location search algorithms find solutions within 0.5% of the best known solution within a few hours. In problem instances A and C, they also seem to converge to the same solution given enough time. Whether this final solution they agree on is actually the best solution or just a local optimum, is hard to tell. But the fact that these distinctive search methods seem to agree on the best solution is a strong indication that the best known solution is indeed a good solution.

Recall from Section 7.1.6 that there is a trade-off between speed and solution quality. This trade-off is better captured by using the primal integral g^{PI} in addition to the gap to the best known solution g^{BKS}. The solution methods are compared in

Table 7.8 with respect to these evaluation metrics. Except for problem instance A, CATS has the lowest primal integral up until 8 hours of runtime. At termination after 12 hours, BTS and GRASP perform best on problem instances A and B, respectively. On problem instance C, GRASP finds the best known solution quite fast but is punished for a slow start, finding its first solution after around 30 minutes. CATS and BTS find their first solution value after around 5 minutes. Due to the definition of g^{PI}, which is 100% until a solution is found, GRASP consequently obtains a much higher g^{PI} than the other two searches. As the convergence profile of CATS in Figure 7.7 also shows, these results indicate that CATS is fast at the beginning of the search, but that improving solutions are hard to come by after 8 hours. GRASP, which relies more on randomness, cannot guarantee to find a good solution fast but tend to perform well in the long run. In general, both CATS and GRASP manage the trade-off between speed and solution quality better than BTS.

Table 7.8: Comparison of final gap (g^{BKS}) and primal integral (g^{PI}) for Basic Tabu Search (BTS), Constructive Adaptive Tabu Search (CATS) and Greedy Randomized Adaptive Search Procedure (GRASP) on problem instances A, B and C.

		2 hours		8 hours		12 hours	
Instance	Algorithm	$g^{BKS}(\%)$	$g^{PI}(\%)$	$g^{BKS}(\%)$	$g^{PI}(\%)$	$g^{BKS}(\%)$	$g^{PI}(\%)$
A	BTS	0.039	3.372	0.039	**0.873**	**0.000**	**0.589**
	CATS	0.451	**2.520**	0.368	0.949	0.038	0.679
	GRASP	**0.021**	24.749	**0.014**	6.199	0.014	4.137
B	BTS	0.845	3.061	0.342	1.085	0.342	0.837
	CATS	**0.528**	**2.760**	**0.137**	**0.827**	0.137	**0.597**
	GRASP	0.824	25.381	0.413	6.674	**0.000**	4.549
C	BTS	**0.588**	3.100	0.020	1.010	0.020	0.680
	CATS	0.714	**3.006**	0.013	**0.882**	0.013	**0.592**
	GRASP	1.070	33.405	**0.000**	8.485	**0.000**	5.657

7.4.2 Assessment of Diversification and Intensification

One challenge in the location search algorithms is to balance diversification and intensification. First, it is important not to waste too much time on parts of the solution space with a suboptimal number of open depots. The algorithms seem to find approximately the right number of open depots in all problem instances, but there is some difference in how much effort that is put into exploring the number of open depots in proximity to the apparently best one. In Figure 7.8 we show this on problem instance C, indicated by the number of depots that are open in all explored solutions and the best found solutions during the search. After only a few evaluations, CATS and GRASP find their best solutions, where they agree that around 3 depots should open. Even though they agree on this, GRASP concentrates on exploring depot configurations with 3 to 4 depots, while CATS explores 1 to 5 depots. This indicates that the adaptive neighborhood generator in CATS directs the search towards more diversification compared to the random component in GRASP. The adaptive neighborhood generator uses the Open and Close operator more often because the probability of choosing Swap decays when it cannot find any improvement for a period of time. Hence, the range of the number of open depots explored increases. As shown in Table 7.8, CATS never finds the best known solution in problem instance C, which according to Figure 7.8 could be a result from spending too much time on diversification. Consequently, it seems like GRASP performs better on problem instances where there are no alternative depot sizes and there is a number of open depots that seems very promising.

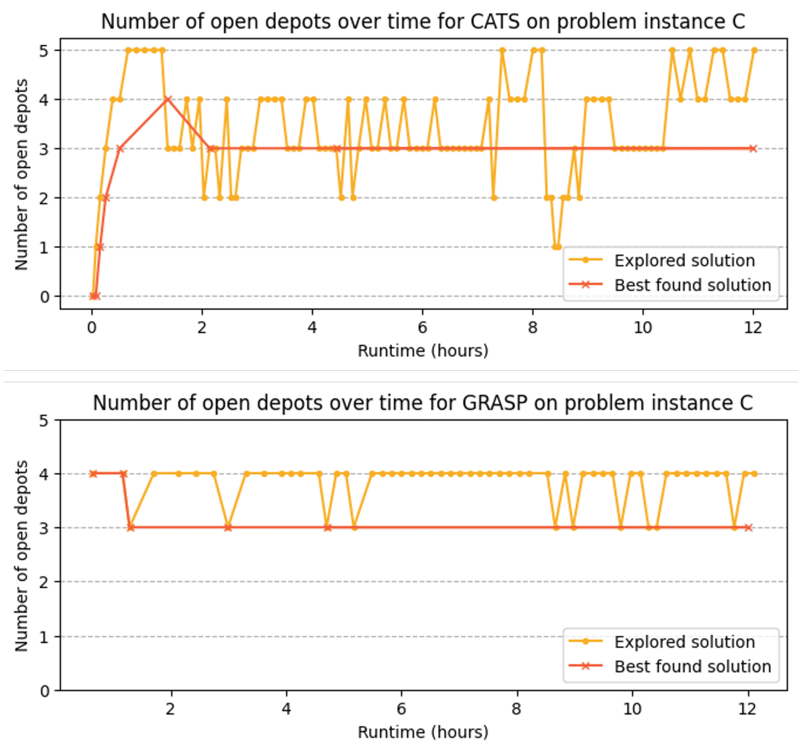

Figure 7.8: Number of open depots in all explored and best found solutions during the search on problem instance C for Constructive Adaptive Tabu Search (CATS) and Greedy Randomized Adaptive Search Procedure (GRASP).

Another aspect, which is only relevant in problem instance B, is how much time is used to explore alternative sizes versus new locations. In Figure 7.9 it becomes apparent that GRASP has depot D17 open in all depot configurations that are fully evaluated, indicating that D17 is an attractive depot. This is also true because alternative 1 for depot D17 is open in the best known solution for problem instance B. However, our experiments also show that CATS find the other alternative for D17 to be good, as it use alternative 2 to reach a gap to the best known solution of approximately 0.25% in under 3 hours. Still, GRASP never evaluates alternative 2 for depot D17. The same reasoning holds for D11, D12 and D14 as well, where the other alternative never is evaluated. Additionally, D16 is never evaluated by GRASP. This is a weakness of the GRASP algorithm, as it might ignore some alternative depot sizes if they are not good to open when only one depot should be opened. The reason why this happens is that GRASP relies only on the Open operator. Opening one depot prevents alternatives at the same location from being explored later in the same iteration because changing to the alternative is a Swap move. In other words, if one of the depot sizes is so good it always opens early on, the alternative depot size is never explored later on. Conceivably, a depot size that is not good to open alone, could nevertheless be good to open together with other depots. GRASP cannot find this solution, and this weakness can explain why CATS has a better performance than GRASP on problem instance B, where there are alternative depot sizes.

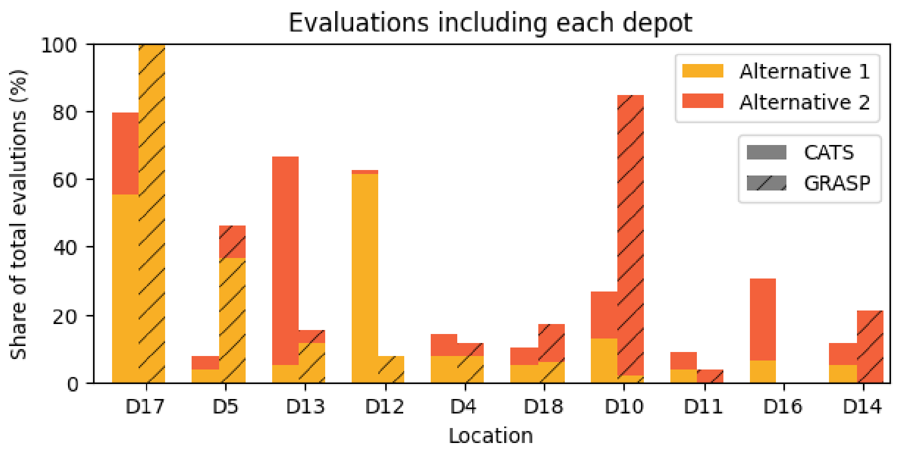

Figure 7.9: Share of evaluated depot configurations including each alternative depot for Constructive Adaptive Tabu Search (CATS) and Greedy Randomized Adaptive Search Procedure (GRASP) on problem instance B.

7.4.3 Impact of Components in Tabu Search

CATS uses a number of different heuristic components. This raises the question of how the overall performance is affected by these components. Three components that are assumed to have a big impact on the performance are the use of swap and flip tabu lists, the neighborhood sorting, as well as which neighborhood generator is used. To test the impact of these components, we individually disable or change the usage of each of them, and compare the results with CATS.

Table 7.9 presents the impact on the final gap and primal integral after 2, 8 and 12 hours when disabling the swap and flip tabu lists and the neighborhood sorting by expected mean distance to the closest depot. By disabling the neighborhood sorting, the neighbor in the neighborhood to visit next is randomly chosen. The increase in both gap and primal integral indicates that the neighborhood sorting helps to identify more promising solutions earlier in the search. However, neighborhood sorting is not a perfect measure of promising solutions, because CATS without neighborhood sorting has after 12 hours found a solution that is closer to the best known solution. This might be because more randomness is involved in choosing the next neighbor, and while this results in evaluating more poor solutions, at this point it found a better one. Less severe is the impact we see from disabling the swap and flip tabu lists. As this allows more neighbors to be visited, it seems from the results that it takes longer to find the best solutions. This implies that the solutions prohibited from these tabu lists are indeed solutions not worth evaluating. However, the low marginal difference from disabling this component shows that the most important impact comes from other components.

Table 7.9: Analysis of the final gap (g^F) and primal integral (g^{PI}) of Constructive Adaptive Tabu Search (CATS) on problem instance A when disabling one component at a time.

	2 hours		8 hours		12 hours	
Disabled component	$g^{BKS}(\%)$	$g^{PI}(\%)$	$g^{BKS}(\%)$	$g^{PI}(\%)$	$g^{BKS}(\%)$	$g^{PI}(\%)$
None	**0.451**	**2.520**	**0.368**	**0.949**	0.038	**0.679**
Swap and flip tabu lists	**0.451**	2.541	0.404	0.963	0.039	0.691
Neighborhood sorting	1.214	3.133	0.820	1.530	**0.017**	1.081

Another central component is the neighborhood generator. Recall from Section 5.1.1 that the neighborhood generator that is used in CATS, $N^{CA}(\delta)$, is a combination of two other neighborhood generators: the constructive neighborhood generator, $N^{C}(\delta)$, is used to find a reasonable number of open depots, before the adaptive neighborhood generator, $N^{A}(\delta)$ is used in the rest of the search. To assess whether this combination works well, CATS has been run using each of these neighborhood generators alone on problem instances A, B and C.

In Table 7.10, the resulting final gaps and primal integrals are compared for these runs. For the first 2 hours, the constructive neighborhood generator $N^{C}(\delta)$ obtains the lowest gap to the best known solution. The combined generator $N^{CA}(\delta)$ clearly inherits some of the benefits from the constructive generator as they both have a lower gap than adaptive generator $N^{A}(\delta)$. The combined generator also has the lowest primal integral on all problem instances so far, meaning it finds good solutions fastest. After 8 hours, the combined generator outperforms the other neighborhood generators on problem instances A and C. On problem instance B, it also has the lowest primal integral, but the adaptive generator performs better in terms of the gap to the best known solution. We know that the adaptive phase of the combined generator is less biased toward opening and closing depots than the adaptive generator. This is because the adaptive generator learns from the positive experience of opening depots at the beginning of the search when starting at no open depots. However, this experience comes at the cost of taking longer to reach good solutions, as shown by the higher primal integral. All in all, these results indicate that the combined neighborhood generator $N^{CA}(\delta)$ performs better than the other ones overall, especially when it comes to finding good solutions fast. However, due to randomness and different behavior in the generators, Table 7.10 shows that $N^{CA}(\delta)$ is not the highest-performing strategy at all points of the search.

Table 7.10: Final gap (g^{BKS}) and primal integral (g^{PI}) for the Constructive Adaptive Tabu Search (CATS) with combined $N^{CA}(\delta)$, constructive $N^{C}(\delta)$ and adaptive $N^{A}(\delta)$ neighborhood generators.

		2 hours		8 hours		12 hours	
Instance	N	$g^{BKS}(\%)$	$g^{PI}(\%)$	$g^{BKS}(\%)$	$g^{PI}(\%)$	$g^{BKS}(\%)$	$g^{PI}(\%)$
A	$N^{C}(\delta)$	**0.448**	2.546	0.404	0.959	0.039	**0.667**
	$N^{A}(\delta)$	0.454	2.527	0.404	0.961	0.039	0.731
	$N^{CA}(\delta)$	0.451	**2.520**	**0.368**	**0.949**	**0.038**	0.679
B	$N^{C}(\delta)$	**0.520**	3.808	0.195	1.111	0.195	0.805
	$N^{A}(\delta)$	0.620	4.563	**0.000**	1.491	**0.000**	0.994
	$N^{CA}(\delta)$	0.528	**2.760**	0.137	**0.827**	0.137	**0.597**
C	$N^{C}(\delta)$	**0.420**	4.904	0.420	1.540	0.420	1.166
	$N^{A}(\delta)$	1.158	5.868	0.015	1.714	0.015	1.148
	$N^{CA}(\delta)$	0.714	**3.006**	**0.013**	**0.882**	**0.013**	**0.592**

7.5 Analysis of Attributes Influencing Costs

As well as analyzing how we find the depot configurations with the lowest cost, we are interested in what affects the costs. In the following section, we analyze how the total costs and the cost components relate to solution attributes such as the number of depots open and which depots are included in the solution. Subsequently, an in-detail analysis of some specific solutions is performed. To keep figures displaying depots compact, we use problem instance C, which has the fewest number of possible depots, for the analysis in this section. The trends are, however, the same in the other datasets as well.

7.5.1 Impact from the Number of Open Depots

Some of the neighborhood generators that are used in the search, like the constructive neighborhood generator, are based on a hypothesis that the best solutions have the same number of open depots. Figure 7.10 shows that the best solutions are indeed centered around three open depots. However, we also see that there are big overlaps in the costs of different numbers of depots. For instance, there are solutions with only 1 open depot that have lower costs than some solutions with 3 open depots. From the colors in the plot, we see that another factor that seems just as important as the number of open depots is the number of vehicles available in the depots, or the depot capacity. 8 of the 10 best solutions found in problem instance C have 36 vehicles available in the depot, and within the same number of open depots, the solution cost seems to be highly correlated with the number of vehicles available.

Figure 7.11 shows how the number of open depots affects the cost components in the best solutions found. We see that, as expected, the depot opening costs increase as

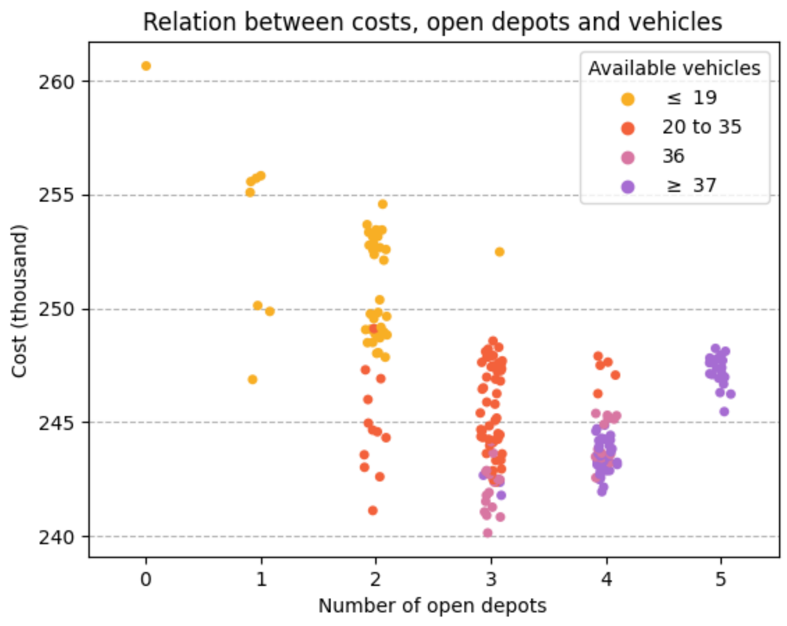

Figure 7.10: The relation between costs and the number of open depots in the best solutions, as well as the number of vehicles available in the open depots (shown by colors). Each dot represents an evaluated solution in problem instance C.

more depots open. More interesting is that the fleet cost remains about the same, meaning that the number of vehicles used in total is almost independent of how many depots are used. This might be affected by a skew in the objective of the HGSR solver compared to our problem, as it minimizes the time used and does not take into account the number of vehicles itself. But the fact that it changes so little, even though many solutions have been considered by our search algorithms, is a strong indication that there is not much to save when it comes to the number of vehicles itself.

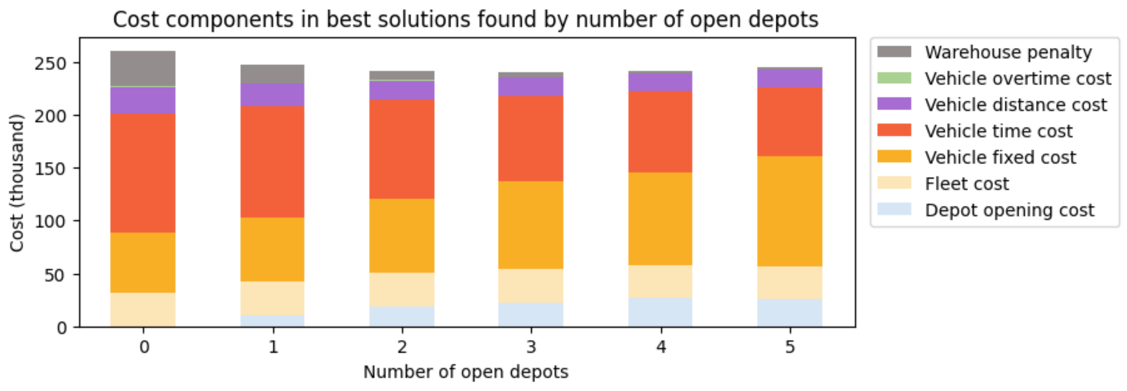

Figure 7.11: Cost components in the best solution found for each number of open depots in problem instance C.

However, by increasing the number of depots, it is possible to reduce vehicle costs. With more depots, the routes become shorter, resulting in lower distance and time used, when depots are closer to the customers. This has a direct impact on the costs of driving the routes. Furthermore, we also see a shift in which type of vehicles that is used. The warehouse predominantly has time-cost vehicles, whereas the depots have a more moderate mix of fixed and time-cost vehicles (see Appendix C). When no depots are open, the warehouse supplies all customers and it seems like the time-cost vehicle is the most attractive type. However, when increasing the number of depots the vehicle fixed costs increase while time costs decrease. In this case, a fleet mix with more fixed-cost vehicles becomes better. It is also worth noting that the vehicle overtime cost is so small that it is negligible.

At last, there is the warehouse penalty cost. We see that without this cost, the solution with 0 open depots would be the best found. In other words, the reduction in vehicle costs with an increasing number of depots is less significant than the increase in depot opening costs. This penalty is, as discussed in Section 5.2.1, modeled to reflect restrictions in capacity at the warehouse, so by this, we show that the warehouse capacity is very important in deciding how many depots that should open. As its impact on the results is so important, this penalty must be modeled as realistically as possible when applied to specific business contexts.

7.5.2 Impact from the Depot Attributes

In addition to how many depots are open, which depots are open is also important. Figure 7.12 shows for each depot the cost of all evaluated depot configurations in which this depot is open. The minimum costs vary from depot to depot, which must be seen as a result of the combination of depot attributes.

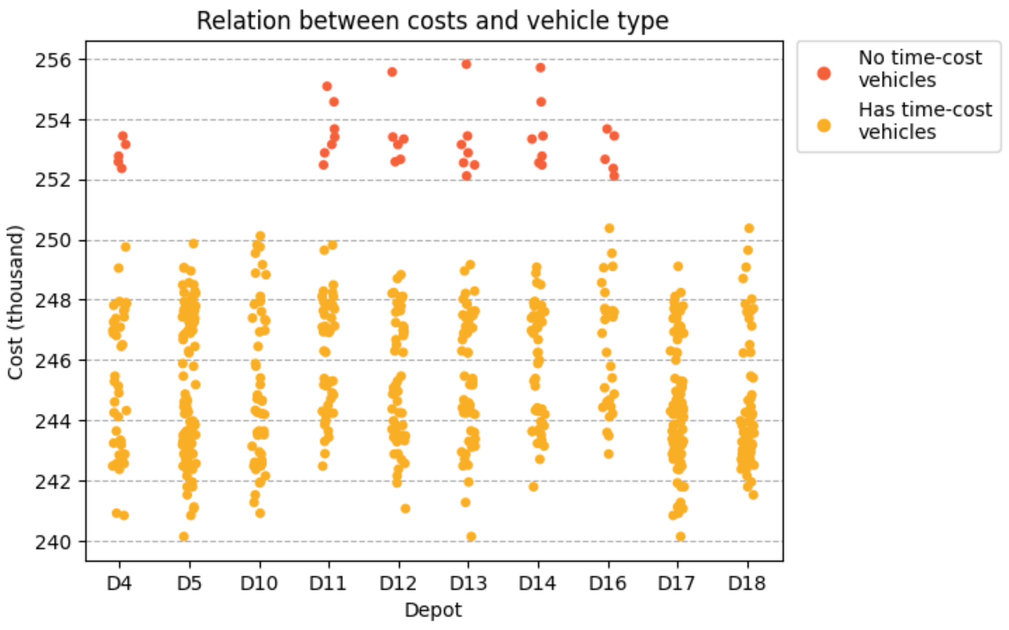

Figure 7.12: The cost of all evaluated depot configurations in which each depot is open. Colors indicate whether the open depots in the depot configuration have time-cost vehicles available or not.

One attribute that shows strong indications of having a big influence on the cost is the number of time-cost vehicles available. In Figure 7.12 we see that all the highest-cost solutions only include open depots without time-cost vehicles available. This is further highlighted by Figure 7.13, which shows the correlation between the solution costs and the percentage of time-cost vehicles that are available in the open depots. There is a tendency that more time-cost vehicles available mean lower solution costs, as shorter routes are cheaper when driven by a time-cost vehicle. Because the number of customers with open time windows is not constant during a shift, shorter routes are beneficial to handle the busiest part of the shift. However, fixed-cost vehicles are useful as well, and the best solutions found often include one depot that has only fixed-cost vehicles. For example, D13 has only fixed-cost vehicles and is open in the best solution to problem instance C. This indicates that other depot attributes might be important as well.

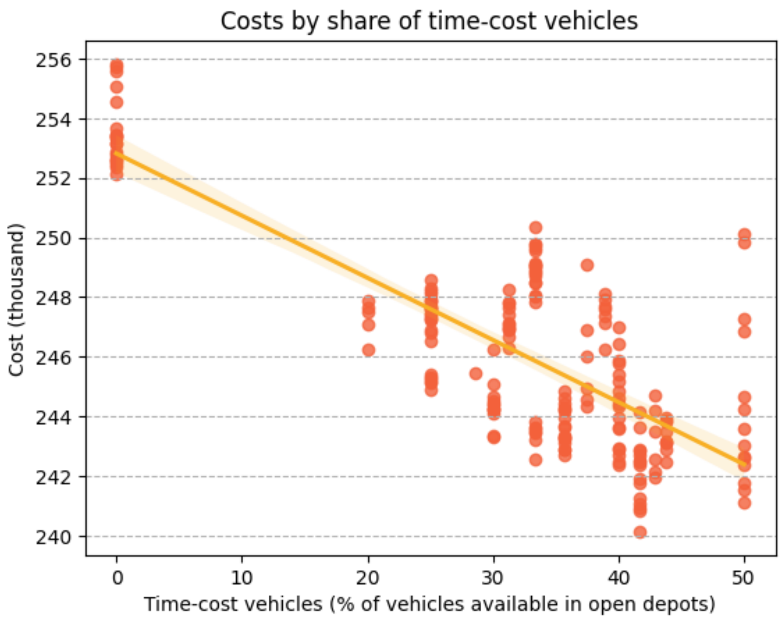

Figure 7.13: The percentage of time-cost vehicles that are available in the open depots for all solutions found in problem instance C compared to the cost.

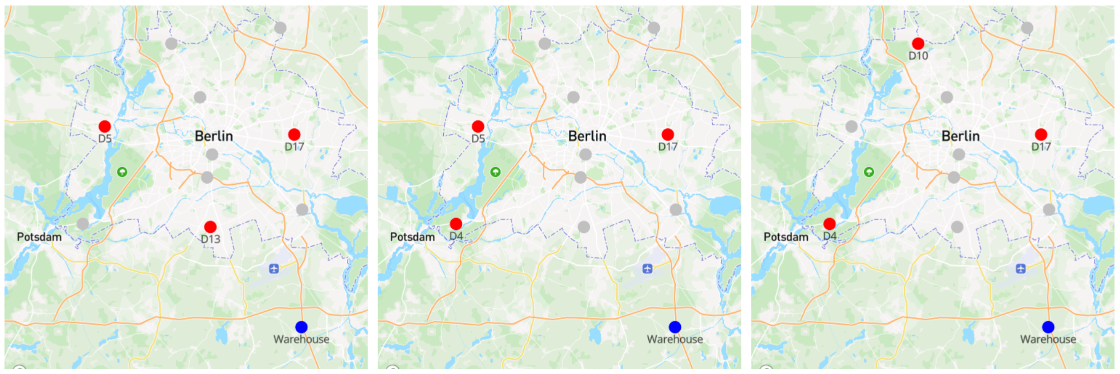

Figure 7.14: The open depots in the 3 best solutions found for problem instance C. The best solution is to the left.

Another indication of an important attribute can be seen on the maps of the open depots in the 3 best found solutions for problem instance C in Figure 7.14. We see that the best found depot locations vary, but have a certain spread across the city; the depots are not too close to each other. They also seem to be placed in the suburbs of the city rather than the city center. It is hard to say whether this is because of higher leasing costs in the city center, or that the location better allows them to serve the suburban customers that are the furthest away from the warehouse, but it might be thought that both are important factors.

7.6 Case Study for Oda

In this section, we answer three main questions for Oda. First, at how many customers does opening depots become viable? Second, what effect does the vehicle fleet composition have on finding the best depots? Third, what could make opening depots more attractive?

To answer these questions, we start by establishing a base case for how many customers are needed to make depots necessary. Next, analysis is performed to determine the impact of the initial vehicle fleet and cost estimates for depots. Finally, an analysis of what the results mean for Oda is performed. To achieve this, we use different parameters for the penalty function as mentioned in Section 7.1, 500 for every vehicle above 120 that the warehouse dispatches. This represents the large capacity that Oda has at the main warehouse. The O problem instances used in this section are designed to give Oda information on both the number of depots and what size they should be. We also want the results to give an estimate of the actual costs to Oda. Therefore, we start by finding a constant that makes our algorithm yield costs that are very close to Navegante's costs in Section 7.6.1.

7.6.1 Adjustment of the Routing Costs

To make the analysis even more relevant for Oda, we find an adjustment constant that minimizes the average gap between the costs found by our algorithm and benchmarks from Navegante. Routing problems T1-T23 are used, which vary in the number of depots and the number of customers as shown in Table C.3 in Appendix C.

The results for the cross-validation test is shown in Table 7.11. AVG is the average of the gap values on all 5 splits. The minimum MSE method results in the average lowest gap in the cross-validation test. Therefore, we select the minimum MSE method when choosing how to calculate the adjustment constant. This gives an adjustment constant of 0.917. This gives a MAPE of 3.88% after adjustment. In comparison with the estimated costs before adjustment, presented in Section 7.3, the adjustment leads to a reduction in MAPE of 5.19%. We show the effect of adjusting our routing costs with the constant on the routing problems T1-T23 in Figure 7.15. As can be observed, some larger errors occur for T4, T11, T20 and T21, but in general the errors are small. Seen in conjunction with Section 7.3, where it is shown that HGSR has a Kendall's τ correlation coefficient of 0.937, the estimates are considered accurate.

Table 7.11: Cross validation of adjusted routing cost with 5 folds. The constant that minimizes the mean squared error (MSE), the mean gap and the median gap of the training fold are tested as adjustment constants (C). The gap is the mean percentage gap for the solutions in the test folds. Average gap (AVG) of the 5 folds are shown.

	Minimum MSE		Mean		Median	
Split	Gap (%)	C	Gap (%)	C	Gap (%)	C
1	1.121	0.917	0.648	0.913	2.241	0.928
2	3.963	0.926	3.220	0.919	4.479	0.930
3	-2.490	0.912	-3.420	0.904	-0.544	0.931
4	-2.325	0.913	-3.195	0.905	-1.055	0.925
5	-0.969	0.915	-1.763	0.908	-0.676	0.931
AVG	-0.140		-0.902		1.159	

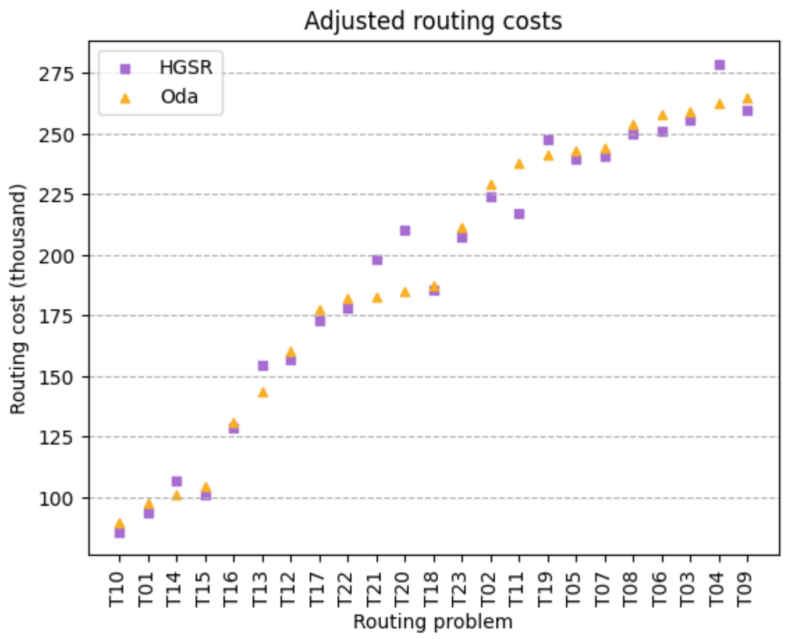

Figure 7.15: Routing cost from Oda's solver, Navegante, compared with the estimated routing cost from our solution method using Hybrid Genetic Search Router (HGSR) after adjustment by 0.917.

7.6.2 The Effect of Increasing Number of Customers

With problem instance O2, although the largest scenarios would benefit from opening a depot, it would not be beneficial on average because of the scenarios with fewer customers. Hence, as long as Oda considers what is best over many different realizations of customer presence, they should not open depots if they expect that an average of 2000 customers place orders. Figure 7.16a shows how the lowest cost found does not change during the search, as the first solution checked is opening no depots.

When expecting around 3000 customers, opening a depot becomes economically viable. As we can see from Table 7.12, opening D17-6 is the cheapest alternative with O3. This alternative is found within an hour after starting the search, and no better solution is found during the next 11 hours, as illustrated in Figure 7.16b.

When expecting 4000 customers on average in O4, depot D10-24 becomes the cheapest alternative. However, opening the depot D17-24 instead increases the objective value by only 200, or 0.05%. This solution could be advantageous if they already purchased the D17-6 depot when expecting 3000 customers, as it requires only an upgrade to an already owned location. If D17-6 was already acquired, additionally opening D10-24 with 4000 customers would yield a worse solution. For this problem, Figure 7.16c shows that the search progresses to better solutions several times during the 12 hours.

Finally, when expecting an average of 5000 customers in O5, opening D17-24 and D10-18 together gives the lowest distribution costs. If Oda expects to have this many customers, they could acquire the two locations D10 and D17 and potentially upgrade them later. Up to 233 vehicles are required to serve the largest scenarios of 7500 customers. This makes using only the main warehouse impractical, and Figure 7.16d shows how the search quickly finds better solutions, in which vehicles are distributed at different depot locations.

Table 7.12: Overview of the best solutions for problem instances O2, O3, O4 and O5, whose average shift demands are 2000, 3000, 4000 and 5000 customers, respectively.

Problem instance	Expected shift cost	Open depots
O2	206,078	None
O3	304,275	D17-6
O4	403,935	D10-24
O5	509,491	D10-18, D17-24

When exceeding 3000 customers on average, the original capacity at the main warehouse is too low to serve all customer demand. Even with the possibility to extend the warehouse fleet with a penalty cost, as described in 5.2.1, this makes opening depots necessary to distribute vehicle loading. The main insight from varying the average number of customers is that depots should be opened when the capacity at the main warehouse is exceeded. Then, it is valuable to investigate how the required number of vehicles should be distributed at different depot locations so that costs are minimized.

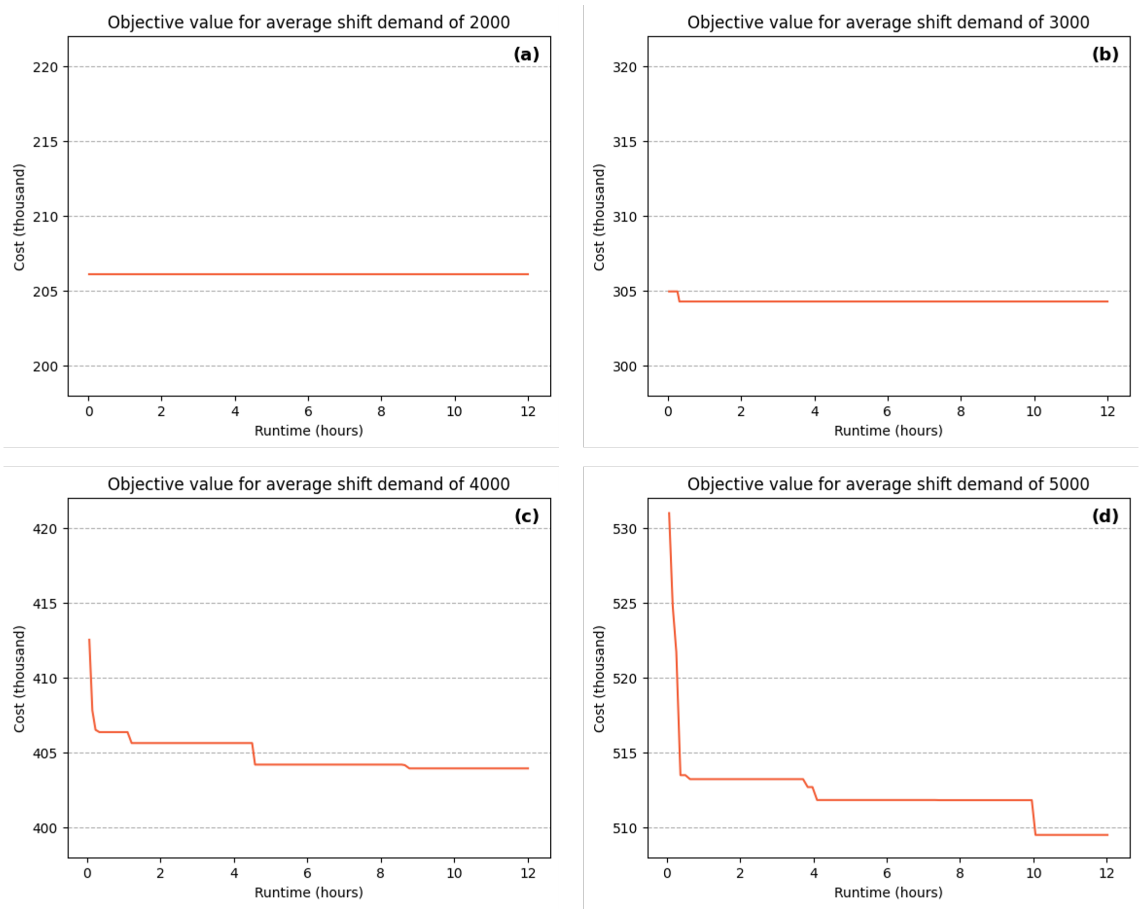

Figure 7.16: Convergence profiles for problem instances O2, O3, O4 and O5, whose average shift demands are 2000, 3000, 4000 and 5000 customers, respectively.

7.6.3 The Effect of Changing the Fleet Composition

In the problem instances used so far, there has been a mix of fixed-cost vehicles and time-cost vehicles at the depots and the warehouse. Some depots have only fixed-cost vehicles, whereas others have a mix between fixed-cost vehicles and time-cost vehicles. The main warehouse has mostly time-cost vehicles. To analyze what effect the vehicle type has, we run two tests with a fleet consisting of only fixed-cost vehicles with high fixed costs in the first test and a fleet consisting of only time-cost vehicles with higher variable costs in the second test.

Interestingly, both using only time-cost vehicles and only fixed-cost vehicles are more expensive than the reference split. Table 7.13, which shows solutions for different splits of time-cost vehicles and fixed-cost vehicles, indicates two points. First, it implies that the duration of most routes is shorter than a shift because the routing costs are significantly higher for the 100/0. As discovered in Section 7.1, a fixed-cost vehicle must drive for 6.9 hours before being cheaper than a time-cost vehicle. Therefore, if route durations matched the 8-hour shifts, one would expect that using

Table 7.13: Costs for problem instance O3 with varying fleet composition, indicated by split between percentage of fixed-cost vehicles and time-cost vehicles in the fleet.

Split	Solution	
Fixed-cost/Time-cost	Cost	Best depot
100/0	325,890	None
Reference	304,275	D17-6
0/100	306,548	D10-12

only fixed-cost vehicles would be better. However, because the case with only time-cost vehicles is also more expensive than the reference case, it shows that there are some routes long enough that fixed-cost vehicles become the cheaper option. This indicates that there is a trade-off between paying a large fixed price for having a vehicle for an entire shift and paying a cost that increases with the duration of a vehicle route. Exactly where this balance lies depends on the relative cost levels of the two types of vehicles.

7.6.4 Determinants of Depot Attractiveness

Given the costs used in the previous tests, routing costs do not offset the increased costs from opening depots. As mentioned, the argument for opening depots is mainly to move vehicles away from the main warehouse. However, we shift the focus toward what cost changes that would make opening depots more attractive. We introduce an assumption that for each scenario it is possible to either operate depots or use only the main warehouse. When using only the main warehouse, leasing costs for the closed depots must still be paid. Then, depots should be open only on shifts with sufficient customers, which could potentially make larger depots more attractive.

Here, fixed costs and time costs for the vehicles are added together into a salary category. Salary accounts for the vast majority of costs, as illustrated in Table 7.14. The figure shows the normal depot cost for problem instance O3, which according to Table 7.12 corresponds to having D17-6 open, and flexible depot costs, which means that D17-6 can be closed when not needed. Although there is a reduction in time cost and an increase in fixed costs when opening D17-6 only when needed, when the differences in time costs and fixed costs are added together as salary, the net difference is negligible. One way that using depots can lower salary costs is by having a higher ratio of time-cost vehicles to fixed-cost vehicles. When depots are closer to the customers, the total driving times of the vehicles are shorter. As long as the number of vehicles in use stays the same, only time-cost vehicles become cheaper when the driving time is shorter. Although using fixed-cost vehicles at depots may improve the solution if they require fewer total vehicles, this is likely not the case as depots usually serve total demand close to the capacity of their vehicles. Therefore, any savings due to shorter routes are achieved by time-cost vehicles. As described in Section 7.5, having time-cost vehicles at the depots is found to be beneficial, and the fact that opening depots give access to more time-cost vehicles makes opening depots valuable. Interestingly, both depots opened in Table 7.12 have high ratios of time-cost vehicles. The depots that are not open in any of the best solutions, D12 and D13, have no time-cost vehicles.

Table 7.14: Change in cost composition when allowing flexible opening of depots for problem instance O3. This corresponds to having the option to open and close depot D17-6 when profitable. Bold marks where the cost is the cheapest.

Cost type	Normal depot cost	Flexible depot cost
Depot opening cost	3,978	**3,282**
Fleet cost	49,617	49,617
Salary to drivers	**219,113**	219,230
Vehicle distance cost	**30,053**	30,313
Vehicle overtime cost	**574**	586
Warehouse penalty	940	940
Total	304,275	**303,968**

When testing flexibly opening depots in scenarios with sufficient customers on all O problem instances, there is no change in which depots are best to open. Nevertheless, there are cost savings of between 0.1% to 0.36% when closing depots when they are not needed, as Table 7.15 shows. These cost savings arise mainly from saving the cost of driving line-haul trucks to the depots, which is included in the depot opening costs. When there are not enough customers, the extra cost of a line haul overshadows the routing costs saved from driving from depots instead of the warehouse. On the other hand, there is a small increase in driver salary, overtime cost and distance cost, as more trips must be made from the warehouse. Although the difference in total cost for O3 seems small, NOK 307 saved for 60 shifts each month for 12 months is over NOK 200,000 saved yearly. The same calculation for O5 yields over NOK 1.3 million saved yearly.

Table 7.15: Effect of flexibly operating depots only in scenarios with a sufficient number of customers for problem instances O3, O4 and O5, whose average shift demands are 3000, 4000 and 5000 customers, respectively.

Problem instance	Cost with normal depots	Cost with flexible depots	Change (%)
O3	304,275	303,968	-0.10
O4	403,935	402,761	-0.16
O5	509,491	507,637	-0.36

Chapter 8

Concluding Remarks

To survive in the online grocery market it is necessary to develop highly efficient distribution networks. The research in this thesis is motivated by the problem faced by the online grocery retailer Oda. They face a problem of where to locate depots, from which last-mile delivery vehicles are driven to customers, to minimize both depot opening costs and vehicle routing costs. Additionally, there is uncertainty related to which customers place an order on a given day, also called stochastic customer presence. The problem can therefore be categorized as a Location Routing Problem (LRP) with stochastic customer presence. To find the routing costs, a very large multi-depot capacitated Vehicle Routing Problem (VRP) with time windows, heterogeneous fleet and tour duration constraints is solved.

Even though some literature has considered aspects of our problem, we observe that the complexity of the LRP, especially with respect to the VRP solved to compute the routing costs, surpasses previous research. To close this gap, this thesis contributes with a two-stage stochastic solution method for the LRP with stochastic customer presence, including a scenario generation method. For the first-stage problem of searching through combinations of open depots, or depot configurations, we propose two different solution methods. One of these is a Constructive Adaptive Tabu Search (CATS) that first finds a promising number of open depots, and then performs a local search with adaptive neighborhood probabilities. The other is a Greedy Randomized Adaptive Search Procedure (GRASP) which evaluates only one scenario per depot configuration in a construction phase, enabling high efficiency through parallel computing, before fully evaluating the most promising solutions. For the second-stage VRP, the contribution is a three-phase algorithm that first decomposes the multi-depot problem into several single-depot problems, then clusters customers to super-customers, and finally creates routes with a state-of-the-art Hybrid Genetic Search algorithm.

The solution methods are tested on real-world problem instances in Berlin with 12 hours of computational time. Additionally, the second-stage routing algorithm is separately benchmarked against Oda's routing solver, Navegante. The results show that the proposed solution methods find solutions with a gap of less than 1% to the best known solutions in under 2 hours on all problem instances. The estimated routing costs are found to have a mean absolute percentage error of 3.88% to Navegante after adjustment by a constant factor of 0.917, making it highly accurate on average. This is promising considering that our routing algorithm uses 95% less time than Navegante. Stability testing of our approach to handling stochastic customer presence indicates that a scenario tree with 16 scenarios is sufficient to obtain stability. At this number of scenarios, we observe that generating a new scenario tree can change the objective value by 3.46%. Nevertheless, Kendall's τ correlation coefficient when comparing the ranking of depot configurations across two scenario trees is at least 0.82. On average, the relative difference in costs between misplaced depot configurations is only 0.32%, indicating that the scenario generation has little impact on the solution quality. By using CATS on a problem with 4 locations and 4 alternative sizes at each location, we identify that Oda should open their first depot when exceeding an average of 3000 customer orders in a shift, but that vehicle fleet size and composition have a considerable effect on this decision.

The aim of this thesis is to answer to what degree the proposed solution methods are able to efficiently find high-quality solutions for this problem, and how the stochastic customer presence can be modeled. Both of our solution methods are promising, but have different applications. CATS is found to perform better on problem instances with alternative depot sizes at each location and is found to have a good performance overall. GRASP achieves good results in some cases, especially where there are no alternative depot sizes, but is somewhat less reliable in terms of how fast good solutions are found. Regarding the scenario generation method, we interpret our results as an important step towards putting more emphasis on how stochastic customer presence can be modeled. We learned that testing stability is not straightforward when using heuristics with random components as increasing the number of scenarios only improves stability up to a point. By introducing the test of ranking using τ, we are however still able to conclude that the scenario generation method is stable enough for our application. Furthermore, because the model finds good results in just a few hours, and strategic decision processes may take weeks, this model can be run repeatedly with different input data to give highly valued insight for making the best decisions.

Chapter 9

Future Research

There are several opportunities for future research within the field of LRPs with stochastic customer presence, both with respect to how the problem is modeled and how it is solved. Based on the experience from the research in this thesis, we suggest that the focus should be on fleet decisions as part of stochastic programming, revenue management and scalability.

The literature review in Chapter 3 shows that there are few papers that use stochastic programming for the LRP. In this thesis, we modeled the LRP with stochastic customer presence as a two-stage model. We show that fleet composition and size of depots are important contributors to the best solution. Therefore, a proposition for future research is to include more detailed decisions regarding fleet size and possibly fleet mix in the first stage. Potentially, the introduction of electric vehicles could present new considerations as a part of the fleet. Moreover, a more granular investigation of the number of vehicles at each depot, and conceivably the allocation of a fixed-size fleet across depots could be insightful.

A closely related extension is to consider revenue management, where it is decided what areas to serve in addition to locating the depots and deciding their size. An underlying assumption of the LRP is that the supply is at least as high as the total demand of customers. However, in the real world, decision-makers must maximize the profit from only a limited amount of resources. For this reason, both Oda and Mara et al. (2021) emphasize this extension. In connection to our problem, this means that the online grocery retailer is forced to not serve certain customers because of limited resources, for instance in terms of the fleet size. To achieve this, a notion of profit, the decision to not serve a customer and the consequence of this decision must be added to the model. In our view, this is a decision that would have been especially interesting to take in conjunction with the fleet size, because this would introduce a trade-off between fleet size and which customers to serve.

One of the main contributions of this thesis is to solve very large VRPs as the subproblem of the LRP. A challenge with such a complex problem is the trade-off between solution quality and computational time. To reduce computational time, we suggest three directions: techniques for pruning the solution space, prioritizing which solution to evaluate and machine learning. Techniques for pruning the solution space can be to use a simpler mathematical model to find a bound on the number of depots that should open or discard solutions that are determined to be worse by a confidence level. The latter is most applicable when the differences in the best objective values are greater than in this thesis. Prioritizing which solution to evaluate includes sorting the neighboring solutions by some metric. Our solution methods use the average distance between customers and their nearest depot, but a rough evaluation of the cost by a function or a very simple heuristic might also be possible. Several sorting methods could also be used interchangeably depending on when they work best, which might lead to even more efficiency in the search. A final way to manage the trade-off could be to use machine learning to estimate the costs of the VRP and not create any routes, as solving the VRP in this problem is only performed to estimate the cost of a depot configuration. The analysis of attributes influencing costs in Section 7.5 indicates a strong correlation between costs and features like the number of open depots, number of customers and available vehicle types, all of which could have been used as features for machine learning. Future research in any of these directions could enable even more complex and realistic LRPs to be solvable in a reasonable time.

Bibliography

Albareda-Sambola, M., Fernández, E., and Laporte, G. (2007). Heuristic and lower bound for a stochastic location-routing problem. *European Journal of Operational Research*, 179:940–955.

Arnold, F., Gendreau, M., and Sörensen, K. (2019). Efficiently solving very large-scale routing problems. *Computers & Operations Research*, 107:32–42.

Arnold, F. and Sörensen, K. (2019). What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. *Computers & Operations Research*, 106:280–288.

Baldacci, R., Mingozzi, A., and Calvo, R. W. (2011). An exact method for the capacitated location-routing problem. *Operations research*, 59(5):1284–1296.

Belenguer, J.-M., Benavent, E., Prins, C., Prodhon, C., and Wolfler Calvo, R. (2011). A branch-and-cut method for the capacitated location-routing problem. *Computers & operations research*, 38(6):931–941.

Berthold, T. (2013). Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6):611–614.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Contardo, C., Cordeau, J. F., and Gendron, B. (2014). A GRASP+ILP-based metaheuristic for the capacitated location-routing problem. *Journal of Heuristics*, 20:1–38.

Corredor-Montenegro, D., Consuegra-Laino, M. J., Solano-Blanco, A. L., and Gómez, C. (2021). Vehicle fleet sizing, positioning and routing problem with stochastic customers. `http://arxiv.org/abs/2109.08114`. Last accessed on May 30, 2023.

CVRPLIB (2023). Vehicle routing problem benchmarks. `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/`. Last accessed on May 30, 2023.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

De Maio, A., Musmanno, R., and Vocaturo, F. (2022). Unbiased decision making in location-routing problems with uncertain customer demands. *Soft Computing*. doi:10.1007/s00500-022-06785-7.

DIMACS (2022). VRPTW competition updates. `http://dimacs.rutgers.edu/programs/challenge/vrp/vrptw/vrptw-competition/`. Last accessed on May 30, 2023.

Dondo, R. and Cerdá, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176:1478–1507.

Duhamel, C., Lacomme, P., Prins, C., and Prodhon, C. (2010). A GRASP×ELS approach for the capacitated location-routing problem. *Computers & Operations Research*, 37:1912–1923.

E24 (2022). Omsetningen til Oda steg 25 prosent i fjor. `https://e24.no/naeringsliv/i/lVL9zL`. Last accessed on May 30, 2023.

Feo, T. and Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Ferreira, K. M. and de Queiroz, T. A. (2018). Two effective simulated annealing algorithms for the location-routing problem. *Applied soft computing*, 70:389–422.

Finansavisen (2021). Kahoot, Gelato, Autostore, Oda og Cognite er blitt norske enhjørninger. `https://www.finansavisen.no/lordag/reportasje/2021/10/29/7762220`. Last accessed on May 30, 2023.

Gehring, H. and Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *EUROGEN99*, pages 57–64.

Gendreau, M. and Potvin, J.-Y. (2019). *Handbook of Metaheuristics*, chapter Tabu Search, pages 37–55. Springer International Publishing, Cham.

Giosa, I. D., Tansini, I. L., and Viera, I. O. (2002). New assignment algorithms for the multi-depot vehicle routing problem. *Journal of the Operational Research Society*, 53:977–984.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549.

GPS 56 (2017). 2007 MAN. `https://www.flickr.com/photos/91807507@N03/34363739282/`. Last accessed on May 30, 2023. Image licensed under CC BY-ND 2.0.

Gulaker, D., Indbjo, J. L., and Sørensen, M. W. (2022). Quickly evaluating depot configurations in very large last-mile delivery problems using decomposition and routing heuristics. Project report.

Guo, Z., Wallace, S. W., and Kaut, M. (2019). Vehicle routing with space- and time-correlated stochastic travel times: Evaluating the objective function. *Informs Journal on Computing*, 31:654–670.

Helsgaun, K. (2017). An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. doi:10.13140/RG.2.2.25569.40807.

Helsgaun, K. (2022). Lkh-3. `http://webhotel4.ruc.dk/~keld/research/LKH-3/`. Last accessed on May 31, 2023.

Javid, A. A. and Azad, N. (2010). Incorporating location, routing and inventory decisions in supply chain network design. *Transportation Research Part E: Logistics and Transportation Review*, 46:582–597.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1-2):81–93.

Klibi, W., Lasalle, F., Martel, A., and Ichoua, S. (2010). The stochastic multiperiod location transportation problem. *Transportation Science*, 44:221–237.

Klose, A. and Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, 162(1):4–29.

Kool, W., Juninck, J. O., Roos, E., Cornelissen, K., Agterberg, P., van Hoorn, J., and Visser, T. (2022). Hybrid genetic search for the vehicle routing problem with time windows: a high-performance implementation. *12th DIMACS Implementation Challenge Workshop.* `https://wouterkool.github.io/pdf/paper-kool-hgs-vrptw.pdf`. Last accessed on May 30, 2023.

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.*, 21:498–516.

Mara, S. T. W., Kuo, R. J., and Asih, A. M. S. (2021). Location-routing problem: a classification of recent research. *International Transactions in Operational Research*, 28:2941–2983.

Maranzana, F. E. (1964). On the location of supply points to minimize transport costs. *Operational Research Quarterly*, 15(3):261–270.

McKinsey & Company and Eurocommerce (2023). Living with and responding to uncertainty – state of grocery retail 2023: Europe. `https://www.mckinsey.com/industries/retail/our-insights/state-of-grocery-europe-2023-living-with-and-responding-to-uncertainty`. Last accessed on May 30, 2023.

Nagy, G. and Salhi, S. (1996). Nested heuristic methods for the location-routeing problem. *The Journal of the Operational Research Society*, 47:1166.

Nagy, G. and Salhi, S. (2007). Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672.

Oda (2022). Transport - Bærekraft hos Oda.com. `https://sustainability.oda.com/transport`. Last accessed on May 30, 2023.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435.

Prodhon, C. and Prins, C. (2014). A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1–17.

PyVRP (2023). PyVRP v0.1.0. https://github.com/PyVRP/PyVRP/releases/tag/v0.1.0. Last accessed on May 30, 2023.

Salhi, S. and Rand, G. K. (1989). The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156.

Simmons, V., Spielvogel, J., Timelin, B., and Gi, M. T. P. (2022). The next S-curve of growth: Online grocery to 2030. https://www.mckinsey.com/industries/retail/our-insights/the-next-s-curve-of-growth-online-grocery-to-2030. Last accessed on May 30, 2023.

Thedens, T. and Hachibiti, M. (2022). Ultimate overview of online food retailing in Germany in 2022 and beyond. https://www.efoodinsights.com/de-online-grocery-report/. Last accessed on May 30, 2023.

Tordecilla, R. D., Panadero, J., Juan, A. A., Quintero-Araujo, C. L., and Montoya-Torres, J. R. (2020). A simheuristic algorithm for the location routing problem with facility sizing decisions and stochastic demands. *2020 Winter Simulation Conference (WSC)*, 2020-December:1265–1275.

Vidal, T. (2022). Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140:105643.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489.

Voigt, S., Frank, M., Fontaine, P., and Kuhn, H. (2022). Hybrid adaptive large neighborhood search for vehicle routing problems with depot location decisions. *Computers & Operations Research*, 146:105856.

Wallace, S. W. and King, A. J. (2012). *Modeling with Stochastic Programming.* Springer New York.

Walpole, R. E., Myers, R. H., Myers, S. L., and Ye, K. (2012). *Probability & Statistics for Engineers*, volume 9. Pearson / Prentice Hall, United States of America.

Yu, X., Zhou, Y., and Liu, X.-F. (2019). A novel hybrid genetic algorithm for the location routing problem with tight capacity constraints. *Applied soft computing*, 85:105760.

Zhang, S., Chen, M., and Zhang, W. (2019). A novel location-routing problem in electric vehicle transportation with stochastic demands. *Journal of Cleaner Production*, 221:567–581.

Appendix A

Benchmarking of Routing Solvers

Literature benchmarks of routing solvers from Chapter 3. Table A.1 shows the best known solution values of the large Gehring and Homberger (1999) problem instances with 1000 customers, and is included to give context to the results in Table A.2. R1, R2, C1, C2, RC1 and RC2 are different problem instances using different customer distributions. The values used to compute the gaps in Table A.2 for HGSADC, HGSR and LKH-3 are retrieved from Vidal et al. (2013), the results from DIMACS (2022) and the computational results available from Helsgaun (2022), respectively.

Table A.1: Benchmark for the vehicle routing problem with time windows. Best known solution values on the large Gehring and Homberger (1999) problem instances with 1000 customers. Retrieved 4th of May 2023 from CVRPLIB (2023).

	R1	R2	C1	C2	RC1	RC2
1	53,046.5	36,881.0	42,444.8	16,841.1	45,790.8	28,122.6
2	48,263.1	31,241.9	41,337.8	16,462.6	43,678.3	24,248.6
3	44,677.1	24,399.0	40,064.4	16,036.5	42,122.0	19,618.1
4	42,440.7	17,811.5	39,434.1	15,459.5	41,357.4	15,657.0
5	50,406.7	34,132.8	42,434.8	16,521.3	45,028.1	25,797.5
6	46,930.3	29,124.7	42,437.0	16,290.7	44,903.6	25,782.5
7	43,997.4	23,102.2	42,420.4	16,378.4	44,417.1	24,395.8
8	42,279.3	17,403.8	41,652.1	16,029.1	43,916.5	23,280.2
9	49,162.8	31,990.6	40,288.4	16,075.4	43,858.1	22,731.6
10	47,364.6	29,840.5	39,816.8	15,728.6	43,533.7	21,736.1

Table A.2: Percentage gap to the best known solution values, shown in Table A.1, for Hybrid Genetic Search with Advanced Diversity Control (HGSADC) by Vidal et al. (2013), Hybrid Genetic Search Router (HGSR) by Kool et al. (2022) and Lin-Kernighan-Helsgaun version 3 (LKH-3) by Helsgaun (2017). The lowest gap to the best known solution is indicated in boldface. Maximum runtime is 120 minutes for HGSADC and HGSR, and is not specified for LKH-3.

	HGSADC			HGSR			LKH-3		
	R1	R2	C1	R1	R2	C1	R1	R2	C1
1	1.15	14.74	0.08	**0.56**	**0.03**	**0.00**	0.80	14.39	0.08
2	1.74	7.45	2.33	**0.52**	**0.46**	**0.12**	1.70	7.10	2.20
3	1.25	2.68	0.44	**0.72**	**0.13**	0.38	1.06	2.21	**0.09**
4	0.95	1.28	0.17	**0.44**	**0.33**	0.23	0.68	0.39	**0.09**
5	2.90	6.78	0.08	**0.38**	**0.18**	**0.00**	1.45	6.15	0.08
6	1.98	3.77	0.08	**0.74**	**0.19**	**0.00**	1.50	3.26	3.28
7	1.20	1.52	1.07	**0.40**	0.88	**0.00**	0.97	**0.66**	2.44
8	0.75	1.14	1.36	**0.41**	0.34	**0.60**	0.49	**0.22**	3.31
9	2.70	3.88	0.70	**0.52**	**0.21**	0.34	1.70	3.16	**0.18**
10	2.56	2.54	0.29	**0.69**	**0.14**	0.84	1.62	1.26	**0.11**
	C2	RC1	RC2	C2	RC1	RC2	C2	RC1	RC2
1	0.23	1.05	7.90	**0.00**	**0.56**	**0.25**	0.22	0.76	7.67
2	4.06	1.03	8.58	**0.00**	**0.49**	**0.08**	4.03	0.67	8.58
3	5.30	0.87	2.22	**0.00**	**0.45**	**0.24**	5.29	0.57	2.00
4	1.28	0.62	0.58	**0.03**	0.53	**0.38**	1.28	**0.41**	0.54
5	0.24	1.19	5.58	**0.00**	0.71	**0.25**	0.24	**0.70**	5.21
6	4.06	0.89	4.67	**0.00**	1.02	**0.16**	3.86	**0.46**	3.94
7	10.89	1.10	3.69	**0.00**	**0.54**	**0.25**	9.18	0.61	2.94
8	3.42	1.02	2.18	**0.00**	0.52	**0.12**	3.42	**0.36**	1.84
9	2.22	0.96	1.69	**0.00**	0.52	**0.28**	1.84	**0.48**	1.30
10	1.37	0.83	1.57	**0.00**	**0.23**	**0.58**	1.37	0.55	0.80

Appendix B

Mathematical Model

This appendix contains the complete mathematical model from Chapter 4. It is included to give an overview without the explanations of Chapter 4.

Sets

$\mathcal{N}$	The set of all nodes, depots and customers
$\mathcal{N}_v$	The set of nodes reachable by vehicle v
$\mathcal{N}^D$	The set of depots
$\mathcal{N}^C$	The set of customers
$\mathcal{T}$	The set of time intervals in which loading can occur
$\mathcal{V}$	The set of all vehicles
$\mathcal{V}_i$	The set of vehicles available at depot i

Indices

i	A node, either customer or depot
j	A node, either customer or depot
k	A depot
v	A vehicle
τ	A time interval

Parameters

G_i	Depot opening costs for depot i
G^F	Fleet cost per vehicle
ξ	The stochastic parameter representing customer presence
$\tilde{\xi}$	A realization of the stochastic parameter ξ

$\tilde{F}_j$	1 if customer i has placed an order in $\tilde{\xi}$, 0 otherwise
$\mathcal{K}_{ij}$	1 if depot option i and j can be open simultaneously, 0 otherwise
A_{ij}	The distance between node i and j
B^D	Break duration
B_v^C	The break center coefficient of vehicle v
C_v^T	Costs per unit time for vehicle v
C_v^O	Costs per time unit of overtime for vehicle v
C_v^D	Costs per unit distance for vehicle v
C_v^F	Fixed costs for vehicle v
H_v^V	Capacity for vehicle v
R_v	1 if vehicle v is mandatory to use, 0 otherwise
L_i	The maximum number of vehicles that can load simultaneously at depot i
D_i	The demand of customer i
T_i^S	The time that vehicle v uses to load or unload at node i
T_v^B	Maximum driving duration before vehicle v must take a break
T_v^D	Maximum duration for a shift, including overtime, for vehicle v
T_v^O	Maximum duration for ordinary work time for the driver of vehicle v
T_{ij}	Driving duration between nodes i and j
$\underline{T}_\tau^I$	Start of loading interval
$\overline{T}_\tau^I$	End of loading interval
$\underline{T}_j^N$	Start of service time window for customer j
$\overline{T}_j^N$	End of service time window for customer j
$\underline{T}_v^V$	Start of vehicle availability time window for vehicle v
$\overline{T}_v^V$	End of vehicle availability time window for vehicle v

Variables

γ	The vehicle fleet size
δ_i	1 if depot i is open, 0 otherwise. δ is the vector of all δ_i
x_{ijv}	1 if a vehicle v drives directly from node i to j, 0 otherwise
y_v	1 if a vehicle v is used to drive at all, 0 otherwise
a_v	Time units of overtime for vehicle v
b_{ijv}	1 if vehicle v has a break when driving between nodes i and j, 0 otherwise
$d_{v\tau}$	1 if vehicle v loads in time interval τ, 0 otherwise
$k_{v\tau}$	1 if vehicle v starts loading in time interval τ, 0 otherwise
$e_{v\tau}$	1 if vehicle v ends loading in time interval τ, 0 otherwise

t_{iv}	The time at which vehicle v arrives at node i
t_v^E	The time at which vehicle v ends its shift

First-stage problem

$$\min z = \sum_{i \in \mathcal{N}^D \setminus \{0\}} G_i \delta_i \tag{B.1a}$$
$$+ G^F \gamma \tag{B.1b}$$
$$+ \mathbb{E}_\xi \left[Q(\delta, \gamma, \xi) \right] \tag{B.1c}$$

$$\delta_i + \delta_j \leq 1 + \mathcal{K}_{ij} \qquad i \in \mathcal{N}^D, j \in \mathcal{N}^D \setminus \{i\} \tag{B.2}$$
$$\delta_0 = 1 \tag{B.3}$$
$$\delta_i \in \{0,1\} \qquad i \in \mathcal{N}^D \tag{B.4}$$
$$\gamma \in \mathbb{Z}_{\geq 0} \tag{B.5}$$

Second-stage problem

$$Q(\delta, \gamma, \tilde{\xi}) = \min \sum_{v \in \mathcal{V}} C_v^F y_v \tag{B.6a}$$
$$+ \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N}_v} \sum_{j \in \mathcal{N}_v \setminus \{i\}} C_v^D A_{ij} x_{ijv} \tag{B.6b}$$
$$+ \sum_{i \in \mathcal{N}^D} \sum_{v \in \mathcal{V}_i} C_v^T (t_v^E - t_{iv}) \tag{B.6c}$$
$$+ \sum_{v \in \mathcal{V}} C_v^O a_v \tag{B.6d}$$

$$\sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N}_v \setminus \{j\}} x_{ijv} = \tilde{F}_j \qquad j \in \mathcal{N}^C \tag{B.7}$$
$$\sum_{j \in \mathcal{N}^C} x_{ijv} - \delta_i y_v = 0 \qquad i \in \mathcal{N}^D, v \in \mathcal{V}_i \tag{B.8}$$
$$\sum_{j \in \mathcal{N}_v \setminus \{i\}} x_{jiv} - \sum_{j \in \mathcal{N}_v \setminus \{i\}} x_{ijv} = 0 \qquad v \in \mathcal{V},\ i \in \mathcal{N}_v \tag{B.9}$$

$$\sum_{i \in \mathcal{N}^C} \sum_{j \in \mathcal{N}_v \setminus \{i\}} D_i x_{ijv} \leq H_v^V \qquad v \in \mathcal{V} \tag{B.10}$$
$$\sum_{v \in \mathcal{V}} y_v \leq \gamma \tag{B.11}$$
$$y_v \geq R_v \delta_i \qquad i \in \mathcal{N}^D, v \in \mathcal{V}_i \tag{B.12}$$

$$(t_v^E - t_{kv})(1 - \sum_{i\in\mathcal{N}_v}\sum_{j\in\mathcal{N}_v\setminus\{i\}} b_{ijv}) \le T_v^B \qquad k\in\mathcal{N}^D,\ v\in\mathcal{V}_k \tag{B.13}$$

$$b_{ijv} - x_{ijv} \le 0 \qquad v\in\mathcal{V}, i\in\mathcal{N}_v, j\in\mathcal{N}_v\setminus\{i\} \tag{B.14}$$

$$\frac{1-B_v^C}{2}(t_v^E - t_{kv})b_{ijv} \le t_{iv} \qquad k\in\mathcal{N}^D, v\in\mathcal{V}_k, i\in\mathcal{N}_v, j\in\mathcal{N}_v\setminus\{i\} \tag{B.15}$$

$$\frac{1+B_v^C}{2}(t_v^E - t_{kv})b_{ijv} \ge t_{jv} \qquad k\in\mathcal{N}^D, v\in\mathcal{V}_k, i\in\mathcal{N}_v, j\in\mathcal{N}_v\setminus\{i\} \tag{B.16}$$

$$(t_{iv} + T_i^S + B^D b_{ijv} + T_{ij} - t_{jv})x_{ijv} \le 0 \qquad v\in\mathcal{V}, i\in\mathcal{N}_v, j\in\mathcal{N}^C\setminus\{i\} \tag{B.17}$$

$$(t_{iv} + T_i^S + B^D b_{ijv} + T_{ij} - t_v^E)x_{ijv} \le 0 \qquad i\in\mathcal{N}^C, j\in\mathcal{N}^D\setminus\{i\}, v\in\mathcal{V}_j \tag{B.18}$$

$$\underline{T}_j^N \le t_{jv} \le \overline{T}_j^N \qquad i\in\mathcal{N}^D, v\in\mathcal{V}_i,\ j\in\mathcal{N}_v\setminus\{i\} \tag{B.19}$$

$$t_{iv} \ge \underline{T}_v^V \qquad i\in\mathcal{N}^D, v\in\mathcal{V}_i \tag{B.20}$$

$$t_v^E \le \overline{T}_v^V \qquad v\in\mathcal{V} \tag{B.21}$$

$$t_v^E - t_{iv} \le T_v^D \qquad i\in\mathcal{N}^D,\ v\in\mathcal{V}_i \tag{B.22}$$

$$t_v^E - t_{iv} - a_v \le T_v^O \qquad i\in\mathcal{N}^D,\ v\in\mathcal{V}_i \tag{B.23}$$

$$\sum_{\tau\in\mathcal{T}} \underline{T}_\tau^I k_{v\tau} - t_{iv} \le 0 \qquad i\in\mathcal{N}^D,\ v\in\mathcal{V}_i \tag{B.24}$$

$$(t_{iv} + T_{iv}^S - \sum_{\tau\in\mathcal{T}} \overline{T}_\tau^I e_{v\tau})y_v \le 0 \qquad i\in\mathcal{N}^D,\ v\in\mathcal{V}_i \tag{B.25}$$

$$\sum_{\tau\in\mathcal{T}} k_{v\tau} - y_v = 0 \qquad v\in\mathcal{V} \tag{B.26}$$

$$\sum_{\tau\in\mathcal{T}} e_{v\tau} - y_v = 0 \qquad v\in\mathcal{V} \tag{B.27}$$

$$k_{v\tau} + d_{v\tau} - d_{v(\tau+1)} - e_{v(\tau+1)} = 0 \qquad v\in\mathcal{V},\ \tau\in\mathcal{T}\setminus\{|\mathcal{T}|\} \tag{B.28}$$

$$\sum_{v\in\mathcal{V}_i}(k_{v\tau} + d_{v\tau} + e_{v\tau}) \le L_i \qquad i\in\mathcal{N}^D,\ \tau\in\mathcal{T} \tag{B.29}$$

$$x_{ijv} \in \{0,1\} \qquad v\in\mathcal{V},\ i\in\mathcal{N}_v,\ j\in\mathcal{N}_v\setminus\{i\} \tag{B.30}$$

$$y_v \in \{0,1\} \qquad v\in\mathcal{V} \tag{B.31}$$

$$b_{ijv} \in \{0,1\} \qquad v\in\mathcal{V},\ i\in\mathcal{N}_v,\ j\in\mathcal{N}_v\setminus\{i\} \tag{B.32}$$

$$t_{iv} \ge 0 \qquad v\in\mathcal{V},\ i\in\mathcal{N}_v \tag{B.33}$$

$$a_v \ge 0 \qquad v\in\mathcal{V} \tag{B.34}$$

$$k_{v\tau} \in \{0,1\} \qquad v\in\mathcal{V},\ \tau\in\mathcal{T} \tag{B.35}$$

$$e_{v\tau} \in \{0,1\} \qquad v\in\mathcal{V},\ \tau\in\mathcal{T} \tag{B.36}$$

$$d_{v\tau} \in \{0,1\} \qquad v\in\mathcal{V},\ \tau\in\mathcal{T} \tag{B.37}$$

Appendix C

Details of the Problem Instances

The attributes for the depots used in the case study and the other experiments are presented in Table C.1 and Table C.2, respectively. Table C.3 shows the routing problems used to compare the routing algorithms and adjust the routing costs.

Table C.1: Depots in problem instances O2, O3, O4 and O5 in the case study for Oda. The Warehouse driving time is the round trip driving time that the line haul trucks must drive to restock depots. The costs are per shift.

Name	Transport				Cost	
	Fixed-cost vehicles	Time-cost vehicles	Line haul trucks	Warehouse driving time	Line haul cost	Leasing cost
D6-6	6	0	1	3787	842	818
D6-12	12	0	2	3787	1683	1103
D6-18	18	0	3	3787	2525	1388
D6-24	24	0	4	3787	3366	1673
D10-6	3	3	1	6182	1374	952
D10-12	6	6	2	6182	2748	1237
D10-18	9	9	3	6182	4121	1522
D10-24	12	12	4	6182	5495	1807
D12-6	6	0	1	3224	716	1085
D12-12	12	0	2	3224	1433	1370
D12-18	18	0	3	3224	2149	1655
D12-24	24	0	4	3224	2866	1940
D17-6	3	3	1	4501	1000	818
D17-12	6	6	2	4501	2000	1103
D17-18	9	9	3	4501	3001	1388
D17-24	12	12	4	4501	4001	1673

Table C.2: Depots in problem instances A, B and C. The Warehouse driving time is the round trip driving time that the line haul trucks must drive to restock depots. The costs is per shift.

Name	Problem instances	Transport				Cost	
		Fixed-cost vehicles	Time-cost vehicles	Line haul trucks	Warehouse driving time	Line haul cost	Leasing cost
Warehouse	A,B,C	160	800	0	0	0	0
D1	A	6	6	2	4315	1918	1237
D2	A	6	6	2	6986	3105	1103
D3	A	6	6	2	3855	1713	1170
D4	A,B,C	6	0	1	4644	1032	685
D4-Alternative	B	12	0	2	4644	2064	970
D5	A,B,C	6	6	2	5792	2574	1237
D5-Alternative	B	3	3	1	5792	1287	952
D6	A	6	0	1	3787	842	818
D7	A	6	0	1	4746	1055	885
D8	A	6	0	1	4751	1056	818
D9	A	6	6	2	4162	1850	1303
D10	A,B,C	6	6	2	6182	2748	1237
D10-Alternative	B	9	9	3	6182	4121	1522
D11	A,B,C	6	0	1	5429	1206	1218
D11-Alternative	B	6	6	2	5429	2413	1503
D12	A,B,C	6	0	1	3224	716	1085
D12-Alternative	B	12	0	2	3224	1433	1370
D13	A,B,C	6	0	1	2961	658	952
D13-Alternative	B	12	0	2	2961	1316	1237
D14	A,B,C	6	0	1	3723	827	1285
D14-Alternative	B	6	6	2	3723	1655	1570
D15	A	6	0	1	5079	1129	818
D16	A,B,C	6	0	1	5531	1229	885
D16-Alternative	B	12	0	2	5531	2458	1170
D17	A,B,C	9	9	3	4501	3001	1388
D17-Alternative	B	6	6	2	4501	2000	1103
D18	A,B,C	6	6	2	2428	1079	1170
D18-Alternative	B	3	3	1	2428	540	885
D19	A	6	6	2	4272	1899	970
D20	A	6	6	2	4571	2032	1103

Table C.3: Overview of the routing problems used to calibrate the routing costs and when evaluating the routing algorithm, with depot configurations, number of open depots (D), number of present customers (C) and cost computed by Oda's solver Navegante with 30 minutes runtime.

Name	Depot configuration	D	C	Cost
T1	Warehouse, D14	1	1027	97,241
T2	Warehouse, D14	1	2639	229,124
T3	Warehouse	0	2991	259,103
T4	Warehouse, D18, C, D7, D20, D13, D11, D9, D5, D16, D4	10	2991	262,471
T5	Warehouse, D18, D17, D11, D9, D5	5	2991	242,754
T6	Warehouse, D14	1	2991	257,844
T7	Warehouse, D17, D20, D9, D3	5	2991	244,163
T8	Warehouse, D9, D15	2	2991	253,680
T9	Warehouse, D14	1	3090	264,909
T10	Warehouse	0	950	89,479
T11	Warehouse, D18, C, D7, D20, D13, D11, D9, D5, D16, D4	10	950	237,821
T12	Warehouse, D18, D17, D11, D9, D5	5	950	160,413
T13	Warehouse, D17, D20, D9, D3	4	950	143,472
T14	Warehouse, D9, D15	2	950	100,679
T15	Warehouse, D14	1	1111	104,080
T16	Warehouse, D14	1	1429	130,938
T17	Warehouse, D14	1	2003	177,337
T18	Warehouse	0	2076	187,454
T19	Warehouse, D18, C, D7, D20, D13, D11, D9, D5, D16, D4	10	2076	241,410
T20	Warehouse, D18, D17, D11, D9, D5	5	2076	185,149
T21	Warehouse, D17, D20, D9, D3	4	2076	182,355
T22	Warehouse, D9, D15	1	2076	182,060
T23	Warehouse, D14	1	2403	211,169

Appendix D

Details of the Stability Tests

A selection of depot configurations are used in the stability testing. Table D.1 describes the 5 solutions used in the out-of-sample stability testing, and Table D.2 describes the 10 depot configurations used for testing the ranking stability.

Table D.1: Overview of the 5 routing problems used for out-of-sample stability testing with the depot configurations and the number of open depots in the problem.

Solution no	Depot configuration	Open depots
1	Warehouse, D18, D14, D17, D9, D5	5
2	Warehouse, D9, D5	2
3	Warehouse, D18, D14, D1	3
4	Warehouse, D18	1
5	Warehouse, D18, D20, D9, D5	4

Table D.2: Overview of the 10 routing problems used for evaluating ranking in the stability testing with the depot configurations and number of open depots in the problem.

Depot configuration	Open depots
Warehouse	0
Warehouse, D18	1
Warehouse, D17	1
Warehouse, D20	1
Warehouse, D17, D9	2
Warehouse, D5, D16	2
Warehouse, D17, D9, D5	3
Warehouse, D20, D10, D9	3
Warehouse, D18, D10, D9	3
Warehouse, D18, D10, D16	3