universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

A multi-depot multi-compartment vehicle routing problem, solved by a hybrid variable neighborhood search

verfasst von | submitted by

David Binder BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 915
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft
Betreut von | Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner Privatdoz.
Mitbetreut von | Co-Supervisor:	Emilio Jose Alarcon Ortega MA PhD

Contents

List of Figures

List of Algorithms

List of Tables

1 Abstract

This master thesis aims to present a hybrid algorithm to solve a vehicle routing problem with multiple depots and multiple compartments(MDMCVRP). To solve this problem, two heuristics, the greedy randomized adaptive search procedure(GRASP) and the variable neighborhood search(VNS) are combined, to create tours and improve the obtained solutions. For multiple compartments, split delivery is considered and allowed, in a way that single products can be delivered by different vehicles, but single products are not allowed to be split.

Diese Masterarbeit präsentiert einen hybriden Algorithmus, um das Vehicle Routing Problem mit mehreren Depots und mehreren Produkten (MDMCVRP) zu lösen. Dafür wurden zwei Heuristiken kombiniert. Zum einen, die greedy randomized adaptive search procedure(GRASP), welche benutzt wird um Touren zu generieren und zu optimieren. Zum anderen, die variable neighborhood search(VNS), welche die Ergebnisse des GRASP weiter optimiert. Für die Erweiterung der mehreren Produkte sind mehrere Lieferungen erlaubt, aber nur unter der Bedingung, dass ein einziges Produkt zur Gänze von einem LKW geliefert werden muss.

2 Motivation

The first initial ideas for tracking this specific problem in a Master thesis came during the first COVID lockdown in Austria in early 2020. As people were relying more and more on food delivery services, Alfies or Gurkerl, for example, the idea grew to cover this topic.

For the reader who is not familiar with those companies and their business models, they specialize in having an online supermarket where you can do your groceries and get delivered in a rather short time, unlike Lieferando, Wolt or Foodora where you primarily order finished meals from third parties. In Austria, those two companies were among the first to establish a business model like that, and other, bigger supermarkets like Spar or Billa, created their online supermarkets way later.

In the example of Alfies, it is an online supermarket that delivers in an hour with a great variety of products ranging from beers to toilet articles to frozen goods. In this context, it makes sense that these companies use vehicles that have multiple compartments for all these different products.

With the addition of multiple depots to the vehicle routing problem with multiple compartments(MCVRP), it becomes more complex but adds up to a problem that is closer to real-life situations.

3 Introduction

The vehicle routing problem(VRP) is a very old problem, thoroughly studied in the field of operational research. It is classified as an NP-Hard Problem and was first introduced by Danzig and Ramser[DR59] in 1959 in their paper "the truck dispatching problem" as the capacitated vehicle routing problem (CVRP).

Efficient vehicle routing is a critical component of logistics and supply chain management, directly impacting operational costs, service quality, and environmental sustainability. The VRP and its variants serve as foundational models for optimizing the delivery and distribution of goods across diverse industries. By minimizing travel distances, fleet size, or delivery times, these models help organizations reduce fuel consumption, improve customer satisfaction, and improve overall logistical efficiency.

As global supply chains grow increasingly complex, solving VRPs effectively becomes essential for maintaining competitiveness and achieving sustainable logistics operations.

In 1964, Clarke and Wright[CW64] introduced the first effective greedy algorithm to solve the VRP. A greedy algorithm is a heuristic that tries to get good results for an NP-Hard problem in a reasonable time. Heuristics, in general, do not try to get the best solutions but rather try to get close to the best solutions.

Since being first introduced in 1959, it has been tried to make the VRP more of a real-life problem than only an abstract one. To achieve this, multiple extensions have been introduced in the last 65 years.

Laporte et al.[LNA84] introduced the extension of multiple depots(MDVRP) in 1984, by presenting formal procedures to find optimal solutions. With multiple depots, the complexity of the problem increases due to the consideration of which customer needs to be served by which depot.

The extension considering vehicles with multiple compartments(MCVRP) has become more studied in the last 15 years, although it was already introduced by Brown et al.[BG81] in 1981. It was not referred to as MCVRP at the time, the core features of the MCVRP are present: multiple products, compartment-specific loading, and complex routing constraints.

This extension and the MDVRP are covered in detail in the next chapter.

Cordeau et al. [CGL97] presented a paper in 1997 in which they introduced a tabu search heuristic to solve the MDVRP, the periodic vehicle routing problem(PVRP) and the periodic traveling salesman problem(PTSP).

Another notable extension is the extension dealing with time windows(VRPTW). VRPTW was first formally introduced by Solomon in 1987 [Sol87], who also proposed benchmark instances that have since become a standard for evaluating solution algorithms. The addition of time constraints significantly increases the complexity of the problem, as it restricts route flexibility and often leads to infeasible solutions if not properly managed. The problem is particularly relevant in logistics sectors such as parcel delivery, urban transportation, and the distribution of perishable goods, where customer availability and service punctuality are critical.

The Vehicle Routing Problem with Pickup and Delivery (VRPPD) extends the classical VRP by requiring the transport of goods or passengers from specific pickup locations to designated delivery locations. This extension was already studied through the 1980 s but was first introduced with a model by Min in 1989 [Min89]. Each request consists of a pair of nodes, pickup and delivery, that must be visited in a specific order by the same vehicle. The challenge lies in optimizing routes to minimize cost or distance while satisfying vehicle capacity, time windows, and precedence constraints.

This variant is highly relevant in real-world logistics, such as courier services, ride-sharing platforms, and healthcare transportation. Due to its complexity, exact methods struggle with large-scale instances, prompting the widespread use of heuristic and metaheuristic approaches such as tabu search, genetic algorithms, and GRASP. In VRPPD, maintaining route feasibility while sequencing tightly coupled pickup-delivery pairs often leads to highly constrained solution spaces. Recent research has focused on dynamic and stochastic versions of VRPPD to address real-time customer requests and traffic variability.

The Split Delivery Vehicle Routing Problem (SDVRP) allows a single customer's demand to be split across multiple deliveries, potentially by different vehicles. This contrasts with the classical VRP, where each customer must be visited exactly once. The main advantage of permitting split deliveries is the potential to reduce total transportation cost or the number of vehicles used, especially when dealing with large or uneven customer demands.

Originally introduced by Dror and Trudeau in 1989[DT89], SDVRP challenges traditional routing assumptions and introduces new trade-offs in solution design, minimizing cost while managing increased route complexity and coordination overhead. Although split deliveries offer logistical flexibility, they may not be desirable in all contexts due to operational concerns such as customer service expectations or increased handling. Metaheuristic methods, including GRASP, simulated annealing, and hybrid algorithms, are often used to effectively solve the SDVRP. The problem is particularly relevant in bulk material delivery, humanitarian logistics, and situations where supply constraints require partial fulfillment.

The MDMCVRP is highly relevant in various real-world logistics scenarios where goods must be transported from multiple depots and kept separated due to compatibility, contamination risk, or product-specific handling requirements.

In addition to supermarkets and groceries, used examples are in fuel distribution, waste collection and recycling, pharmaceutics, dairy product distribution, military logistics, and post-disaster logistics.

After this Instruction, the Thesis is structured into the following chapters.

In Chapter 4, there will be a literature review, where some of the more recent literature on the two extensions, multi-depots and multi-compartments are reviewed. In addition, a short review on the literature concerning the GRASP and the VNS is given.

Chapter 5 is the description of the problem, where the problem assumptions are explained in detail and the mathematical formulation is presented.

In Chapter 6 the Solution Method is presented. The algorithm is explained in detail. What are the different parts of the algorithm and how they work.

Chapter 7 discusses the experiment that was conducted. Some of the results are discussed in detail and compared with each other.

The last chapter 8 reviews the thesis and gives a short insight into what future problem assumptions might be.

4 Literature review

In the following subsections, some of the literature on the two extensions multiple-depots and multiple-compartments is reviewed with a clear focus on newer research areas. In addition, a short literature review on the parts of the proposed algorithm, the greedy randomized adaptive search procedure(GRASP), and the variable neighborhood search(VNS).

4.1 Multiple-Depots

In an MD setup, customers are served by more than one depot. This reflects real-world logistics networks(for example, companies with regional warehouses) and aims to reduce transportation costs and delivery times.

The decision on how customers need to be served can be either static (the customer is assigned to a fixed depot based on a specific rule) or dynamic (the customer is served by a depot that fits best in the overall optimization).

Another decision that must be made beforehand is regarding the fleet of vehicles. Vehicles can be homogeneous(every vehicle has the same capacity and restrictions) or heterogeneous(different types of vehicles are available). This in particular is not a decision concerning only MDVRPs, but if vehicles are heterogeneous, the decision can be made that certain types of vehicles are only available in certain depots or are restricted in other ways.

Although the extension with multiple depots(MD) was first introduced in 1984 by Laporte et al.[LNA84] it has become more studied in recent years. One of the main reasons for this is the ability to make VRPs more like real-life problems. The second reason is the increasing computational power of PCs that makes them more capable of solving more complex problems.

In 2015 Montoya-Torres et al.[Mon+15] conducted a survey on MDVRPs. They considered papers published between 1984 and 2014 in which different variants of the problem were considered, such as time windows, split delivery, heterogeneous fleet, periodic deliveries, and pickup and delivery. Their review also classifies the various approaches for optimizing single or multiple objectives.

The survey gives a good idea on how the number of scientific papers considering some variation of MDVRPs evolved over time. While in the early years, from 1984 to 1999 only 18 papers considering the MD were published, from 2000 until 2014 this number increased to 129. This shows the increasing interest and relevance of the MDVRP and its variations for logistic problems.

Calamoneri et al.[CCM24] studied the multi-depot multi-trip VRP with total completion time minimization(MDMT-VRP-TCT) in 2024. In this problem setting, they focused on managing a post- disaster emergency scenario, where the use of a fleet of unmanned aerial vehicles(UAVs) helps the rescue teams to identify people needing help inside an affected area. They base their problem assumption on having multiple depots on having several different positions where these UAVs can start to cover more areas more effectively.

Another study that uses a variation of the MDVRP was done by Su et al.[SZZ24], where they constructed a multi-depot green vehicle routing problem with time windows con-

sidering customer satisfaction(MDGVRPTW-CS). To optimize this problem, they proposed a lightweight genetic algorithm with variable neighborhood search(LGAVNS). Their experimental results show that their approach shows advantages in terms of effectiveness and convergence speed compared to six state-of-the-art algorithms, especially when addressing MDGVRPTW-CS in large-scale scenarios.

Liu et al.[Liu+24] constructed an adaptive large neighborhood water wave optimization algorithm(ALNSWWO) to solve a multi-depot capacitated vehicle routing problem with time windows(MDCVRPTW). In their research, they show that the combination of variable neighborhood search(VNS), large neighborhood search(LNS), and adaptive large neighborhood search(ALNS) significantly outperforms their algorithms without one of those operators in terms of solution accuracy and stability.

The Multi-Depot Vehicle Routing Problem (MDVRP) represents a crucial extension of the classic VRP, reflecting the distributed nature of real-world logistics networks. By allowing multiple depots to serve a shared customer base, MDVRP increases flexibility, reduces total transportation costs, and enables better utilization of resources. However, this increased flexibility comes with added computational complexity, particularly when combined with other constraints such as compartmentalized loading, time windows, or heterogeneous fleets.

The integration of multiple depots not only improves logistical efficiency, but also provides a foundation for more resilient and sustainable supply chains. As routing problems grow in scale and complexity, MDVRP offers a robust framework for modeling decentralized, multi-origin delivery systems, making it a vital area for continued research and practical application, especially when hybridized with advanced metaheuristics.

4.2 Multi-Compartment

These problems involve vehicles equipped with multiple compartments, which allows for the simultaneous transportation of goods with different characteristics that would otherwise require separate vehicles.

As already stated before, the extension considering vehicles with multiple compartments(MC) has become more studied in the last 15 years, although it was already introduced by Brown et al.[BG81] in 1981. It was not referred to as MCVRP at the time, the core features of the MCVRP are present: multiple products, compartment-specific loading, and complex routing constraints.

They developed a real-time dispatch system designed to improve the operational efficiency of petroleum tank truck logistics. The system, implemented by Chevron U.S.A., manages complex routing and scheduling tasks across a large distribution network involving more than 80 terminals and 300 vehicles. Using a heuristic-based optimization model, the system supports dispatchers in effectively assigning loads, accounting for vehicle capacities, product compatibility, and cost minimization. In particular, the approach led to measurable reductions in transportation costs and allowed greater workload balancing across the fleet.

Ostermeier et al. [Ost+21] provide a comprehensive review of multi-compartment vehi-

cle routing problems (MCVRPs) up to 2021. The authors introduce a unified typology to categorize the various variations of the MCVRP, facilitating the identification of common attributes across different applications. They also propose a general mathematical model that encompasses these variants, offering a structured approach to problem formulation. Through an extensive literature review, the paper highlights existing research gaps and suggests avenues for future investigation, emphasizing the need for further exploration in this evolving field.

The MCVRP has gained increasing attention in recent years due to its relevance in complex logistics operations such as waste management, multi-temperature distribution, and pharmaceutical delivery. Recent literature highlights a range of innovative approaches designed to address both classical and emerging challenges within MCVRP frameworks.

Gonçalves-Dosantos et al. (2024) [GDC24] propose a two-stage heuristic algorithm tailored for MCVRP with stochastic demands, particularly within agricultural logistics. Their approach combines a constructive heuristic with an iterated tabu search to efficiently handle demand uncertainty, demonstrating promising results on real-world data.

A significant shift towards sustainable logistics is seen in a bi-objective optimization framework introduced in mid-2024 by Ouertana and Sidi Moh [ON24]. This study formulates a Mixed-Integer Linear Programming (MILP) model that minimizes both costs and carbon emissions, solved via a modified Iterated Local Search algorithm. The findings underline the growing importance of integrating environmental objectives into vehicle routing strategies.

In the context of urban logistics, Mohammadi et al. (2023) [Moh+23] develop a dynamic routing model for MCVRP in waste collection. Their work incorporates demand variability and real-time traffic conditions, utilizing a hybrid Genetic and Particle Swarm Optimization algorithm. The dynamic model enhances route flexibility and sustainability in smart cities.

Another recent study by Fan et al. [FYY23] extends the MCVRP by introducing traffic congestion effects and a mixed carbon policy, demonstrating how regulatory and environmental factors can be embedded into routing decisions. This reflects an emerging trend towards policy-aware routing models that align with governmental sustainability goals.

Finally, Chamurally and Rieck (2024) [CR24] present a machine learning-based approach to solve the MCVRP under uncertainty of demand. Their model uses predictive analytics to inform routing decisions, showcasing the potential of data-driven techniques in improving robustness and adaptability in real-world operations.

Together, these studies represent a diverse set of methodological advancements, ranging from heuristics and metaheuristics to predictive modeling and bi-objective optimization. They signal a clear evolution in the field, emphasizing realism, sustainability, and adaptability as core research directions for future MCVRP developments.

4.3 Variable Neighborhood Search

The variable neighborhood search(VNS) is a metaheuristic to solve combinatorial and global optimization problems. The central idea is to systematically change the neighborhood structures in order to escape from local optima and find better solutions.

It was first introduced by Hansen and Mladenović[MH97] in 1997. The algorithm was developed to improve the quality of the solution by exploring increasingly distant neighborhoods of a current solution. The big advantages of the VNS are that it is simple to implement, it escapes local optima effectively, and its adaptability to many types of optimization problems.

Since its introduction, several variations of the VNS have been implemented. Hansen et al. [Han+17]provided an in-depth exploration of the VNS in 2017, detailing its basic principles, variants, and applications in various optimization problems in their paper.

In recent years, VNS has been adapted to a variety of new domains, reflecting both methodological advances and the integration of data-driven insights.

For example, Saarinen et al. (2023) [SCK23]proposed an opportunistic VNS tailored to detect heavy subgraphs in social networks, dynamically adjusting neighborhood exploration based on partial success.

In logistics, Kant et al. (2024)[Kan+24] designed a VNS for the multi-depot multiple set orienteering problem (MDMSOP), incorporating profit-driven shaking and hybrid local search.

Finally, Cai et al. (2024)[CKD25] introduced Balans, an adaptive VNS-based framework for solving general Mixed Integer Programming (MIP) problems, leveraging online learning for intelligent neighborhood selection.

These studies highlight the flexibility and continual evolution of VNS for solving modern, large-scale, and domain-specific optimization challenges.

4.4 Greedy Randomized Adaptive Search Procedure

The greedy randomized adaptive search procedure (GRASP) was first introduced by Feo and Resende in 1995 [FR95] and is a metaheuristic algorithm. It was initially applied to set covering problems but has been extended to a wide range of combinatorial optimization problems.

In recent years, GRASP has been widely applied to solve complex variants of the VRP, demonstrating both flexibility and effectiveness.

A notable study by Machado et al. in 2021 [Mac+21] proposed a hybrid approach combining GRASP with Variable Neighborhood Search (VNS) and mathematical programming techniques such as set-covering and set-partitioning. This method, applied to the Capacitated VRP, produced high-quality solutions and emphasized the strength of integrating metaheuristics with exact formulations.

Similarly, Souza et al.[Sou+22]developed a reactive GRASP algorithm for the MDVRP, incorporating adaptive mechanisms and local search (VND) to improve the diversity of solutions and avoid premature convergence. Their findings highlighted GRASP's ability to adaptively balance exploration and exploitation in more complex routing contexts.

More recently, Yu Wu and Xiaoping Qiu [WQ24] addressed dynamic and stochastic routing challenges in home pickup services, using GRASP for initial request allocation alongside

a Markov Decision Process (MDP) for real-time vehicle routing. This hybrid method accounted for service continuity and stochastic demand, showcasing GRASP's relevance in real-world, time-sensitive logistics.

Collectively, these studies confirm GRASP's utility in both static and dynamic VRP scenarios and support its further application in optimization problems that require robust and scalable solutions.

5 Problem description

The goal of this master thesis is to combine two known heuristics, the variable neighborhood search (VNS) and the greedy randomized adaptive search procedure (GRASP), to solve a multi-depot, multi-compartment vehicle routing problem.

The main objective of this problem is to minimize the total traveled distances of all tours. The additional setting of multiple depots means that each tour has multiple options to start and end. To cover more possible solutions, after the initialization of the tours with GRASP, the tours are not restricted to end at their starting depots. The number of vehicles in each depot is restricted, but the depots have no restriction on how many vehicles finish in them. Concerning the capacity of the different products for the depots, no restrictions are considered.

The capacity of each vehicle is limited. Due to the multi-compartment setting, each vehicle has individual capacities for each compartment. In the problem setting, split deliveries are allowed but are restricted to one split. Splits of a single product are prohibited. This means that the demand of one customer for a certain product needs to be satisfied by one vehicle and cannot be split up. If there are more than two compartments, only one split is allowed, which means that all products must be delivered by two vehicles at most. All customer demands are predetermined and deterministic.

The travel distance of the vehicles is restricted for most cases; exceptions are mentioned as such, this will be done in Chapter 7. The fleet of vehicles is homogeneous, which means that the capacities and restrictions of every truck and its compartments are the same.

5.1 Problem assumptions

- Each depot has a set number of vehicles.
- If a vehicle departs from a depot, it needs to finish its tour in a depot.
- Vehicles are not restricted to finish their tours at their starting depot.
- The vehicles have a limit for their distance traveled, except if specifically stated otherwise.
- Every vehicle has multiple compartments.
- Each compartment is dedicated to one type of product.
- The demands of every customer must be met fully.
- Split deliveries are allowed, but only for different products and not for single products.
- Split deliveries are restricted to one split at most.
- Demands are predetermined and deterministic.
- The fleet of vehicles is homogeneous.

5.2 Sets, parameters and variables

5.2.1 Sets

N_{tot}: the set of all nodes, customers and depots, denoted by the indices i and j

N_D: the set of depots

N_V: the set of customers

G: the set of products; denoted by index g

K: the set of vehicles; denoted by index k

5.2.2 Parameters

$D_{i,g}$: demand of node i for product g

$C_{i,j}$: distance between nodes i and j

Q_g: the capacity of vehicle compartment dedicated to product g

MV_i: the maximum number of vehicles departing from depot i

MD: the maximum distance, each vehicle is allowed to travel

M: a very large number

5.2.3 Decision variables

$x_{i,j}^k$: equals 1 if the route between nodes i and j is traveled by vehicle k and is zero otherwise

$y_{i,g}^k$: equals 1 if the demand of node i for product g is delivered by vehicle k, and is zero otherwise

z_i^k: equals 1 if node i is visited by vehicle k and is zero otherwise

u_k: equals 1 if the vehicle k is used , and is zero otherwise

$ST_{i,k}$: variable used for elimination of sub-tours

5.3 Mathematical Formulation

The mathematical formulation is derived from Alinaghian and Shokouhi[AS18] and modified for the problem solved in this thesis. The main differences are in the objective function and the allowance for vehicles to be able to end their tours at a different depot.

The objective function differs, because in this thesis the number of used vehicles is not considered as the main objective, but only the maximum traveled distance.

$$min \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} C_{i,j} x_{i,j}^k \tag{1}$$

s.t:

$$\sum_{i \in N_D} \sum_{j \in N_V} x_{i,j}^k \leq u_k, \qquad \forall k \in K \tag{2}$$

$$\sum_{k \in K} y_{i,g}^k = 1, \qquad \forall i \in N_V, g \in G \tag{3}$$

$$\sum_{k \in K} z_i^k \leq 2, \qquad \forall i \in N_V, \tag{4}$$

$$\sum_{g \in G} y_{i,g}^k \leq M \sum_{j \in N_{tot}} x_{j,i}^k, \qquad \forall i \in N_V, k \in K \tag{5}$$

$$\sum_{j \in N_{tot}} x_{j,i}^k \leq \sum_{g \in G} y_{i,g}^k, \qquad \forall i \in N_V, k \in K \tag{6}$$

$$\sum_{i \in N_{tot}} x_{i,j}^k \leq 1 \qquad \forall j \in N_V, k \in K \tag{7}$$

$$\sum_{i \in N_{tot}} x_{i,j}^k \leq \sum_{i \in N_{tot}} x_{j,i}^k \qquad \forall j \in N_{tot}, k \in K \tag{8}$$

$$\sum_{i \in N_{tot}} y_{i,g}^k * D_{i,g} \leq Q_g \qquad \forall g \in G, k \in K \tag{9}$$

$$\sum_{i \in N_{tot}} \sum_{j \in N_{tot}} C_{i,j} * x_{i,j}^k \leq MD, \qquad \forall k \in K \tag{10}$$

$$\sum_{k \in K} \sum_{j \in N_V} x_{i,j}^k \leq MV_i, \qquad \forall i \in N_D \tag{11}$$

$$\sum_{i \in N_D} \sum_{k \in K} ST_{i,k} = 0, \tag{12}$$

$$ST_{i,k} + 1 \leq ST_{j,k} + M(1 - x_{i,j}^k), \qquad \forall i \in N_{tot}, j \in N_V, k \in K \tag{13}$$

$$ST_{i,k} \geq 0, \qquad \forall i \in N_{tot}, k \in K, \tag{14}$$

$$x_{i,j}^k, y_{i,g}^k, z_i^k, u_k \in \{0, 1\} \qquad \forall i, j \in N_{tot}, k \in K, g \in G \tag{15}$$

The first equation (1) describes the objective function of the problem. Its function is to minimize the total distance traveled by all tours.

The first constraint (2) states that if a vehicle is used, the route of that vehicle has to start from a depot.

Constraint (3) denotes that the demand, of a specific product, from a customer, needs to be delivered by a single vehicle. As mentioned above, a single product cannot be split.

The constraint(4) makes sure, that the maximum number of vehicles that can serve a single customer is 2.

The constraints (5) and (6) ensure that the demand of the customer i for all products g can only be fulfilled by a vehicle k, if the vehicle k visits that customer i.

The next constraint (7) makes sure that every customer can only be visited at most once by each vehicle.

Constraint (8) specifies that if a vehicle visits a node, the vehicle must leave that node again if it is not a depot.

The restriction (9) limits the maximum capacities for each different compartment in the vehicles.

Constraint (10) limits the maximum distance traveled for each tour.

The next constraint (11) limits the capacities for each depot.

The constraints (12) and (13) are for the elimination of sub-tours.

The last two constraints (14) and (15) define the domains of the decision variables.

6 Solution Method

In this part of the master thesis, it is described how the proposed algorithm works. As stated above, this thesis proposes an algorithm that combines a greedy randomized adaptive search procedure (GRASP) with a variable neighborhood search (VNS).

First, the construction of the solution with GRASP will be described and explained in detail. After that, the VNS and its different neighborhood structures will be described. At the end of this chapter, the combined algorithm (GRASP-VNS Hybrid) will be presented.

6.1 Construction of an Initial Solution

To construct an initial solution, a GRASP algorithm was implemented. In Chapter 7 it is explained what different variants and adjustments of the GRASP are used. Here, the algorithm is explained in detail.

As shown in Algorithm 1, it first creates empty tours for every available vehicle, then takes the first customer randomly and inserts it into a random tour. For this customer, the distances to all other customers are compared and the nearest X customers are selected.

Algorithm 1 Greedy Randomized Adaptive Search Procedure (GRASP)

```
1:  Initialize m empty routes
2:  Select random starting customer, insert in random tour
3:  while not all customers are visited do
4:      create candidate list (CL) with the nearest X customers
5:      select random customer of CL, add to current tour
6:      if restrictions hold then
7:          add customer to current tour
8:      else
9:          close current tour and insert random customer in new tour
10:     end if
11: end while
12: perform local search on every tour
```

The X in this case can have different values. For the experimental parts, the values 1, 2 or 3 are used. To obtain the computational results, all experiments were performed with these three values. This will be described in more detail in the experimental results.

In the case of X being 1, the creation of the tours is like using the Nearest-Neighbor Algorithm. The Nearest-Neighbor Algorithm for VRP builds routes by always selecting the nearest unvisited customer next, aiming for quick, feasible solutions with minimal travel distance.

From the list with those X customers, one customer is chosen randomly. After that, all restrictions are checked, and if they hold, the customer is added to the tour. The first restriction in this case considers the length of the tour. If the customer can be added to the tour and does not exceed the maximum distance of the tour, the second restriction is

checked. The second restriction considers the vehicle capacity. The algorithm checks for each demand if the capacity of the current tour is sufficient.

Every compartment is checked to see if the next customer can be added. If both of these constraints are satisfied, the customer is added. Otherwise, the tour is closed. This means that the tour is finished in the generating process and added to the initial solution. The customer that could not be added is then discarded. A new tour starts by randomly selecting a depot that has empty tours left. For this depot, the closest X customers are selected, and one of them is randomly chosen.

This procedure continues until no customer is left unvisited.

6.1.1 Local Search

In this phase the proposed algorithm tries to optimize the obtained solutions and therefore tries to find new optimal solutions in the neighborhood of the current solution. This local search is performed after the GRASP construction and after each shaking phase. There are some minor differences in the use of the operators depending on when the local search is performed.

There are three intra-route optimizations and two inter-route optimizations. The intra-route operators focus on optimizing a single tour, while the inter-route operators focus on optimizing the combined distance of two tours.

All of the following optimization procedures are performed on each tour at least once after the construction is completed. After each iteration of the shaking phase, if the tour has at least two customers, all operators are used. If a tour has one customer, only the one-move optimization is tried. It can happen that one tour gets to run through this phase more than once, this will be explained in detail in the sections for one-move and two-move.

It should be noted that, for the construction of the initial solution, every tour starts and ends at the same depot. Changes in the starting or arriving depots are allowed only in the local search after a shaking phase was performed.

The local search consists of two phases. The first tries to optimize every tour, considering only customers. In the second phase, the algorithm searches for tours, where the starting or arrival depot can be changed for a better result.

For the local search, three different search operators are used. However, when a local search is performed on a tour, only one of these operators is selected. This selection happens randomly. It should also be noted that these operators are only used for intra-route searches. This means that one route at a time is considered, and the customers are allowed to switch places only on the selected route.

For the calculation of the cost changes if an operator is used, all tours are copied and then tested on. Only if an improvement is found in the local search, the original tours are changed.

The first operator is the one-move operator. This operator calculates the cost of every customer in every possible position with one move in the tour.

It works in a way that it takes one customer and moves it from its position to a different position on the tour. All possible positions are calculated for each customer. The customer with the best improvement is moved from his current position to the best position for the tour.

These steps are repeated until no improvement can be made with this operator. Figure 1 shows how this could look like. In this simple case customer 3 is moved from his position to the first position, and therefore the other two customers are pushed back in their positions.

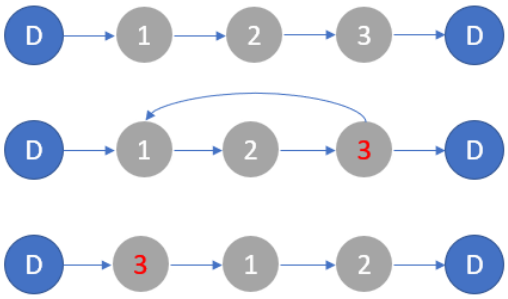

Figure 1: One move intra-route

The second operator for the local search is the swap operator. This operator calculates the costs for each pair of customers (i, j) if they swapped positions. The calculations for these costs are similar to those of the one-move operator: two customers swap their positions in a tour, and then the cost of this new tour is calculated.

After the calculation of every possible swap, the swap with the best improvement is performed. Then this operator starts these steps again until no further improvement is found. Figure 2 shows how the swap operator could work in a simple case of 3 customers. In this case, customers 1 and 3 switch places in the tour.

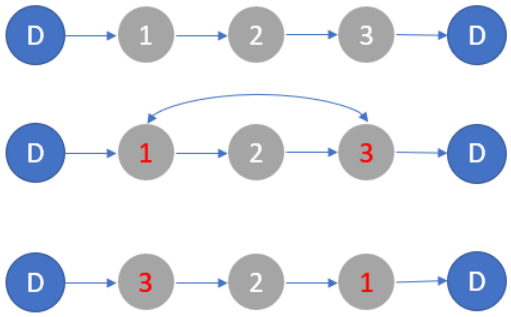

Figure 2: Swap

It is worth mentioning that a swap like this would only make sense in setting where either the starting depot and the finishing depot are different, or the traveling costs between customers and depots are asymmetric.

The last local search operator is the 2-opt operator. This operator takes two customers (i, j) in a tour and reverses all customers between and including them. Figure 3 shows how this would look in a tour with 3 customers.

The 2-opt operator goes through every possible reverse string of the tour and returns with the best result. In the 3 customer case, three different cases need to be calculated. As the tours get bigger, more and more cases need to be calculated.

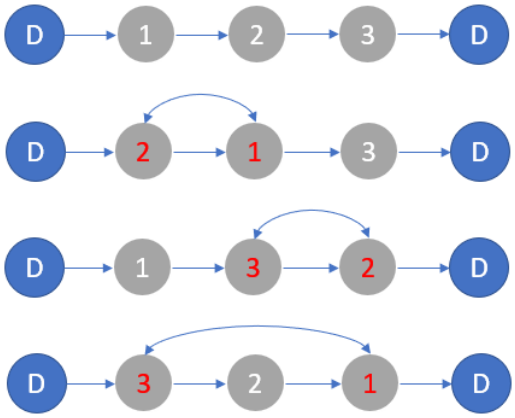

Figure 3: 2-opt

As mentioned above, after all tours are optimized, the algorithm tries to change the depots. Figure 4 shows how this might look like.

Changing a depot is confronted with restrictions. As the depots only have a limited number of vehicles, the changes are limited. This prevents that all tours start or end in the same depot.

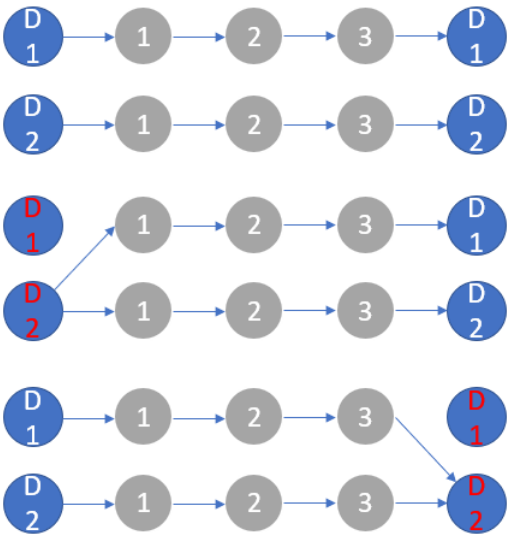

Figure 4: Starting depot change and finishing depot change

First, the algorithm tries to change the starting depots. To do this, the distances between the first customer and every possible depot are calculated for every tour.

If there are depots that are closer to the first customer in a tour than the current depot, the best depot is considered for change. Afterwards, the depots are being changed if the new depots have enough capacity.

This starts with the tour that has the biggest improvement and is done until no more tours are left to change the starting depot. The limitation for this is three changes at maximum.

If a tour changes a depot, it is important to keep track of the vehicles for the affected depots. The previous starting depot gets a freed up vehicle because one tour less starts from there, whereas the new starting depot needs to get an occupied vehicle.

After the changes with the starting depots, the algorithm performs the same procedure for the finishing depots. The procedure for the finishing depots is the same as in the starting depots, concerning the calculation and the selection of changed depots.

Since this problem only considers one period, the tracking of how many vehicles go in which depot is redundant. However, for a potential periodic setting, this needs to be tracked. Otherwise, the possibility arises that multiple depots are left without a vehicle in future periods.

After the depot changes, the algorithm tries again to optimize the tours with the three optimization operators introduced earlier.

6.2 Variable Neighborhood Search (VNS)

As already stated in the literature review above, the VNS is a simple to implement but effective metaheuristic to solve optimization problems (4.3). Its main function is to shake the obtained solutions to escape local optima and then perform a local search. This is done until a set termination criterion is met.

The proposed VNS algorithm, which is used for the experimental runs, is shown in Algorithm 2.

For the reader who might not be as familiar with pseudo codes, the following is an explanation of the algorithm. The Algorithm takes the initial solutions S created by the GRASP and saves them as the best solution T. Line two mentions a set of neighborhood structures, which are all operators that are explained in the next section the shaking phase.

The parameter l is a counter that is used to decide which neighborhood structure is used, and k is a counter that tracks the number of runs without improvement. The last parameter α is used, so the algorithm can reset the count of improvement runs when the current solution is not better than the best solution, but in the range of 5% to the best solution.

The basic construct of the VNS is very simple. The algorithm performs until a set termination criterion is met, in this case a predefined number of runs without improvement. While this criterion is not met, the algorithm takes the solution S and first performs a shaking phase, to get the new solution s'. On this new solution, a local search is performed to get to the solution s^*. Then this solution s^* is compared with the best current solution t.

There are three different outcomes when the two solutions are compared.

Algorithm 2 Variable Neighborhood Search (VNS)

1: Input: initial Solution S (), best Solution T ();
2: a set of neighborhood structures N_l , l = 1,2,..., l_{max}
3: $l = 1$;
4: $k = 0$;
5: $\alpha = 5\%$
6: **while** $k < improvruns$ **do**
7: s' = Shaking (S, N_l);
8: s^* = local search (s');
9: **if** $s^* < t$ **then**
10: $t \leftarrow s^*$
11: l = 1;
12: k = 0;
13: **else if** $s^* < s$ or $s^* < t \cdot (1 + \alpha)$ **then**
14: $s \leftarrow s^*$
15: l = 1;
16: k = 0;
17: **else**
18: $k = k$ + 1;
19: **if** $l \neq l_{max}$ **then**
20: $l = l$ + 1;
21: **else**
22: l = 1;
23: **end if**
24: **end if**
25: **end while**

First, the new solution s^* is better than the best solution. In this case, s^* becomes the best solution t. If this happens, then both counters l and k are reset,and the algorithm starts again with a shaking phase on the new solution t. (Algorithm lines 9-12)

Secondly, the new solution s^* is better than the current solution s, or it is in the 5% range of the best solution. If this happens, the new solution s^* becomes the current solution s and both parameters are reset. Then the algorithm starts again with the shaking procedure. (Algorithm lines 13-16)

The last result is that the new solution s^* is none of the above. In this case, k is incremented by one and if l is not at its maximum, it is also incremented by one. If l is already at its maximum, it is reset to 1. (Algorithm lines 17-23)

6.2.1 Shaking phase

As mentioned above, the shaking phase consists of various different neighborhood structures. These shaking operators are used to change tours, to find different neighborhoods of the initial solution that can be searched for better solutions. In the experiment for this thesis, six different shaking operators were used, and they are explained in detail below.

The first shaking operator is the one move operator, as shown in Figure 5. This operator tries to shift one customer from one tour to a different tour. In this easy example, customer 3 is shifted from its tour to the other tour.

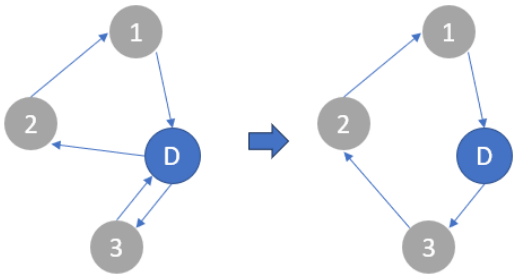

Figure 5: One Move

The algorithm does this by selecting two random tours, tour A and tour B, and then checks which customers can be moved from tour A to tour B, considering first the length and then the demand restrictions of the vehicles. For each of these possible customers, every possible position is considered. From all customers in tour A, that are allowed to move to tour B, one is selected randomly, and inserted in a random possible position.

If a combination of two tours has no feasible customer to move, this process is started again with two different randomly selected tours until a move is performed. Before the program starts, it is ensured that all depots combined have more than enough vehicles to visit all customers, with respect to both length and demand restrictions. This ensures that after the construction phase, a few vehicles are always left unused. So in case that all tours are completely full and can not move one customer between two tours, there is the possibility that an empty tour gets selected and the customer is moved to this tour.

The second operator is the two move operator. This operator is shown in Figure 6, and as the name indicates, it is very similar to the one move operator. This operator takes a pair of customers that are next to each other in a tour and moves them to a different tour.

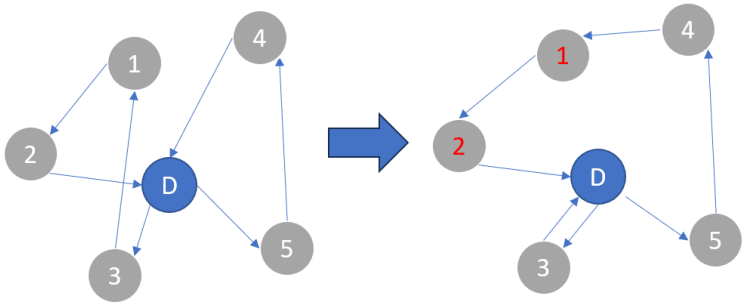

Figure 6: Two Move

The selection of a pair works the same as in the one move operator. All possible subsequent pairs in tour A are considered to move in all available positions in tour B. After

checking with the length and demand restrictions, from the remaining pairs, one is selected randomly. This pair is removed from tour A and inserted into tour B at one of the possible positions for this pair randomly.

As an example in 6, from tour A pairs 1 - 2 and 3 - 1 are tested if the restrictions hold. From all possible pairs, one is selected randomly and inserted in a possible random position on tour B. In this example, this looks like it makes the tours better because the distances between the customers are shorter. Normally, because this is always done randomly, it is more likely to be the other way around.

Like in the one move operator, if no possible pairs can be moved, two new tours are selected, and the same procedure is started again.

The third shaking operator is called the one exchange operator. As the name indicates, this operator tries to take one customer on tour A and one customer on tour B and switch them. Figure 7 shows how this may look. In this simple model, customers 3 and 4 are exchanged in their tours.

The procedure of this operator is not so different from the one move operator. First, it selects two random tours. If one of those tours is an empty tour, a new tour is selected until both tours have at least one customer. For each customer in tour A it is checked if it can switch positions with every customer in tour B. Note that not only the restrictions of tour B have to be checked, but also for tour A, because a customer is switched back.

After all these possible exchanges are calculated, an exchangeable customer from tour A is selected randomly. This customer may not have only one customer option for exchange, so if there is more than one option, out of all possible exchanges, one is selected randomly.

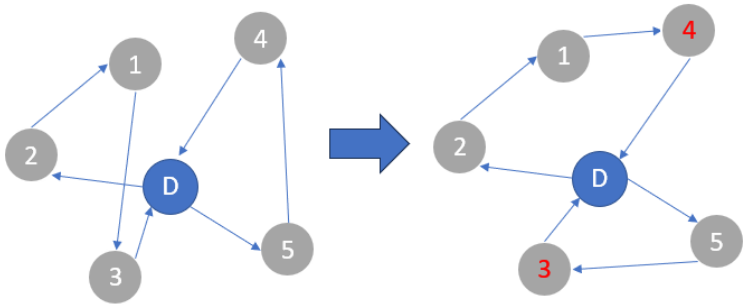

Figure 7: One Exchange

This operator terminates if an exchange is found and performed. Otherwise, it tries again for up to 30 times to find two tours to exchange customers.

The fourth shaking operator is the two exchange operator, presented in Figure 8. It works like a combination of the two move and the one exchange operator. Like the previous operator, this one selects two tours randomly, until two tours are found that have customers in them, except in this case there need to be at least two customers in each tour.

As the name indicates, two pairs are now exchanged between tour A and tour B. To get a list of the possible exchanges, the operator takes all customer pairs that are next to each

other, and calculates if they can be exchanged for pairs in tour B. All possible subsequent pair exchanges are calculated and checked for the length and demand restrictions.

If multiple feasible exchange options are found, the operator selects one random possible pair of tour A and then selects a random possible pair of tour B. The two pairs are then exchanged and inserted in the other tour.

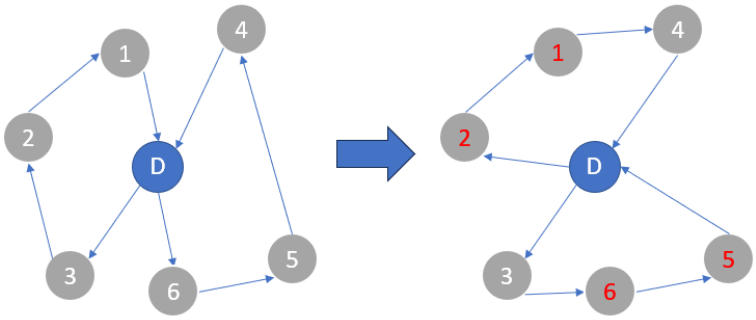

Figure 8: Two Exchange

The termination of this operator is again either a successful found exchange, or it tries up to 30 times to find a feasible exchange.

The last two operators are different from the previous four. In the previous operators, if customers were moved from one tour to another, all their demands were then taken with them. For example, if customer 3 moves from tour A to tour B, all three of his different demands are now served by tour B. The following two operators are different because the first one splits customers, so it has to be served by two different tours, and the second one tries to recombine such split customers.

The fifth operator is called a split operator, which can be seen in Figure 9, and as just stated, it splits the demands of a customer. There are different ways to divide the demands. One way might be to split a single demand. In such a case, the one specific demand needs to be served by two vehicles.

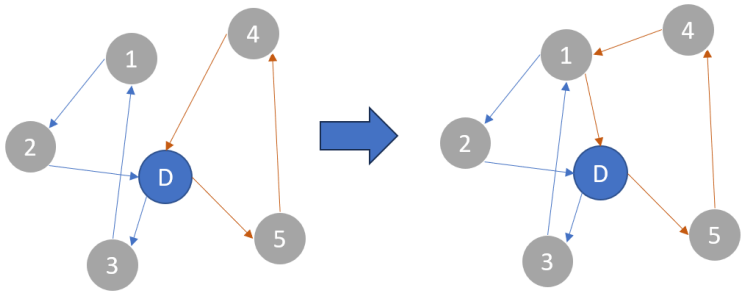

Figure 9: Split

The other way, which is used in this thesis, is splitting the demands of a customer, so that a the split demand needs to be fulfilled whole by a second tour. For example, if a customer has three different demands, the operator takes one of these demands, let it be the second demand, and transfers it into a different tour. Therefore, this customer is still visited by

its original tour and is served by it, but only the first and the third demand. The second demand is satisfied by the new tour.

In the experiment, the number of splits for a specific customer is limited to one. If more than two demands are considered in a test instance, only one demand is allowed to change to another tour.

The way the algorithm works for this operator is that if a certain number of split customers is not reached yet, it searches for a customer until it finds one that is not already split and then chooses one of its demands randomly.

After that, the algorithm considers every tour that does not contain the customer and checks if the length and capacity restrictions hold. Of all those possible tours, one is randomly selected. The customer demand is split, and the split part is inserted into the new tour.

Because the customer needs to be visited by the new tour and not every position in the tour might be possible, the customer gets inserted at a place in the tour that is possible randomly. In the end, the split demand that is still in the original tour is removed from the original tour.

The last operator, shown in Figure 10, works as a split reverse operator. It is used to try an recombine a split customer, to make room for different customers to be split. As mentioned before there is a limit on how many customers are allowed to be split, and this operator tries to delay that this capacity is reached.

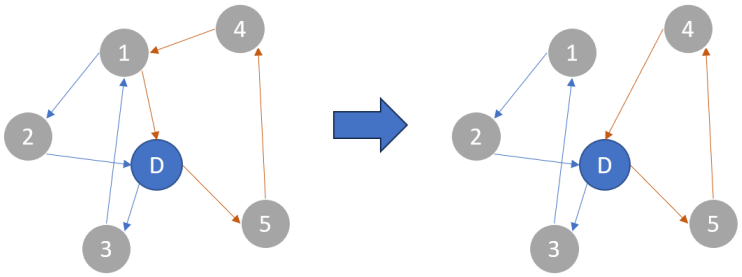

Figure 10: Split Reverse

Of all split customers, one is selected randomly. It is checked into which tours the customer is split and if it is possible to recombine the customer in one of those tours. If both tours are feasible, one of them is selected randomly and the customer and all his demands are combined in this one tour.

6.2.2 Local search

After the shaking procedure, a local search is performed. As this local search is very similar to the local search in the GRASP, with the addition of the changed starting and finishing depots, the reader might want to reread about it in chapter 6.1.1.

6.3 GRASP - VNS Hybrid

The general layout of the GRASP-VNS hybrid algorithm is rather simple as shown in the algorithm 3. The hybrid algorithm reads the test instances and creates the initial solution S with the GRASP.

While a certain number of runs without improvement is not reached, the algorithm performs the VNS. If that number is reached, it generates a new initial solution S.

Until a certain time limit is not reached, these steps repeat themselves. The initial solution S is also stored and updated if a better initial solution is found.

After reaching the time limit, the algorithm returns the best solution found by the GRASP, S^{GRASP}, and the best overall solution found, S^*.

Algorithm 3 GRASP - VNS Hybrid

```
Input: Test Instance
Solution S = greedy randomized adaptive search procedure (GRASP)
S* = S
S^GRASP = S
while time limit not reached do
    S* = Variable Neighborhood Search(S);
    if number of runs without improvement are reached then
        generate new initial Solution S with GRASP
        if S < S^GRASP then
            S^GRASP = S
        end if
    end if
end while
return S*
return S^GRASP
```

6.3.1 Solution Representation

When the algorithm has reached its termination criterion, it prints the values of the solution in a text file. An example of what this looks like is added in the appendix(9).

As mentioned in the previous section, the main values that are printed are the best GRASP value, the best solution value, and the number of vehicles used in the best solution.

First, the text file shows different time stamps, every 10 minutes, where it is possible to see how the GRASP solution and the best solution improve over time. The other time stamps show when a new best solution is found.

In addition, the tours of the best solution are printed, including relevant data. This means: which depot does a tour start and where it ends, the length of a tour, the load of a tour, and the free capacity it still has for each compartment and every customer that is in this tour.

This was mainly created in the beginning to check if the algorithm works as it is intended. As in every project, this one also had its starting problems. The information from these solution printouts was used to check if the length and capacity constraints were working correctly. In addition, they were also used to check if all customers were visited.

7 Experimental Results

As the algorithm has been explained in detail, this chapter focuses on the experimental results and explains them.

The experiments were run on a home computer with an AMD Ryzen 5 3600 6-core processor and a 32 GB RAM. It should be noted that at the beginning, no multi-threading was used and one test instance at a time resulted in an average usage of 8% of the CPU.

As the experimenting evolved, multi-threading was used to save time. To avoid overloading the CPU, 10 test instances were run simultaneously, resulting in an average usage of 80%. The algorithm is programmed in the language C# in Visual Studios.

During the course of building the complete algorithm, many different variations of the program were made. Some of them were used to test the algorithm for feasibility and fine-tuning, which will not be explained in detail.

7.1 Test Instances

As mentioned in Section 5, the hybrid algorithm needs test instances as input files to work with. Before the GRASP construction starts, the algorithm needs the input file. This file is structured as shown in Figure 11.

The instances used were first introduced by Cordeau et al.[CGL97] in 1997. Of the 33 available test instances, only the first 23 were considered. These large-size problems have populations ranging from 50 to 360 customers and multiple depots ranging from 2 to 9.

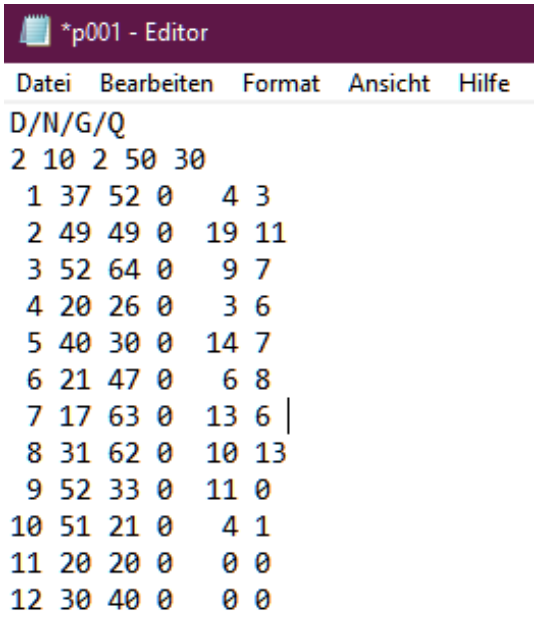

Figure 11: Example of an input file

As they original instances only considered the extension of multi-depots, the instances are modified to fit the problem description for multi-depots and multi-compartments. In the

example shown, the original first instance was used and reduced to 10 customers and 2 depots, to get a small instance, to test if the algorithm was working properly.

The first row shows what comes in which order in the second row. The D represents the depots, the N represents the customers, the G represents the number of compartments, and the Q represents the capacity in each of the compartments. Going to the second row, there are 2 depots, then 10 customers. Each of those customers has two different demands. The first three parts are always one single number, and after that, the last numbers are a list which can be either 2 or more, depending on how many compartments the test instance has. As this was only a test instance, it was used to confirm if the algorithm works, if the compartments of the vehicle can be different values as well.

From the third row onward, until every customer is listed, the rows are in the same order. The first number always represents the ID of a customer, and the second and third numbers are the X and Y coordinates. The fourth number represents the service duration and can be ignored. The last two numbers represent the demands for each product of the customer.

The rows following the customer rows represent the depots and their data. The first number represents the ID of the depot, and the second and third numbers are the X and Y coordinates. The other numbers can be ignored since the depot has no service duration and no demands or capacity restrictions in the depots are considered.

Since depots do not have restricted capacity considering demands, they are only restricted by the initially number of vehicles assigned to them.

In Table 1, all test instances are listed with their basic information. In these instances, the first column shows the names of the different test instances. The other four columns are the same as the columns shown in Figure 11.

Column D shows that the number of depots in each test instance can vary from 2 to 9 depots. The next column shows that the amount of customers is at least 50 and up to 360 customers, that need to be served. Column G shows that each instance deals with either 2 or 3 different demands, and the last column shows the capacity of each of those demands of each vehicle.

The test instances p01-p07 and p12, p15, p18, p21 do not have any restrictions regarding the traveled distance.

7.2 Demands

During the construction of different scenarios, the idea of Alinaghian and Shokouhi [AS18], to use randomized demands, was used for test runs.

In the following sections different demand variations were used, but all data tables were constructed in the same way.

In the first run of a test instance in a certain demand setting, the demand is created. This is done by taking the single demand of the original test instances and dividing it by a certain percentage.

In the two-compartment setting, the first demand is equal to the random percentage of

Instance	D	N	G	Q
p01	4	50	3	30
p02	4	50	3	60
p03	5	75	2	70
p04	2	100	2	50
p05	2	100	2	100
p06	3	100	2	50
p07	4	100	2	50
p08	2	249	3	170
p09	3	249	3	170
p10	4	249	3	170
p11	5	249	3	250
p12	2	80	2	30
p13	2	80	2	30
p14	2	80	2	30
p15	4	160	2	80
p16	4	160	3	70
p17	4	160	3	60
p18	6	240	2	30
p19	6	240	2	100
p20	6	240	2	90
p21	9	360	2	30
p22	9	360	2	75
p23	9	360	2	90

Table 1: Instances

the original demand that is calculated, and the second demand is the difference between the original demand and the first demand. These percentages can only be in a predefined range. For example, if the range is set to 30-70, the first demand percentage can be anything between those numbers.

In the three-compartment setting, the first demand is calculated equally. The second demand is calculated in the same way, but only as a percentage of the difference between the original demand and the first demand. The last demand is calculated by the difference between the original demand and the first and second demand.

It should be mentioned that the percentages of the first and second demand are not the same and are calculated individually. For a better distribution of the demands, closer percentages were used in the three-compartment setting as shown in Figure 12.

This example shows that even though the percentages are closer, due to the randomization of the distribution, some products have a higher demand than others. For example, customer 2, 5 and 8, and their demands for product 1.

After creating those demands for a test instance, they are stored to be reused. Each test instance created in this way is run five times. This is done for every different best-of variation

Figure 12: Demand Example

and for every improvement scenario. There are three different improvement scenarios in every demand scenario.

For example, a p01 test instance in the first demand scenario is run five times with the best of 1(BO1), five times with the best of 2(BO2) and five times with the best of 3(BO3), in an improvement run setting of 50. From all those runs the minima of each best of are compared and the absolute minimum of those is taken into the comparison table, which are shown in the next sections.

All other results are added in the appendix.

7.3 First Demand Variation

For the first demand variation, the results of these experiments are shown in Table 2. The demands for this experiment were done in a 30-70 percent setting for the two demand cases and a 10-60 percent setting for the three demand cases.

This means that in the first setting the first demand is chosen by randomly selecting a percentage between 30 and 70. In the three demand cases, the first demand is a random percentage between 10 and 60 percent of the original demand, and the second is a random new percentage of the difference of those two. The last demand is the remaining demand.

The following tables show the results obtained by the algorithm and show their data in the following way: The first row, from left to right, explained is, first, the test instances and then groups of two different values. The number of runs indicates what value the improvement run parameter was set in each scenario. As stated above, every scenario is done five times. The next value, called GRASP%, indicates how close the GRASP-generated result, without the VNS, got to the final result.

Instance	50 Runs	GRASP%	100 Runs	GRASP%	150 Runs	GRASP%
p01	665,26	1,61[2]	670,78	0,51[2]	670,91	2,32[2]
p02	515,09	3,08[1]	505,34	7,68[2]	509,07	4,39[2]
p03	678,87	1,22[1]	687,17	1,91[1]	692,28	1,97[1]
p04	1141,34	2,20[2]	1153,82	1,19[2]	1169,88	0,64[2]
p05	838,68	0,67[2]	827,77	2,90[2]	841,88	2,86[2]
p06	998,51	2,68[2]	993,54	1,82[2]	1019,61	0,39[2]
p07	1005,33	1,86[2]	1010,38	2,98[1]	1011,55	2,41[2]
p08	5961,68	1,32[1]	5988,14	0,81[1]	5955,49	1,49[1]
p09	5098,35	1,36[2]	5114,12	1,70[2]	5177,71	1,92[2]
p10	4825,72	0,74[2]	4799,73	1,38[1]	4828,65	1,34[1]
p11	4144,82	1,99[1]	4204,30	1,22[1]	4174,21	0,98[1]
p12	1375,88	3,87[2]	1399,46	4,33[2]	1433,87	2,70[2]
p13	1442,53	9,80[2]	1448,39	9,72[2]	1453,08	11,83[2]
p14	1577,71	4,70[2]	1633,72	6,46[2]	1603,05	8,02[2]
p15	2412,95	0,58[1]	2356,62	2,81[1]	2424,94	0,72[1]
p16	3040,64	4,41[1]	3008,52	5,31[1]	2981,30	6,85[1]
p17	3227,14	14,08[2]	3129,00	18,63[1]	3251,65	14,38[1]
p18	4469,85	3,56[1]	4455,18	5,50[1]	4436,67	6,12[1]
p19	4619,99	7,85[1]	4668,63	7,05[1]	4704,81	6,40[1]
p20	4962,62	12,69[1]	4977,47	9,70[1]	5020,37	11,46[1]
p21	6843,84	5,60[1]	6888,71	4,84[1]	6988,80	4,15[1]
p22	7020,54	11,37[1]	7176,04	8,87[1]	7142,19	9,24[1]
p23	7642,11	10,69[1]	7587,21	10,05[1]	7689,98	10,41[1]
Average	3239,54	4,69	3247,13	5,10	3268,78	4,91

Table 2: Comparison: VNS - GRASP Demands 30-70, 10-60

In this case, in row p01, the number 665,26 is the minimum of those five runs. The next number is the percentage of how close the GRASP came to the best solution. In this example, in the first row, this percentage is 1,61. This means that GRASP was able to achieve a value of 675,97. The small index over the percentages indicates, which best-of was able to achieve this value. In this case, this would be the BO2.

The next segment states the best value achieved in a setting with 100 improvement runs, in this case 670,78. The best value achieved by GRASP in this setting is only 0,51 % higher, which means 674,20 and was also achieved in the BO2 setting.

For the last pair, this is the same in a setting with 150 improvement runs.

The last line in the table shows the average values of all runs. The average distance over all runs for the three different variations is not that far apart. Although the first variation has the best average, the second variation is very close. Even the third variation is in the range of 1% to the best average. Although the average GRASP percentage is higher in the second variation, the average distance is close to the best average.

The close to 5 % range of the best GRASPs indicates that overall it gives good initial solu-

tions that can be improved. There are some instances that are clearly far away from that percentage, especially from p17 to p23. This indicates that the complexity in these test instances might be too high for the GRASP to get good initial solutions. The combination of many customers, more depots, and little capacity in the compartments could be the reason for that.

In p08 to p11 the GRASP performs better, even though the number of customers is very high. This might be due to the higher capacities in the compartments.

The table also shows that, for all runs, the best GRASP solution was not reached by the BO3 variation. In smaller instances, GRASP performs better with a BO2 approach, but as instances grow larger and more complex, BO1 dominates.

An example that indicates that BO1 and BO2 are not that far apart is p17. Although this is the instance where the difference between GRASP and the best found solution is the largest, the best found solution for the GRASP was found in the 50 runs case by BO2 and in the other cases by BO1.

7.4 Second Demand Variation

In this second demand variation, as shown in Table 3, demand percentages are made in a 30-70 percent setting for the two demand cases and a 20-60 percent setting for the three demand cases.

As for the results of this demand variation, the overall results are not much different than in the first variation. The first run variation has the best overall distance value with 3219,83, but is closely followed by the second run variation with 3251,39 and the third variation with 3251,51.

The overall GRASP performance is in a similar percentage window between around 4-5% of the total distance. In these cases, the second variation has the best GRASP percentage, and the first variation even got the highest percentage out of all three variations.

The BO variation performed similar to the first demand variation, with the BO1 performing best in the later instances and the BO2 better in the earlier test instances. The BO3 has no best result.

Although BO3 has no best results to show in this table, the data obtained shows that it also obtains good results. As the tables in 9.2 show, BO3 performs best in the 150 run case. In multiple instances it gets close to the other BO variants and in case of p04 it even gets the best result of all BO variants.

Instance	50 Runs	GRASP%	100 Runs	GRASP%	150 Runs	GRASP%
p01	628,37	2,99[2]	643,02	0,59[2]	639,29	1,08[2]
p02	505,42	1,48[2]	505,23	2,16[2]	494,36	5,64[2]
p03	709,77	0,19[2]	718,13	1,36[1]	721,44	1,21[1]
p04	1126,26	2,41[2]	1153,82	1,09[2]	1157,58	1,16[2]
p05	829,96	1,35[2]	820,39	3,53[2]	836,04	1,94[2]
p06	988,72	3,12[2]	1004,51	1,74[2]	1007,74	0,49[2]
p07	1011,55	1,76[2]	1010,18	2,12[1]	1012,66	1,41[1]
p08	5754,55	0,51[1]	5726,39	1,05[1]	5711,93	1,41[1]
p09	4974,23	1,58[2]	5012,41	1,11[1]	5001,8	0,81[1]
p10	4617,06	1,71[1]	4665,14	0,05[1]	4602,42	2,52[1]
p11	4202,77	1,72[1]	4220,08	1,51[1]	4232,01	0,56[1]
p12	1364,16	4,18[2]	1389,52	3,28[2]	1382,83	5,03[2]
p13	1445,8	9,2[2]	1480,75	7,55[2]	1435,6	10,98[1]
p14	1562,89	12,36[2]	1631,4	6,09[2]	1617,23	9,01[1]
p15	2405,67	0,47[1]	2371,38	3,24[1]	2371,38	1,43[1]
p16	2994,38	5,34[1]	3024,14	5,31[1]	2986,32	7,04[1]
p17	3177,67	17,04[1]	3212,25	15,55[1]	3310,42	12,35[1]
p18	4430,64	5,57[1]	4508,69	3,60[1]	4489,98	5,11[1]
p19	4701,96	6,69[1]	4685,35	6,53[1]	4668,19	6,69[1]
p20	4875,89	13,88[1]	5057,08	1,10[1]	5072,56	9,92[1]
p21	7063,08	4,60[1]	7125,98	5,49[1]	7168,88	4,96[1]
p22	7100,97	9,14[1]	7188,76	10,30[1]	7191,56	8,90[1]
p23	7584,26	11,13[1]	7627,28	11,65[1]	7672,62	11,87[1]
Average	3219,83	5,15	3251,39	4,17	3251,51	4,85

Table 3: Comparison: VNS - GRASP Demands 30-70, 20-60

7.5 Third Demand Variation

Table 4 shows the third variation that was used. In this variation a 35 to 65 percent setting for the two demand case and a 25 to 65 percent setting for the three demand case were used.

Compared to GRASP, this variation performed very similarly to the other two demand variations, with an average percentage difference of around 5%. As in the other two demands, this one also has some instances where the GRASP performed well and got nearly as good results as the combined algorithm.

For example, in the instances p02, p04, p07, p08, and p12 the GRASP was in the range of 0,5 % percent to the best result found, in one of the different run experiments.

Overall, in the third demand variation, the GRASP obtains very close results for 12 instances.

Unlike in the other demand variations, the best GRASP percentage was in the in 150 run variant. The 100 run variation has the worst percentage, even though all percentages of all run variations are within 0.25%.

Instance	50 Runs	GRASP%	100 Runs	GRASP%	150 Runs	GRASP%
p01	655,77	1,90[2]	655,96	2,35[2]	660,39	2,12[2]
p02	499,26	3,08[2]	503,73	0,25[2]	505,93	1,98[2]
p03	701,71	1,15[1]	689,79	3,05[1]	686,00	3,62[1]
p04	1120,48	1,47[1]	1130,08	0,50[2]	1131,03	1,15[1]
p05	814,87	1,03[2]	826,91	2,63[2]	833,07	1,31[2]
p06	987,41	1,64[1]	976,88	1,88[2]	988,20	1,93[2]
p07	1013,77	0,45[2]	990,87	4,76[2]	1014,91	1,07[2]
p08	6105,57	0,80[2]	6191,06	0,22[1]	6170,80	0,70[1]
p09	5149,02	2,22[2]	5219,42	0,96[2]	5205,19	1,40[2]
p10	4763,51	2,79[1]	4874,55	0,61[1]	4821,78	2,26[1]
p11	4168,57	1,20[1]	4171,23	1,09[1]	4176,87	0,66[1]
p12	1424,92	0,33[2]	1385,59	3,89[2]	1373,39	3,56[2]
p13	1448,39	9,63[2]	1420,81	12,40[2]	1448,39	9,44[2]
p14	1517,95	14,69[2]	1585,10	8,58[2]	1601,17	6,62[2]
p15	2412,79	0,79[1]	2408,25	0,88[1]	2407,13	2,05[1]
p16	2998,46	5,28[1]	3046,02	4,27[1]	3029,13	4,60[1]
p17	3226,14	15,05[1]	3238,80	14,83[1]	3163,28	17,57[1]
p18	4421,56	6,08[1]	4478,41	5,96[1]	4473,27	5,21[1]
p19	4646,12	7,72[1]	4692,35	7,09[1]	4680,42	7,01[1]
p20	5001,14	9,79[1]	4934,50	12,35[1]	5025,40	10,77[1]
p21	6951,91	6,03[1]	6950,38	5,01[1]	7009,19	5,33[1]
p22	7083,12	9,93[1]	7104,52	11,84[1]	7143,97	10,62[1]
p23	7583,74	11,43[1]	7589,66	10,52[1]	7671,30	9,66[1]
Average	3247,66	4,98	3263,69	5,04	3270,44	4,81

Table 4: Comparison: VNS - GRASP Demands 35-65, 25-65

7.6 Comparison of the different Variations

As there are new values in the new test instances created in a different demand variation, the differences in the results are mainly due to the construction of new demands. Therefore, the comparison shows what differences this step can make. Tables 5 and 6 show how much the instant values differ due to the new demands.

The percentage gap between the two demands was calculated with the following formula:

$\% = \frac{D1-D2}{D2} * 100$

As shown in both tables, the two different demands perform similarly, with D2 slightly having better results in the three-compartment cases and D1 slightly performing better in the two-compartment cases. As mentioned above, the difference in the demands between these two variations can be quite significant.

If, for example, one of those instances generates a well-rounded demand, where all the demands are close to the same volume, then it makes it much easier for the algorithm to find a good solution with this customer. On the other hand, if demands are made that have very high demands of one product and the others are more or less not relevant, it makes

Instance	D1	D2	%	D1	D2	%	D1	D2	%
p01	665,26	628,37	5,87	670,78	643,02	4,32	670,91	639,29	4,95
p02	515,09	505,42	1,91	505,34	505,23	0,02	509,07	494,36	2,98
p08	5961,68	5754,55	3,60	5988,14	5726,39	4,57	5955,49	5711,93	4,26
p09	5098,35	4974,23	2,50	5114,12	5012,41	2,03	5177,71	5001,80	3,52
p10	4825,72	4617,06	4,52	4799,73	4665,14	2,89	4828,65	4602,42	4,92
p16	3040,64	2994,38	1,54	3008,52	3024,14	-0,52	2981,30	2986,32	-0,17
p17	3227,14	3177,67	1,56	3129,00	3212,25	-2,66	3251,65	3310,42	-1,81
Average	3333,41	3235,95	3,07	3316,52	3255,51	1,53	3339,25	3249,51	2,67

Table 5: Comparison: Demand variations 1 and 2 with three compartments

Instance	D1	D2	%	D1	D2	%	D1	D2	%
p03	678,87	709,77	-4,35	687,17	718,13	-4,31	692,28	721,44	-4,04
p04	1141,34	1126,26	1,34	1153,82	1153,82	0,00	1169,88	1157,58	1,06
p05	838,68	829,96	1,05	827,77	820,39	0,90	841,88	836,04	0,70
p06	998,51	988,72	0,99	993,54	1004,51	-1,09	1019,61	1007,74	1,18
p07	1005,33	1011,55	-0,61	1010,38	1010,18	0,02	1011,55	1012,66	-0,11
p11	4144,82	4202,77	-1,38	4204,30	4220,08	-0,37	4174,21	4232,01	-1,37
p12	1375,88	1364,16	0,86	1399,46	1389,52	0,72	1433,87	1382,83	3,69
p13	1442,53	1445,80	-0,23	1448,39	1480,75	-2,19	1453,08	1435,60	1,22
p14	1577,71	1562,89	0,95	1633,72	1631,40	0,14	1603,05	1617,23	-0,88
p15	2412,95	2405,67	0,30	2356,62	2371,38	-0,62	2424,94	2371,38	2,26
p18	4469,85	4430,64	0,88	4455,18	4508,69	-1,19	4436,67	4489,98	-1,19
p19	4619,99	4701,96	-1,74	4668,63	4685,35	-0,36	4704,81	4668,19	0,78
p20	4962,62	4875,89	1,78	4977,47	5057,08	-1,57	5020,37	5072,56	-1,03
p21	6843,84	7063,08	-3,10	6888,71	7125,98	-3,33	6988,80	7168,88	-2,51
p22	7020,54	7100,97	-1,13	7176,04	7188,76	-0,18	7142,19	7191,56	-0,69
p23	7642,11	7584,26	0,76	7587,21	7627,28	-0,53	7689,98	7672,62	0,23
Average	3198,47	3212,77	-0,23	3216,78	3249,58	-0,87	3237,95	3252,39	-0,04

Table 6: Comparison: Demand variations 1 and 2 with two compartments

it much harder to find good solutions.

A solution for this problem with random, or changing demands, can be vehicles with changing compartments. After the demands are determined, or created, they are each summed up, and the original demand that in this thesis was divided by 2 or 3, gets divided by the amount of compartments, but each compartment gets weighted this time.

For example, if the original demand would be 200 and the number of compartments would be 2, in the case of this thesis, each compartment gets a capacity of 100. If the compartments are weighted according to the sum of the demands, it can be that the first compartment gets a capacity of 150 and the second one only 50.

This might be better for finding good or optimal solutions, but it may be only a theoretical improvement. In real-world applications, the change of compartments is very limited,

highly dependent on the freight and might be very time consuming.

Table 5 shows that D2 performs better in most three-compartment cases. In the 50 run experiments, D1 could not perform better in any of the instances. In three instances, D2 has a difference close to 5% or higher, two of them in 150 run experiments and one in the 50 run experiments. The results in the 100 run experiments have the best average results and the smallest difference between D1 and D2.

Table 6 shows how D1 and D2 perform in the two-compartment case. Since here, the demand distribution is the same but new demands were made for D2, this might impact some of the solutions. On the other hand, overall D1 performs better, but only by a very small percentage, as the last line shows.

In general, both obtain very similar results again, with better results for D1 in some instances and better results for D2 in others. Only a few outliers are found in all these different instances. Worth mentioning here are the results of D1 in p03 and p21. For D2 in p15 and p12.

With not even a difference of 1% in every average scenario, the results are very close, especially in the 150 run experiments.

The Tables 7 and 8 show how D2 performs compared to D3.

Instance	D2	D3	%	D2	D3	%	D2	D3	%
p01	628,37	655,77	-4,18	643,02	655,96	-1,97	639,29	660,39	-3,20
p02	505,42	499,26	1,23	505,23	503,73	0,30	494,36	505,93	-2,29
p08	5754,55	6105,57	-5,75	5726,39	6191,06	-7,51	5711,93	6170,8	-7,44
p09	4974,23	5149,02	-3,39	5012,41	5219,42	-3,97	5001,80	5205,19	-3,91
p10	4617,06	4763,51	-3,07	4665,14	4874,55	-4,30	4602,42	4821,78	-4,55
p16	2994,38	2998,46	-0,14	3024,14	3046,02	-0,72	2986,32	3029,13	-1,41
p17	3177,67	3226,14	-1,50	3212,25	3238,80	-0,82	3310,42	3163,28	4,65
Average	3235,95	3342,53	-2,40	3255,51	3389,93	-2,71	3249,51	3365,21	-2,59

Table 7: Comparison: Demand variations 2 and 3 with three compartments

Compared to the second demand variation, the third variation performed slightly better overall. For the two compartment instances, in the 50 run case the overall distance is only 0.2%, in the 100 run case 1.53%, and in the 150 run case 0.89%.

In most instances, the results are in a close range of under 3% with only a few exceptions, such as p03, p12, and p13. In instances p12 and p13, these percentages are achieved not only because the result was so much better, but rather because the other demand variation underperformed.

In the three compartment instances, the results change. In nearly every instance and every different run segment, D2 gets better results than D3. In p08, D2 outperforms D3 by up to 7.51%. Although D3 has better results in p02 in the 50- and 100-run experiments, in the 150-run experiment D2 outperforms the other results. In p17 D3 even gets the best result in all 3 experiments.

As the table shows, overall, D2 performs better than D3, but D3 still has some very good results and has the potential to get even better results than D2.

Instance	D2	D3	%	D2	D3	%	D2	D3	%
p03	709,77	701,71	1,15	718,13	689,79	4,11	721,44	686,00	5,17
p04	1126,26	1120,48	0,52	1153,82	1130,08	2,10	1157,58	1131,03	2,35
p05	829,96	814,87	1,85	820,39	826,91	-0,79	836,04	833,07	0,36
p06	988,72	987,41	0,13	1004,51	976,88	2,83	1007,74	988,20	1,98
p07	1011,55	1013,77	-0,22	1010,18	990,87	1,95	1012,66	1014,91	-0,22
p11	4202,77	4168,57	0,82	4220,08	4171,23	1,17	4232,01	4176,87	1,32
p12	1364,16	1424,92	-4,26	1389,52	1385,59	0,28	1382,83	1373,39	0,69
p13	1445,80	1448,39	-0,18	1480,75	1420,81	4,22	1435,60	1448,39	-0,88
p14	1562,89	1517,95	2,96	1631,40	1585,10	2,92	1617,23	1601,17	1,00
p15	2405,67	2412,79	-0,30	2371,38	2408,25	-1,53	2371,38	2407,13	-1,49
p18	4430,64	4421,56	0,21	4508,69	4478,41	0,68	4489,98	4473,27	0,37
p19	4701,96	4646,12	1,20	4685,35	4692,35	-0,15	4668,19	4680,42	-0,26
p20	4875,89	5001,14	-2,50	5057,08	4934,50	2,48	5072,56	5025,40	0,94
p21	7063,08	6951,91	1,60	7125,98	6950,38	2,53	7168,88	7009,19	2,28
p22	7100,97	7083,12	0,25	7188,76	7104,52	1,19	7191,56	7143,97	0,67
p23	7584,26	7583,74	0,01	7627,28	7589,66	0,50	7672,62	7671,30	0,02
Average	3212,77	3206,15	0,20	3249,58	3208,46	1,53	3252,39	3228,98	0,89

Table 8: Comparison: Demand variations 2 and 3 with two compartments

The last two Tables 9 and 10 compare the performance of D3 and D1.

Table 9 shows, in the two-compartment instances, that both demand variations achieve similar results with D3 slightly better average results in the 100 and 150 run experiments. In the 50 run experiments, D1 achieves better overall results, with good performances in p03 and p12. In particular, D3 obtains a very good result in p14.

In the 150 run experiment, with the exception of the results in p04, p06, and p12, every result achieved was in the range of 1% to the other demand variation. With p05 closely over 1%.

Instance	D1	D3	%	D1	D3	%	D1	D3	%
p01	665,26	655,77	1,45	670,78	655,96	2,26	670,91	660,39	1,59
p02	515,09	499,26	3,17	505,34	503,73	0,32	509,07	505,93	0,62
p08	5961,68	6105,57	-2,36	5988,14	6191,06	-3,28	5955,49	6170,80	-3,49
p09	5098,35	5149,02	-0,98	5114,12	5219,42	-2,02	5177,71	5205,19	-0,53
p10	4825,72	4763,51	1,31	4799,73	4874,55	-1,53	4828,65	4821,78	0,14
p16	3040,64	2998,46	1,41	3008,52	3046,02	-1,23	2981,30	3029,13	-1,58
p17	3227,14	3226,14	0,03	3129,00	3238,80	-3,39	3251,65	3163,28	2,79
Average	3333,41	3342,53	0,57	3316,52	3389,93	-1,27	3339,25	3365,21	-0,06

Table 9: Comparison: Demand variations 1 and 3 with three compartments

Table 10 shows that in the three compartment instances D3 performs better in the 50 run experiment, while D1 performs better in the 100 run experiment. In the 150 run experiment both achieve similar average results.

In the instances p01 and p02, D3 performs better in every experiment setting, while in instances p08 and p09 D1 performs better in all settings.

Instance	D1	D3	%	D1	D3	%	D1	D3	%
p03	678,87	701,71	-3,25	687,17	689,79	-0,38	692,28	686,00	0,92
p04	1141,34	1120,48	1,86	1153,82	1130,08	2,10	1169,88	1131,03	3,43
p05	838,68	814,87	2,92	827,77	826,91	0,10	841,88	833,07	1,06
p06	998,51	987,41	1,12	993,54	976,88	1,71	1019,61	988,20	3,18
p07	1005,33	1013,77	-0,83	1010,38	990,87	1,97	1011,55	1014,91	-0,33
p11	4144,82	4168,57	-0,57	4204,30	4171,23	0,79	4174,21	4176,87	-0,06
p12	1375,88	1424,92	-3,44	1399,46	1385,59	1,00	1433,87	1373,39	4,40
p13	1442,53	1448,39	-0,40	1448,39	1420,81	1,94	1453,08	1448,39	0,32
p14	1577,71	1517,95	3,94	1633,72	1585,10	3,07	1603,05	1601,17	0,12
p15	2412,95	2412,79	0,01	2356,62	2408,25	-2,14	2424,94	2407,13	0,74
p18	4469,85	4421,56	1,09	4455,18	4478,41	-0,52	4436,67	4473,27	-0,82
p19	4619,99	4646,12	-0,56	4668,63	4692,35	-0,51	4704,81	4680,42	0,52
p20	4962,62	5001,14	-0,77	4977,47	4934,50	0,87	5020,37	5025,40	-0,10
p21	6843,84	6951,91	-1,55	6888,71	6950,38	-0,89	6988,80	7009,19	-0,29
p22	7020,54	7083,12	-0,88	7176,04	7104,52	1,01	7142,19	7143,97	-0,02
p23	7642,11	7583,74	0,77	7587,21	7589,66	-0,03	7689,98	7671,30	0,24
Average	3198,47	3206,15	-0,04	3216,78	3208,46	0,63	3237,95	3228,98	0,83

Table 10: Comparison: Demand variations 1 and 3 with two compartments

8 Conclusion

This thesis used and adapted an existing mathematical model to solve a multi-depot, multi-compartment vehicle routing problem. The objective function of the proposed problem is to minimize the total traveled distance.

Real-life applications of these problems can be found in all kinds of industries, such as the distribution of dairy products, oil, fuel distribution, etc.

The experimental results show that the algorithm works and that the combination of GRASP and VNS are a viable option to solve the VRP with several extensions. Since there is only one scientific paper considering this very specific problem, but with a slightly different problem setting, it is not possible to directly compare the results.

However, the data obtained shows that considering different GRASP values is very effective in obtaining initial tours. Combined with the VNS the algorithm achieved good results in a reasonable time.

Future possibilities for this problem can be to add another extension, for example, time windows or more periods. Another extension might be a heterogeneous fleet of vehicles to serve customers. This would make this problem even more realistic, but would increase its complexity.

References

[AS18] Mahdi Alinaghian and Nadia Shokouhi. “Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search”. In: *Omega* 76 (2018), pp. 85–99. ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2017.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0305048316303553.

[BG81] Gerald G. Brown and Glenn W. Graves. “Real-Time Dispatch of Petroleum Tank Trucks”. In: *Management Science* 27.1 (1981), pp. 19–32. DOI: 10.1287/mnsc.27.1.19. URL: https://doi.org/10.1287/mnsc.27.1.19.

[CCM24] Tiziana Calamoneri, Federico Corò, and Simona Mancini. “Management of a post-disaster emergency scenario through unmanned aerial vehicles: Multi-Depot Multi-Trip Vehicle Routing with Total Completion Time Minimization”. In: *Expert Syst. Appl.* 251.C (Oct. 2024). ISSN: 0957-4174. DOI: 10.1016/j.eswa.2024.123766. URL: https://doi.org/10.1016/j.eswa.2024.123766.

[CGL97] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. “A Tabu Search heuristic for periodic and multi-depot vehicle routing problems”. In: *Networks* 30 (Sept. 1997), pp. 105–119. DOI: 10.1002/(SICI)1097-0037(199709)30:23.3.CO;2-N.

[CKD25] Junyang Cai, Serdar Kadioglu, and Bistra Dilkina. *Balans: Multi-Armed Bandits-based Adaptive Large Neighborhood Search for Mixed-Integer Programming Problem*. 2025. arXiv: 2412.14382 [cs.AI]. URL: https://arxiv.org/abs/2412.14382.

[CR24] Shabanaz Chamurally and Julia Rieck. “A practical and robust approach for solving the multi-compartment vehicle routing problem under demand uncertainty using machine learning”. In: *Networks* 84.3 (2024), pp. 300–325. DOI: https://doi.org/10.1002/net.22231. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.22231. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.22231.

[CW64] G. Clarke and J. W. Wright. “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points”. In: *Operations Research* 12.4 (1964), pp. 568–581. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/167703 (visited on 05/13/2025).

[DR59] G. B. Dantzig and J. H. Ramser. “The Truck Dispatching Problem”. In: *Management Science* 6 (Oct. 1959), pp. 80–91.

[DT89] Moshe Dror and Pierre Trudeau. “Savings by Split Delivery Routing”. In: *Transportation Science* 23.2 (May 1989), pp. 141–145. DOI: 10.1287/trsc.23.2.141.

[FR95] Thomas Feo and Mauricio Resende. “Greedy Randomized Adaptive Search Procedures”. In: *Journal of Global Optimization* 6 (Mar. 1995), pp. 109–133. DOI: 10.1007/BF01096763.

[FYY23] Xueru Fan, Guanxin Yao, and Yang Yang. “Multi-Compartment Vehicle Routing Problem Considering Traffic Congestion under the Mixed Carbon Policy”. In: *Applied Sciences* 13.18 (2023). ISSN: 2076-3417. DOI: 10.3390/app131810304. URL: https://www.mdpi.com/2076-3417/13/18/10304.

[GDC24] Juan Carlos Gonçalves-Dosantos, Laura Davila-Pena, and Balbina Casas-Méndez. *Two-stage heuristic algorithm for a new variant of the multi-compartment vehicle routing problem with stochastic demands*. 2024. arXiv: 2410.17302 [math.OC]. URL: https://arxiv.org/abs/2410.17302.

[Han+17] Pierre Hansen et al. “Variable neighborhood search: basics and variants”. In: *EURO Journal on Computational Optimization* 5.3 (2017), pp. 423–454. ISSN: 2192-4406. DOI: https://doi.org/10.1007/s13675-016-0075-x. URL: https://www.sciencedirect.com/science/article/pii/S2192440621000873.

[Kan+24] Ravi Kant et al. *Variable Neighborhood Search for the Multi-Depot Multiple Set Orienteering Problem*. 2024. arXiv: 2408.08922 [math.OC]. URL: https://arxiv.org/abs/2408.08922.

[Liu+24] Huan Liu et al. “Multi-constraint distributed terminal distribution path planning for fresh agricultural products”. In: *Applied Intelligence* 55.3 (Dec. 2024). ISSN: 0924-669X. DOI: 10.1007/s10489-024-06076-8. URL: https://doi.org/10.1007/s10489-024-06076-8.

[LNA84] Gilbert Laporte, Y Nobert, and D Arpin. *Optimal Solutions to Capacitated Multi-depot Vehicle Routing Problems*. Les Cahiers du GERAD G-84-02. GERAD, Montréal QC H3T 2A7, Canada: Groupe d’études et de recherche en analyse des décisions, Feb. 1984, pp. 1–10. URL: https://www.gerad.ca/en/papers/G-84-02. published.

[Mac+21] André Manhães Machado et al. “A new hybrid matheuristic of GRASP and VNS based on constructive heuristics, set-covering and set-partitioning formulations applied to the capacitated vehicle routing problem”. In: *Expert Systems with Applications* 184 (2021), p. 115556. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115556. URL: https://www.sciencedirect.com/science/article/pii/S0957417421009623.

[MH97] N. Mladenović and P. Hansen. “Variable neighborhood search”. In: *Computers & Operations Research* 24.11 (1997), pp. 1097–1100. ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(97)00031-2. URL: https://www.sciencedirect.com/science/article/pii/S0305054897000312.

[Min89] Hokey Min. “The multiple vehicle routing problem with simultaneous delivery and pick-up points”. In: *Transportation Research Part A: General* 23.5 (1989), pp. 377–386. ISSN: 0191-2607. DOI: https://doi.org/10.1016/0191-2607(89)90085-X. URL: https://www.sciencedirect.com/science/article/pii/019126078990085X.

[Moh+23] Mostafa Mohammadi et al. “A dynamic approach for the multi-compartment vehicle routing problem in waste management”. In: *Renewable and Sustainable Energy Reviews* 184 (2023), p. 113526. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2023.113526. URL: https://www.sciencedirect.com/science/article/pii/S1364032123003830.

[Mon+15] Jairo R. Montoya-Torres et al. “A literature review on the vehicle routing problem with multiple depots”. In: *Computers & Industrial Engineering* 79 (2015), pp. 115–129. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2014.10.029. URL: https://www.sciencedirect.com/science/article/pii/S036083521400360X.

[ON24] Nasreddine Ouertani and Ahmed Nait Sidi Moh. “Towards Eco-Friendly Multi-compartment Transportation: Anbsp;New Bi-objective Iterated Local Search Framework”. In: *Advances and Trends in Artificial Intelligence. Theory and Applications: 37th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2024, Hradec Kralove, Czech Republic, July 10–12, 2024, Proceedings*. Hradec Kralove, Czech Republic: Springer-Verlag, 2024, pp. 388–400. ISBN: 978-981-97-4676-7. DOI: 10.1007/978-981-97-4677-4_32. URL: https://doi.org/10.1007/978-981-97-4677-4_32.

[Ost+21] Manuel Ostermeier et al. “Multi-compartment vehicle routing problems: State-of-the-art, modeling framework and future directions”. In: *European Journal of Operational Research* 292.3 (2021), pp. 799–817. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2020.11.009. URL: https://www.sciencedirect.com/science/article/pii/S0377221720309553.

[SCK23] Ville P. Saarinen, Ted Hsuan Yun Chen, and Mikko Kivelä. *OVNS: Opportunistic Variable Neighborhood Search for Heaviest Subgraph Problem in Social Networks*. 2023. arXiv: 2305.19729 [cs.SI]. URL: https://arxiv.org/abs/2305.19729.

[Sol87] Marius M. Solomon. “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints”. In: *Operations Research* 35.2 (1987), pp. 254–265. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/170697 (visited on 06/05/2025).

[Sou+22] Israel Pereira de Souza et al. “A Reactive GRASP Algorithm fornbsp;thenbsp;Multi-depot Vehicle Routing Problem”. In: *Computational Science and Its Applications – ICCSA 2022 Workshops: Malaga, Spain, July 4–7, 2022, Proceedings, Part II*. Malaga, Spain: Springer-Verlag, 2022, pp. 81–96. ISBN: 978-3-031-10561-6. DOI: 10.1007/978-3-031-10562-3_7. URL: https://doi.org/10.1007/978-3-031-10562-3_7.

[SZZ24] Yukang Su, Shuo Zhang, and Chengning Zhang. “A lightweight genetic algorithm with variable neighborhood search for multi-depot vehicle routing problem with time windows”. In: *Applied Soft Computing* 161 (May 2024), p. 111789. DOI: 10.1016/j.asoc.2024.111789.

[WQ24] Yu WU and Xiaoping Qiu. “GRASP-based request allocation and MDP-based vehicle routing in home pick-up service with service continuity consideration”. In: *IET Intelligent Transport Systems* 18.2 (2024), pp. 332–345. DOI: https://doi.org/10.1049/itr2.12454. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/itr2.12454. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/itr2.12454.

9 Appendix

The following tables show the results obtained by the GRASP-VNS hybrid Algorithm. Each of these values are the minimum values obtained after five runs in each test instance.

9.1 First Demand Variation

Improvement run setting: 50;

Demand Distribution 30-70, 10-60:

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	674,03	16	1,32	665,26	15	0,00	676,31	16	1,66
p02	526,50	7	2,22	515,09	6	0,00	532,62	7	3,40
p03	678,87	11	0,00	685,44	11	0,97	725,64	11	6,89
p04	1172,71	18	2,75	1141,34	17	0,00	1167,38	17	2,28
p05	879,38	8	4,85	838,68	8	0,00	844,39	8	0,68
p06	1034,38	18	3,59	998,51	17	0,00	1006,02	17	0,75
p07	1017,85	18	1,25	1005,33	17	0,00	1039,91	18	3,44
p08	5995,98	36	0,58	5961,68	35	0,00	6255,50	36	4,93
p09	5230,78	35	2,60	5098,35	34	0,00	5331,55	34	4,57
p10	4825,72	34	0,00	4831,94	35	0,13	4935,36	34	2,27
p11	4144,82	28	0,00	4284,06	28	3,36	4500,22	29	8,57
p12	1542,92	10	12,14	1375,88	9	0,00	1457,36	9	5,92
p13	1442,53	10	0,00	1479,66	10	2,57	1623,59	10	12,55
p14	1683,90	10	6,73	1577,71	10	0,00	1649,23	11	4,53
p15	2412,95	6	0,00	2546,61	6	5,54	2728,03	6	13,06
p16	3040,64	19	0,00	3269,41	21	7,52	3575,60	24	17,59
p17	3227,14	21	0,00	3455,68	21	7,08	3694,18	25	14,47
p18	4469,85	28	0,00	4801,56	27	7,42	4963,91	25	11,05
p19	4619,99	29	0,00	5025,18	32	8,77	5642,54	37	22,13
p20	4962,62	32	0,00	5266,76	35	6,13	5752,05	36	15,91
p21	6843,84	39	0,00	7411,10	39	8,29	7883,06	38	15,18
p22	7020,54	44	0,00	4925,40	49	12,89	8563,07	52	21,97
p23	7642,11	51	0,00	8284,75	55	8,41	9016,86	56	17,99
Average	3264,78	22,96	1,65	3410,67	23,35	3,44	3633,23	24,17	9,21

Table 11: Demand Distribution 30-70, 10-60

Improvement run setting: 100;

Demand Distribution 30-70, 10-60:

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	669,74	15	1,32	661,02	15	0,00	671,57	15	1,60
p02	541,27	7	0,65	537,79	6	0,00	541,07	7	0,61
p03	678,84	11	0,00	707,74	11	4,26	712,12	11	4,90
p04	1175,25	18	2,60	1145,48	17	0,00	1156,68	18	0,98
p05	875,55	8	3,59	845,22	8	0,00	850,04	8	0,57
p06	1034,84	18	3,11	1003,64	17	0,00	1017,62	17	1,39
p07	1023,12	18	1,60	1007,00	18	0,00	1044,93	17	3,77
p08	6003,62	36	0,14	5995,28	36	0,00	6182,70	36	3,13
p09	5282,81	35	3,10	5124,02	34	0,00	5305,34	35	3,54
p10	4837,07	34	0,23	4826,09	35	0,00	5021,43	35	4,05
p11	4217,94	28	0,00	4283,09	28	1,54	4508,90	29	6,90
p12	1542,49	9	11,68	1381,18	9	0,00	1447,51	8	4,80
p13	1448,39	10	0,00	1491,63	11	2,99	1656,21	11	14,35
p14	1670,46	11	1,61	1643,96	11	0,00	1697,79	11	3,27
p15	2434,26	7	0,00	2502,92	6	2,82	2736,86	6	12,43
p16	3027,73	19	0,00	3285,17	22	8,50	3546,09	22	17,12
p17	3216,50	21	0,00	3518,28	23	9,38	3662,31	23	13,86
p18	4456,46	26	0,00	4756,25	25	6,73	4985,01	26	11,86
p19	4639,97	31	0,00	5106,34	31	10,05	5583,32	35	20,33
p20	5010,54	32	0,00	5351,09	35	6,80	5633,92	35	12,44
p21	6834,13	38	0,00	7592,25	39	11,09	7865,53	41	15,09
p22	7086,20	46	0,00	7837,56	49	10,60	8487,26	55	19,77
p23	7562,93	52	0,00	8380,81	52	10,81	8986,75	60	18,83
Average	3272,61	23,04	1,29	3434,08	23,39	3,72	3621,78	24,39	8,50

Table 12: Demand Distribution 30-70, 10-60

Improvement run setting: 150;

Demand Distribution 30-70, 10-60:

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	684.19	15	1.98	670.91	15	0.00	680.73	15	1.46
p02	549.18	7	7.88	509.07	7	0.00	533.97	6	4.89
p03	692.28	11	0.00	716.59	11	3.51	725.29	11	4.77
p04	1177.35	18	0.64	1169.88	17	0.00	1174.66	17	0.41
p05	875.35	8	3.98	841.88	8	0.00	842.84	8	0.11
p06	1037.65	18	1.77	1019.61	17	0.00	1049.02	17	2.88
p07	1023.10	18	1.14	1011.55	17	0.00	1031.33	17	1.96
p08	5955.49	35	0.00	5991.31	36	0.60	6279.97	37	5.45
p09	5239.03	36	1.18	5177.71	35	0.00	5316.81	35	2.69
p10	4828.65	35	0.00	4863.95	35	0.73	4994.08	35	3.43
p11	4174.21	27	0.00	4277.66	29	2.48	4527.89	28	8.47
p12	1530.59	9	6.75	1433.87	9	0.00	1455.25	8	1.49
p13	1453.08	10	0.00	1551.97	11	6.81	1686.78	11	16.08
p14	1726.19	11	7.68	1603.05	10	0.00	1701.84	11	6.16
p15	2424.94	6	0.00	2591.14	6	6.85	2714.26	6	11.93
p16	2981.30	19	0.00	3372.07	23	13.11	3596.17	22	20.62
p17	3251.65	21	0.00	3468.37	23	6.66	3700.48	24	13.80
p18	4436.67	27	0.00	4715.14	26	6.28	4833.09	25	8.94
p19	4704.81	29	0.00	5157.05	32	9.61	5585.70	33	18.72
p20	5020.37	33	0.00	5483.74	36	9.23	5827.35	37	16.07
p21	6988.80	38	0.00	7416.49	39	6.12	7867.85	37	12.58
p22	7142.19	48	0.00	8010.13	50	12.15	8658.39	54	21.23
p23	7689.98	51	0.00	8388.13	53	9.08	8965.01	56	16.58
Average	3286.39	23.04	1.43	3453.97	23.70	4.05	3641.25	23.91	8.73

Table 13: Demand Distribution 30-70, 10-60

9.2 Second Demand Variation

Improvement run setting: 50;

Demand Distribution 30-70, 20-60:

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	652,82	12	3,89	628,37	12	0,00	647,84	13	3,10
p02	525,27	6	3,93	505,42	6	0,00	514,42	6	1,78
p03	709,77	12	0,00	710,24	12	0,07	728,19	12	2,60
p04	1172,02	18	4,06	1126,26	17	0,00	1150,94	17	2,19
p05	904,72	9	9,01	829,96	8	0,00	847,21	8	2,08
p06	1009,43	17	2,09	988,72	17	0,00	1022,61	18	3,43
p07	1011,55	18	0,00	1014,11	17	0,25	1031,66	18	1,99
p08	5754,55	33	0,00	5769,66	33	0,26	5953,46	33	3,46
p09	4981,04	32	0,14	4974,23	32	0,00	5128,31	32	3,10
p10	4617,06	33	0,00	4674,04	33	1,23	4863,76	32	5,34
p11	4202,77	29	0,00	4298,39	28	2,28	4440,60	28	5,66
p12	1364,16	8	0,00	1412,20	8	3,52	1463,46	8	7,28
p13	1445,80	10	0,00	1523,66	10	5,39	1658,39	10	14,70
p14	1708,17	11	9,30	1562,89	10	0,00	1675,05	11	7,18
p15	2405,67	6	0,00	2594,90	6	7,87	2680,57	6	11,43
p16	2994,38	19	0,00	3332,70	22	11,30	3515,64	21	17,41
p17	3177,67	21	0,00	3451,41	22	8,61	3711,05	23	16,79
p18	4430,64	27	0,00	4653,96	25	5,04	5016,98	25	13,23
p19	4701,96	30	0,00	5034,57	30	7,07	5556,34	35	18,17
p20	4875,89	31	0,00	5409,39	36	10,94	5704,57	35	17,00
p21	7063,08	39	0,00	7427,89	38	5,17	7895,28	38	11,78
p22	7100,97	46	0,00	8060,93	47	13,52	8381,73	55	18,04
p23	7584,26	50	0,00	8289,28	54	9,30	8835,46	56	16,50
Average	3234,51	22,48	1,41	3403,18	22,74	3,99	3583,63	23,48	8,88

Table 14: Demand Distribution 30-70, 20-60

Improvement run setting: 100;

Demand Distribution 30-70, 20-60:

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	652,73	13	1,51	643,73	12	0,00	647,36	12	0,67
p02	533,35	6	5,57	505,23	6	0,00	508,91	6	0,73
p02	718,13	12	0,00	733,36	12	0,74	742,13	12	3,34
p04	1171,92	18	1,57	1153,82	17	0,00	1168,10	18	1,24
p05	906,51	9	10,50	820,39	9	0,00	857,72	9	4,55
p06	1019,52	18	1,49	1004,51	17	0,00	1035,64	18	3,10
p07	1022,87	18	1,26	1010,18	17	0,00	1045,70	18	3,52
p08	5726,39	33	0,00	5874,50	34	2,59	6053,01	34	5,70
p09	5023,74	33	0,23	5012,41	32	0,00	5167,61	32	3,10
p10	4665,14	33	0,00	4736,72	33	1,53	4887,28	33	4,76
p11	4220,08	28	0,00	4267,00	28	1,11	4521,35	29	7,14
p12	1404,16	8	1,05	1389,52	8	0,00	1431,57	8	3,03
p13	1480,75	10	0,00	1526,62	10	3,10	1699,50	12	14,77
p14	1689,83	11	3,58	1631,40	11	0,00	1777,00	12	8,92
p15	2371,38	6	0,00	2565,36	6	8,18	2694,52	6	13,63
p16	3024,14	20	0,00	3271,38	22	8,18	3632,95	23	20,13
p17	3212,25	21	0,00	3502,42	24	9,03	3755,70	23	16,92
p18	4508,69	28	0,00	4791,15	25	6,26	5091,63	26	12,93
p19	4685,35	30	0,00	5158,85	31	10,11	5504,38	33	17,48
p20	5057,08	33	0,00	5466,81	37	8,10	5770,49	36	14,11
p21	7125,98	40	0,00	7524,80	39	5,60	7912,96	38	11,04
p22	7188,76	47	0,00	7952,42	49	10,62	8668,34	52	20,58
p23	7627,28	50	0,00	8299,99	54	8,82	8987,37	62	17,83
Average	3262,44	22,83	1,16	3427,48	23,17	3,65	3633,10	24,00	9,10

Table 15: Demand Distribution 30-70, 20-60

Improvement run setting: 150;

Demand Distribution 30-70, 20-60:

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	653.57	12	2.23	639.29	12	0.00	648.72	12	1.48
p02	537.81	6	8.79	494.36	6	0.00	501.58	6	1.46
p03	721.44	12	0.00	730.28	12	1.23	745.20	11	3.29
p04	1171.96	18	1.24	1157.68	17	0.01	1157.58	17	0.00
p05	905.05	9	8.25	836.04	8	0.00	840.89	8	0.58
p06	1009.43	17	0.17	1007.74	17	0.00	1019.45	18	1.16
p07	1021.17	18	0.84	1012.66	17	0.00	1036.24	18	2.33
p08	5711.93	33	0.00	5857.85	34	2.55	6041.13	34	5.76
p09	5001.80	32	0.00	5031.63	33	0.60	5111.73	32	2.20
p10	4602.42	33	0.00	4723.71	33	2.64	4850.90	33	5.40
p11	4232.01	28	0.00	4319.44	29	2.07	4493.58	28	6.18
p12	1404.16	8	1.54	1382.83	9	0.00	1508.13	8	9.06
p13	1435.60	9	0.00	1515.77	10	5.58	1643.65	11	14.49
p14	1765.16	12	9.15	1617.23	10	0.00	1647.42	10	1.87
p15	2371.38	6	0.00	2603.33	7	9.78	2716.32	6	14.55
p16	2986.32	19	0.00	3331.93	19	11.57	3540.69	22	18.56
p17	3310.42	21	0.00	3546.90	24	7.14	3705.69	23	11.94
p18	4489.98	26	0.00	4820.66	26	7.36	4774.08	25	6.33
p19	4668.19	29	0.00	5197.19	32	11.33	5462.82	35	17.02
p20	5072.56	32	0.00	5364.72	35	5.76	5734.42	35	13.05
p21	7168.88	39	0.00	7470.31	38	4.20	7756.44	38	8.20
p22	7191.56	46	0.00	8050.43	50	11.94	8749.15	53	21.66
p23	7672.62	50	0.00	8332.84	52	8.60	8783.10	54	14.47
Average	3265.45	22.39	1.40	3436.73	23.04	4.02	3585.60	23.35	7.87

Table 16: Demand Distribution 30-70, 20-60

9.3 Third Demand Variation

Improvement run setting: 50;

Demand Distribution 35-65, 25-65: Improvement run setting: 100;

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	669,35	14	2,07	655,77	14	0,00	661,56	13	0,88
p02	549,81	6	10,12	499,26	6	0,00	516,96	6	3,55
p03	701,71	11	0,00	708,96	11	1,03	721,35	12	2,80
p04	1130,69	17	0,91	1120,48	17	0,00	1168,72	17	4,31
p05	868,42	8	6,57	814,87	8	0,00	847,97	8	4,06
p06	988,43	17	0,10	987,41	17	0,00	1014,75	17	2,77
p07	1021,13	18	0,73	1013,77	17	0,00	1015,21	17	0,14
p08	6161,44	37	0,92	6105,57	36	0,00	6356,85	37	4,12
p09	5251,38	35	1,99	5149,02	35	0,00	5376,84	35	4,42
p10	4826,97	35	1,33	4763,51	35	0,00	5084,99	36	6,75
p11	4168,57	28	0,00	4312,50	28	3,45	4478,52	29	7,44
p12	1468,84	9	3,08	1424,92	8	0,00	1450,98	8	1,83
p13	1448,39	10	0,00	1518,73	10	4,86	1646,03	12	13,65
p14	1709,71	11	12,63	1517,95	10	0,00	1674,75	11	10,33
p15	2412,79	6	0,00	2538,20	6	5,20	2686,53	6	11,35
p16	2998,46	19	0,00	3312,95	22	10,49	3537,26	22	17,97
p17	3226,14	21	0,00	3438,01	22	6,57	3695,77	24	14,56
p18	4421,56	25	0,00	4755,64	26	7,56	4951,68	25	11,99
p19	4646,12	29	0,00	5077,85	31	9,29	5558,31	33	19,63
p20	5001,14	32	0,00	5364,16	34	7,26	5796,62	39	15,91
p21	6951,91	39	0,00	7259,87	37	4,43	7735,80	38	11,28
p22	7083,12	47	0,00	7935,66	49	12,04	8499,50	49	20,00
p23	7583,74	52	0,00	8364,12	55	10,29	8840,38	54	16,57
Average	3273,47	22,87	1,76	3419,09	23,22	3,59	3622,49	23,83	8,97

Table 17: Demand Distribution 35-65, 25-65

Demand Distribution 35-65, 25-65: Improvement run setting: 150;

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	679.64	14	3.03	659.67	14	0.00	671.40	13	1.78
p02	543.85	6	8.05	503.31	6	0.00	505.60	6	0.45
p03	694.10	11	0.00	703.89	11	1.41	734.65	11	5.84
p04	1130.70	17	0.04	1130.26	17	0.00	1153.77	17	2.08
p05	876.27	8	6.05	826.29	8	0.00	848.92	8	2.74
p06	988.20	17	0.00	991.20	17	0.30	1009.27	17	2.13
p07	1029.56	18	1.35	1015.80	17	0.00	1033.28	18	1.72
p08	6188.35	37	0.00	6224.62	36	0.59	6373.75	37	3.00
p09	5262.71	35	3.59	5080.36	35	0.00	5430.69	34	6.90
p10	4842.71	35	0.00	4893.89	36	1.06	5115.47	36	5.63
p11	4197.45	28	0.00	4310.31	28	2.69	4499.91	28	7.21
p12	1503.47	9	10.67	1358.50	8	0.00	1438.17	8	5.86
p13	1438.94	10	0.00	1517.92	10	5.49	1669.86	11	16.05
p14	1744.61	13	9.75	1589.64	11	0.00	1673.88	11	5.30
p15	2425.49	6	0.00	2546.54	6	4.99	2722.25	6	12.24
p16	3035.35	19	0.00	3324.02	21	9.51	3422.00	20	12.74
p17	3248.48	21	0.00	3467.89	22	6.75	3746.10	24	15.32
p18	4411.24	26	0.00	4868.89	25	10.37	5003.01	24	13.42
p19	4675.11	30	0.00	5146.57	30	10.08	5627.42	34	20.37
p20	4977.13	33	0.00	5530.08	37	11.11	5756.09	37	15.65
p21	6865.47	37	0.00	7459.16	37	8.65	7923.05	38	15.40
p22	7085.21	46	0.00	7979.14	48	12.62	8575.05	54	21.03
p23	7665.70	52	0.00	8382.78	53	9.35	8866.65	56	15.67
p24	3283.03	22.96	1.85	3456.99	23.17	4.13	3643.49	23.83	9.07
Average	3283.03	22.96	1.85	3456.99	23.17	4.13	3643.49	23.83	9.07

Table 18: Demand Distribution 35-65, 25-65

Demand Distribution 35-65, 25-65:

Instance	BO1	Veh	%	BO2	Veh	%	BO3	Veh	%
p01	676.95	14	2.51	666.47	13	0.92	660.39	13	0.00
p02	553.14	6	9.33	505.93	6	0.00	511.36	6	1.07
p03	686.00	11	0.00	708.85	11	3.33	725.75	12	5.79
p04	1134.87	17	0.34	1131.03	17	0.00	1157.16	17	2.31
p05	874.73	8	5.00	833.07	8	0.00	850.51	8	2.09
p06	988.20	17	0.00	989.61	17	0.14	1012.16	17	2.42
p07	1027.60	18	1.25	1014.91	17	0.00	1033.60	17	1.84
p08	6170.80	37	0.00	6242.71	37	1.17	6421.78	37	4.07
p09	5292.00	35	1.67	5205.19	35	0.00	5373.22	35	3.23
p10	4877.58	35	1.16	4821.78	35	0.00	5139.65	37	6.59
p11	4176.87	28	0.00	4201.51	28	0.59	4518.55	28	8.18
p12	1465.54	9	6.71	1373.39	9	0.00	1463.47	9	6.56
p13	1448.39	10	0.00	1554.20	11	7.31	1655.21	11	14.28
p14	1713.24	11	7.00	1601.17	11	0.00	1753.77	12	9.53
p15	2407.13	6	0.00	2650.75	6	10.12	2771.86	6	15.15
p16	3029.13	20	0.00	3236.68	20	6.85	3472.61	21	14.64
p17	3163.28	20	0.00	3488.74	24	10.29	3672.52	24	16.10
p18	4473.27	26	0.00	4762.28	26	6.46	4987.50	25	11.50
p19	4680.42	29	0.00	5170.65	32	10.47	5582.27	35	19.27
p20	5025.40	32	0.00	5538.94	35	10.22	5827.36	37	15.96
p21	7009.19	41	0.00	7301.99	39	4.18	7895.48	38	12.64
p22	7143.97	47	0.00	7995.41	49	11.92	8542.99	53	19.58
p23	7671.30	49	0.00	8407.85	54	9.60	8861.06	58	15.51
Average	3290.83	22.87	1.52	3452.31	23.48	4.07	3647.40	24.17	9.06

Table 19: Demand Distribution 35-65, 25-65