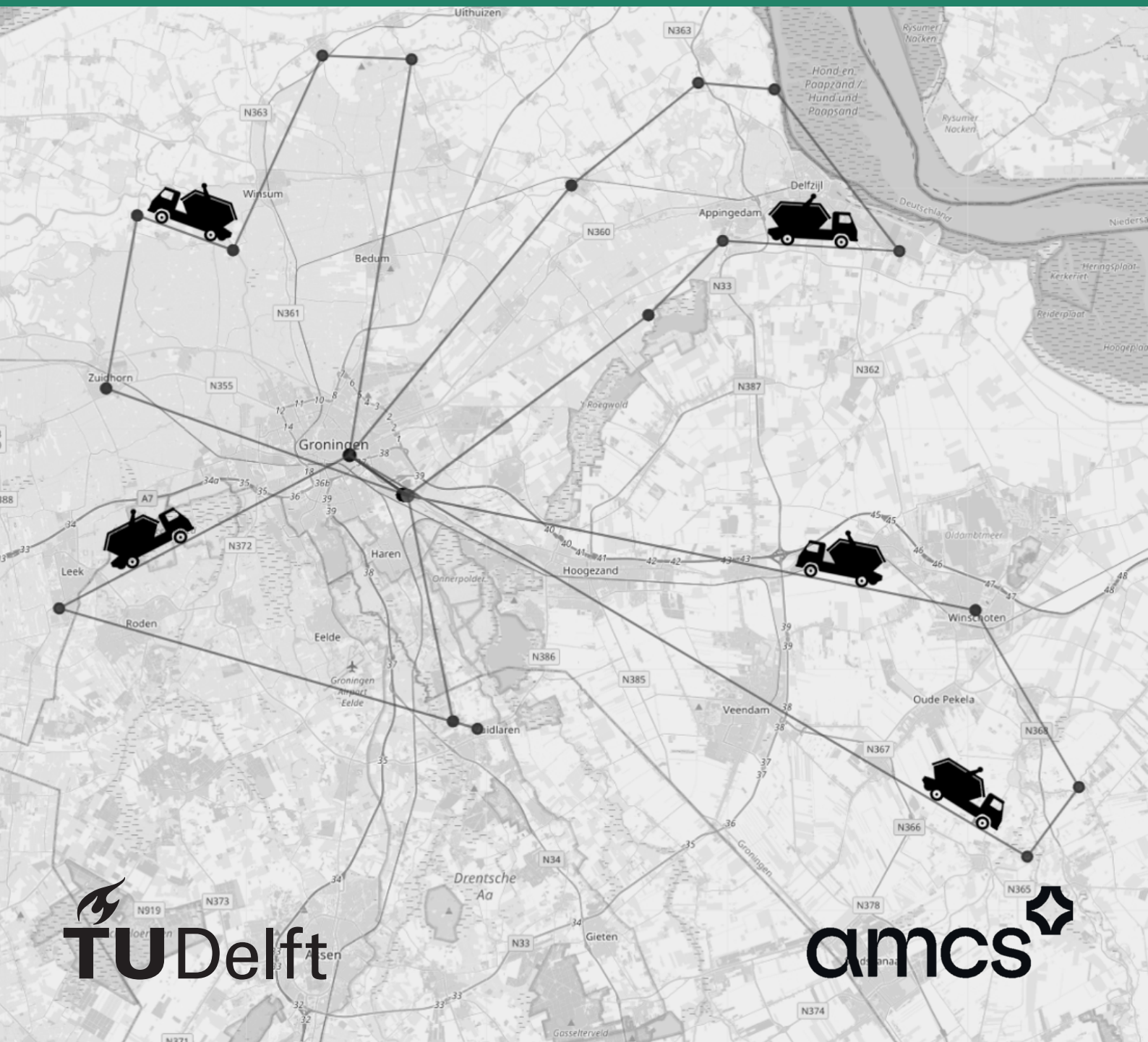
Optimising Vehicle Routing for Skip Containers

A Waste Collection Case Study with Container Reuse and Stacking using LNS

MSc Transport, Infrastructure and Logistics

Caroline Aalders



TU Delft

amcs

Optimising Vehicle Routing for Skip Containers

A Waste Collection Case Study with Container Reuse and Stacking using LNS

by

Caroline Aalders

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday May 29, 2026 at 14:00.

Student number:	5230578	
Project Duration:	January 13, 2026 - May 29, 2026	
Thesis committee:	Dr. B. Atasoy,	TU Delft, chair & supervisor
	Dr. S. Fazi	TU Delft, supervisor
	Dr. J. Duran Micco	TU Delft, supervisor
	K.M. Hauge	AMCS Group, supervisor
	K. Van Duurling	AMCS Group, supervisor

Cover:	Map visualisation created by the author; truck figure adapted from M&K (2026)
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft amcs

Preface

This thesis focuses on skip container routing for waste collection. This study introduced more detailed possibilities for stacking and reuse into the scope. It also resulted in me noticing these types of skip containers everywhere in the city, which confirmed that it is a practically relevant topic to research. With this thesis, I am completing the final step of my master's programme in Transport, Infrastructure and Logistics at Delft University of Technology.

I would like to thank my supervisors from TU Delft, Bilge Atasoy, Stefano Fazi and Javier Duran Micco, for their time and input over the past six months. I would also like to thank them for the flexibility of postponing my kick-off so that I could explore South America before the real work started. Furthermore, I would also like to thank Kristian Hauge and Koen van Duurling from AMCS for their time and interesting conversations in support of completing my thesis work.

I have really enjoyed my time studying here in Delft over the past six years. The friends I made during these years made my student life much more fun, whether through all the different committees, my board year, my exchange in Canada, or the many hours spent playing tennis with my team. During my thesis as well, the time spent together in the thesis room made the whole process far more enjoyable. So, thank you all for making these years such a great time in my life and for giving me so many memories that I can look back on with a smile.

Caroline Aalders
Delft, May 2026

Summary

Growing volumes of construction and demolition waste are creating increasing pressure on waste logistics systems. Skip containers are one of the main container types used to transport this type of waste. They are placed at customer sites, collected when full, transported to disposal facilities, and temporarily stored at storage locations. In practice, routing these containers is operationally complex. Vehicles must coordinate deliveries, pickups, disposal trips, and storage movements while accounting for handling times, loading restrictions, and daily operational limits. Existing research has recognised skip container routing as a complex variant of the Vehicle Routing Problem (VRP), but most studies focus on large rollon-rolloff (RoRo) containers. As a result, the operational characteristics of smaller chain-lift skip containers remain underexplored.

This thesis focuses on chain-lift skip containers, which differ from traditional RoRo containers because they can be stacked on a vehicle. This allows multiple containers to be transported simultaneously, but also introduces additional loading constraints related to stacking feasibility and container compatibility. In addition, empty containers can be reused directly between customers instead of first returning to a storage location. Although both stacking and direct reuse are relevant in practice, they have received limited attention in the literature. Therefore, the main research question of this thesis is:

How can vehicle routing for skip container waste collection be optimised under container reuse and stacking feasibility constraints?

To answer this question, the study first reviewed the literature on skip container routing, order types, vehicle routing problems, and solution methods. The review showed that previous research has gradually incorporated more realistic operational features, such as multiple depots, multiple disposal facilities, time windows, and different container types. However, detailed stacking constraints and direct container reuse remain largely unexplored. The review also confirmed that exact optimisation methods become less practical as routing problems increase in size and complexity, while metaheuristics are generally more suitable for larger operational scenarios.

Based on these insights, two models were developed. First, a mixed-integer linear programming (MILP) optimisation model was formulated as a benchmark representation of standard skip container operations, excluding container reuse and using simplified stacking constraints. This model provides high-quality solutions for smaller problem scenarios and serves as a reference for evaluating the extended model. Second, an extended Large Neighbourhood Search (LNS) metaheuristic was developed to explicitly incorporate detailed stacking constraints, direct container reuse, and multiple disposal and storage location options. After a feasible initial solution is found, the LNS repeatedly removes parts of the solution and reconstructs them to search for improvements.

Both models were tested on data sets of varying sizes and different mixes of order types, including container pickups, deliveries, change, and swap orders. Their performance was evaluated using key performance indicators (KPIs), including total operational time, number of vehicles used, orders per hour, vehicle utilisation, stacking utilisation, and computational runtime. The benchmark results showed that operational time increased gradually as more customers were added, whereas computational time increased exponentially. This confirms that exact optimisation quickly becomes computationally impractical for larger problem cases.

The validation experiments demonstrated that the LNS algorithm produced solutions close to the

benchmark model within substantially shorter runtimes, with an average deviation below 3% for the smaller validation cases and below 5% for larger cases. In some scenarios, the LNS even outperformed the benchmark model by successfully applying container reuse opportunities. Across the larger experiments, vehicle configuration proved to have a significant influence on solution quality. Vehicles were tested across different loading configurations, varying the number of stacking positions available for full versus empty containers. Configurations with limited capacity for transporting full containers consistently performed worst, indicating that full container capacity forms a main operational bottleneck. Once sufficient full container capacity was available, performance differences between configurations became considerably smaller. For several larger data sets, improved configurations also reduced the number of vehicles required.

The experiments also showed that the mix of order types influences routing performance. Data sets with a high combined share of swap and change orders were more difficult to optimise and showed greater sensitivity to vehicle configuration. Swap orders involve the most routing activities, while change orders, which require a container to be returned to the same customer after emptying rather than to a shared storage location, impose additional sequencing constraints that reduce routing flexibility. Data sets dominated by pickup and delivery orders offered more freedom to consolidate stops. Furthermore, reuse consistently reduced operational time, with savings of 34–108 minutes across test cases. This improvement came primarily from reduced handling time rather than a reduction in route stops, and the number of vehicles required remained unchanged. The stacking analysis further showed that supporting two stacking rows already captures most of the operational benefit, with additional rows mainly providing flexibility during temporary peak loading moments rather than being used consistently throughout routes. The sensitivity analyses confirmed that the models behave logically: lower service times improved the main KPIs, whereas higher service times increased route duration and reduced productivity.

Overall, this thesis demonstrates that detailed stacking constraints and direct container reuse can be successfully incorporated into skip container routing models. Scientifically, the research extends the existing skip container routing literature towards a more realistic representation of chain-lift skip operations. Practically, it provides AMCS with insight into how reuse opportunities, stacking feasibility, and vehicle loading capacity influence routing performance. The results indicate that supporting stacking across two rows already yields substantial operational improvements, while larger stacking heights offer limited additional benefits on short sections of the routes. Future research could further extend the scope by incorporating additional operational constraints, such as time windows, driver rest periods, stochastic travel times, and limited container availability. In addition, the current metaheuristic framework could be extended into an Adaptive Large Neighbourhood Search (ALNS) approach in which operators are selected adaptively based on their recent performance, offering potential for further improvements in practical routing applications.

Contents

List of Figures

List of Tables

1

Introduction

Skip containers are waste containers used mainly in construction and demolition to collect debris from customer sites (Wøhlk & Laporte, 2022). These containers are placed at construction sites and periodically transported to disposal facilities when they are full. Within the European Union (EU), construction and demolition waste is the largest waste category among waste generated by economic activities and household waste, accounting for approximately 38% of the total waste generated in 2022 (Eurostat, 2024). On a global scale, construction waste accounts for an estimated 25-40% of total solid waste generation (Sakthibala et al., 2025), and this share is expected to grow at an annual rate of 3–5% due to increasing urbanisation, infrastructure development, and population growth.

Given this substantial and growing volume, the collection and transportation of construction waste forms a significant component of urban waste logistics. Unlike residential and commercial waste collection, where waste is emptied directly into the collection vehicle, skip container operations involve transporting containers as separate units. In practice, skip containers include several types. Most routing studies focus on large rollon-rolloff (RoRo) containers, whereas this research considers smaller stackable chain-lift skips. Regardless of the type, each skip must be delivered, collected, and transported to a disposal facility before further service can be performed. As a result, skip container operations contribute considerably to vehicle kilometres travelled, operational costs, and emissions related to waste collection logistics. Therefore, a specific vehicle routing problem (VRP) focusing on skip container routing has been developed.

In the routing literature, because the focus is on RoRo skip container, these models typically assume that a vehicle can transport only one container at a time, or at most two (Archetti & Speranza, 2004; Bodin et al., 2000; Golden et al., 2002). In contrast, this research focuses on a chain-lift skip that can be stacked on a vehicle, allowing multiple containers to be transported simultaneously. Their stackability introduces additional operational decisions regarding container organisation, compatibility, and loading feasibility. These aspects are largely ignored in classical RoRo VRPs, but they directly impact vehicle utilisation and routing efficiency.

In addition to stacking, skip container operations may allow direct reuse of empty containers because the containers are owned by the same company. Instead of returning an empty container to a depot, an empty skip can be immediately reused to serve another customer who requires a compatible container. This reuse option reduces unnecessary depot visits and improves vehicle utilisation, but introduces additional dependencies between orders, as the availability of an empty container becomes directly linked to the completion of preceding orders. Although reuse

has been considered in a limited number of skip container routing studies (Archetti & Speranza, 2004; Hauge et al., 2014), it is not commonly integrated alongside detailed stacking considerations.

Furthermore, early studies considered simplified problem variants and therefore relied on exact solution methods (Archetti et al., 2005; Baldacci et al., 2006). More recent research has extended these models by incorporating additional features such as container storage yards, multiple container types, and flexible return options, thus moving closer to real operational settings (Li et al., 2018; Raucq et al., 2019; Wøhlk & Laporte, 2022). Despite these extensions, detailed constraints on container stacking and reuse remain largely unexplored in waste collection routing models, whereas similar aspects have been applied in other sectors, such as container drayage and maritime container terminals.

Incorporating detailed container stacking constraints and direct container reuse into a skip container VRP significantly increases its computational complexity. Even simplified variants of skip container VRPs have been shown to be NP-hard (Archetti et al., 2005), making exact solution methods infeasible for larger problems. This is also reflected in the broader VRP literature, where the majority of studies rely on metaheuristic solution approaches rather than exact optimisation methods, such as mixed-integer linear programming (MILP) models. In particular, neighbourhood-based metaheuristics are the most widely applied class of methods to solve complex VRPs (Braekers et al., 2016; Elshaer & Awad, 2020).

In this research, an exact optimisation approach is first used to construct a simplified benchmark model that provides a reference solution. Following this, a metaheuristic algorithm based on large neighbourhood search (LNS) is developed to incorporate stacking and reuse. LNS has proven effective in handling routing problems with sequenced and interdependent decisions and has been successfully applied in earlier skip container routing (Pisinger & Ropke, 2019; Wy et al., 2013).

1.1. Research Questions

The aim of this research is to explore vehicle routing for skip waste containers, incorporating additional constraints on container reuse and more detailed stacking. The main research question is presented in Table 1.1, together with the sub-question and the methods used to answer them.

Table 1.1: Research questions and methods

Main research question	
How can vehicle routing for skip container waste collection be optimised under container reuse and stacking feasibility constraints?	
Sub-questions	**Method**
What is the current state of skip container routing in the literature?	Literature review
Which key performance indicators are suitable for evaluating and comparing routing models?	Literature review
How can a mixed-integer linear programming model be formulated to represent standard skip container operations?	Mathematical modelling
How can a metaheuristic conceptually be designed to handle container reuse and stacking feasibility?	Conceptual design
How does the metaheuristic model perform relative to the mixed-integer linear programming model in terms of the defined key performance indicators?	Case Study Analysis

1.2. Outline

This thesis is organised as follows. Chapter 2 reviews the relevant literature. Chapter 3 defines the scope. Chapter 4 presents the methodology and KPIs. Chapter 5 describes the experimental setup and parameter tuning. Chapter 6 discusses the computational results. Finally, Chapter 7 concludes the thesis and suggests future research.

2

Literature Review

This literature review will provide a deep dive into research on skip waste collection routing problems. It will begin by reviewing different types of waste collection and the role of skip containers within them. Furthermore, the VRP will be discussed in more detail, with a focus on the aspects commonly encountered in skip container waste collection. Finally, metaheuristics will be reviewed to identify the most widely used approaches for addressing VRPs in waste collection.

2.1. Waste Collection

To understand where the skip container fits within waste collection, different types of waste collection can be distinguished. Waste collection can be divided into three areas: commercial, residential and rollon-rolloff (RoRo) (Golden et al., 2002). All types cover municipal solid waste and recycling material; hazardous waste is excluded. Commercial waste collection is focused on commercial locations, such as restaurants and office buildings. Residential waste is the waste from private homes. RoRo differs from commercial and residential operations because it involves moving containers rather than directly emptying them into a vehicle. RoRo problems involve the pickup, transportation, unloading, and drop-off of large containers (Bodin et al., 2000; Wy et al., 2013). Skip containers belong to the RoRo waste collection category.

2.2. Skip Container Routing

The skip container is a type of metal container that is used, for example, to transport debris from construction sites to disposal facilities (Wøhlk & Laporte, 2022). Originally, the term RoRo refers to the physical loading mechanism in which containers are rolled or pulled onto and off a truck. However, in the routing literature, the term is used more broadly to describe the transportation of large containers that are not emptied directly into the vehicle (Bodin et al., 2000).

The term ”skip container” itself covers several container types, of which a few are illustrated in Figure 2.1. A key distinction can be made between chain-lift skips (top row) and RoRo skips (bottom row). Chain-lift skips are smaller open containers that can be stacked, allowing multiple units to be transported on a single vehicle. In contrast, RoRo skips are significantly larger and are handled using a rolling mechanism. Due to their size, typically only one container can be transported per vehicle, or at most two when a trailer is used.

Despite these differences, all skip types are transported as separate units and must be emptied at a disposal facility. Based on the classification of waste collection by Golden et al. (2002), these

characteristics place skip container transport within the RoRo category of waste collection. However, since this research focuses on stackable chain-lift skip containers rather than traditional rolling systems, the routing problem is referred to as the *skip container VRP*. The term RR VRP is only used when referring to studies that explicitly use this terminology.

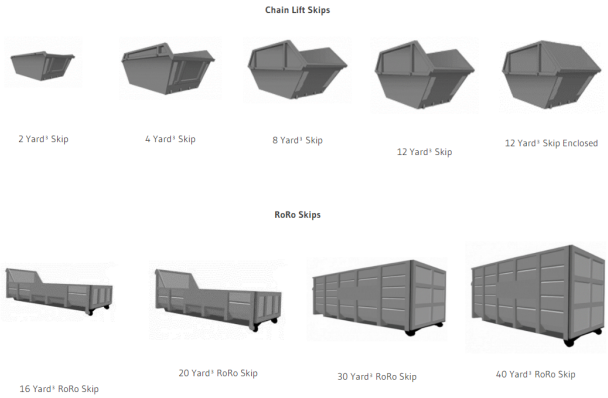


Figure 2.1: Different skip container types (REMONDIS UK, 2025).

These operational characteristics directly affect the routing model. Since vehicles transport entire containers, capacity is defined by the number of containers they can carry simultaneously rather than by weight or volume. The transportation of containers also introduces dependencies between pickups and deliveries. For example, after collecting a full container, the vehicle must first visit a disposal facility before continuing to the next customer. Other commonly considered constraints include container ownership, customer time windows, and different container types (Hess et al., 2024).

Most formulations in the literature assume that each vehicle carries only one container. In the *1-skip collection problem* by Archetti and Speranza (2004), each collected skip must be delivered to a disposal facility before visiting the next customer, with all routes beginning and ending at the depot. Several other studies also describe the RR VRP as a single-container problem (Aringhieri et al., 2018; Bodin et al., 2000; Wy et al., 2013).

Fewer studies consider vehicles that can transport two skips, resembling a truck with an additional trailer capable of carrying two RoRo skips. Archetti et al. (2005) and Raucq et al. (2019) studied such a two-skip problem, although the work of Archetti et al. (2005) focuses only on delivery operations. More recent papers model the two-skip problem as a three-stage Pickup and Delivery Problem (PDP) consisting of a pickup, treatment, and delivery sequence (Belenguer et al., 2024; Wøhlk & Laporte, 2022). Extensions beyond two skips are rare. Only Hauge et al. (2014) modelled up to eight skips, allowing stacks of up to four containers, although under simplified assumptions, such as permitting at most one full skip per stack.

More recently, Li et al., 2018 introduced a generalised RR VRP (G-RRVRP) that brings the problem closer to real operations. The paper describes the synchronisation of loaded and empty container routing. The study moves away from simpler models that assume an unlimited supply of empty containers at a depot. Instead, their model explicitly integrates the management of both full- and empty container movements within the network. This synchronisation is an important addition because a vehicle's ability to collect a full container depends directly on the availability of an empty container and vice versa. This is achieved using the concept of *container pools* to store, supply, and receive empty containers and a separate location type to receive loaded containers from customers.

Although recent models account for more realistic operational constraints, detailed stacking rules remain largely unexplored. Current skip container VRP formulations still focus predominantly on RoRo skips and therefore neglect other types of skip containers that can be stacked. Addressing this gap requires considering not only the number of containers a vehicle can carry but also how

they are organised or stacked within the vehicle, a topic that will be explored further in the next section.

2.3. Container Stacking

In waste collection, vehicle efficiency is influenced not only by the number of containers transported but also by how they are organised within the vehicle. In the context of skip containers, studies have highlighted that container stacking can affect operational efficiency and feasibility. Stacking is constrained by physical and operational considerations, for example, the maximum number of containers that can be stacked and compatibility between container sizes (Hauge et al., 2014). These aspects can be addressed using concepts from related problems, such as the capacitated vehicle routing problem (CVRP) and the bin packing problem (BPP).

2.3.1. Container Stacking in Other Fields

In two-dimensional CVRP (2L-CVRP), loading constraints account for the length and width of each object, ensuring that the elements do not overlap and remain within vehicle boundaries (Zachariadis et al., 2017). This has been extended to three-dimensional CVRP (3L-CVRP), which also takes into account the vertical orientation (Fuellerer et al., 2010; Tao & Wang, 2015). This model guarantees that each customer receives the correct quantity of freight while keeping the total load within the vehicle's weight limit. This can be translated into the skip container VRP to ensure that the number of stacked containers does not exceed the vehicle's weight limit. Similarly, the three-dimensional BPP (3D-BPP) enforces non-overlapping constraints of the placement of containers within the vehicle (Fuellerer et al., 2010; Tao & Wang, 2015). An example of a 3D-BPP from another domain is air cargo, where items of varying sizes must be packed into a container with fixed dimensions. In the container, the items must be adequately supported and fragile items can only be placed on top of other items (Junqueira et al., 2012; Paquay et al., 2016). This modelling logic can also be applied to skip containers, where only the full container is placed on top and only a smaller container can be stacked on a larger one and not vice versa.

Stacking has also been applied in maritime container terminals, where containers are stored vertically in stack lanes within a yard. Gunawardhana et al. (2021) researched the Container Stacking Problem (CSP), focusing on assigning incoming containers to specific yard locations while taking into account container characteristics and yard constraints such as capacity and safe stacking heights. One of the KPIs mentioned is the storage space, which shows that stacking decisions directly affect operational efficiency. Gunawardhana et al. (2021) report cost reductions of approximately 3–7% when rule-based stacking strategies are applied. Furthermore, Dekker et al. (2006) points out strict physical stacking rules stating that stacking can only be done along a lane and that different container sizes cannot be stacked on top of each other. Relating this to skip container operations, a vehicle can be seen as a stacking lane with a few stacking spots. In addition, the vehicle also has limitations for safe stacking heights. In contrast, skip containers can be nested when stacked, meaning that the restriction of stacking only identical container types does not apply.

2.4. Order Types

Beyond stacking, the skip container VRP can be further extended by considering the different order types in skip operations, which reflect the various services. These main order types have been mentioned in multiple studies (Bodin et al., 2000; Golden et al., 2002; Hauge et al., 2014; Raucq et al., 2019; Wy et al., 2013), the terminology used follows that of Hauge et al. (2014). The four main order types are defined as follows (visuals can be found in Appendix F):

1. **Pickup**: pickup a container at a customer, empty it at a disposal facility, and transport it to a storage location
2. **Delivery**: pickup an empty container at a storage location and deliver it to a customer
3. **Change**: pickup a container at a customer, empty it at a disposal facility, and return it to the customer

4. **Swap**: pickup an empty container at a storage location and deliver it to a customer, then pickup another container at the customer, empty it at a disposal facility, and transport it to a storage location

Additionally, Wy et al., 2013 also defines these order types:

- **Bring to yard**: move a container from a customer to a storage location
- **Full from Depot**: move a full container directly from depot to a disposal facility
- **Relocate**: move container between two customer locations

These order types can be further expanded with the option of **reuse**, which requires identical container types (Hauge et al., 2014). In this context, an empty container obtained from a pickup or swap order is immediately reused for another customer order rather than being returned to a depot. In the paper, they apply reuse through an *annihilation* principle, which occurs during the route construction process. This occurs by merging the final visit of a pickup or swap order with the first visit of a subsequent compatible order, combining segments of two separate orders into a single feasible route. Because annihilation eliminates unnecessary depot visits and the associated handling and travel time, it reduces the total cost of the route. Consequently, the Tabu Search (TS) algorithm favours these routes as it aims to generate routes with the lowest possible cost. Although this approach improves efficiency, it also introduces additional dependencies between orders within a route. Furthermore, Archetti and Speranza (2004) already implicitly incorporates this routing order. After emptying a skip at a disposal facility, the vehicle continues directly to the next customer. However, their model considers only a single service order that is repeated throughout the working hours. Only in the case of privately owned containers does the skip have to be returned to the customer, which breaks up this standard routing sequence. A visual of how reuse can be applied can also be found in Appendix F.

The literature also presents several extensions that, while not identical to reuse, address similar challenges in coordinating container flows between customers. For example, the fifth trip type defined by Baldacci et al. (2006) covers relocating partially filled containers between customers. More recently, Wøhlk and Laporte (2022) proposed the flexible return model, which allows empty skips to be returned to any compatible disposal facility instead of their original location. This model, which achieved an average cost savings of 6.1%, extends reuse to the whole system by spreading container flows more evenly over time. However, these models differ from the immediate reuse described in Hauge et al., 2014, in which a specific container serves the next customer directly.

The option of reuse highlights the need to consider additional factors, such as container ownership and availability. Container availability can be assumed to be unlimited (Archetti & Speranza, 2004; Baldacci et al., 2006; Bodin et al., 2000) or limited (Aringhieri et al., 2018; Wy et al., 2013). Regarding ownership, Aringhieri et al., 2018 distinguishes between company-owned and privately-owned containers. Company-owned containers are interchangeable with each other and privately-owned containers have to be returned to their original location. The distinction in ownership is also reflected in certain order types. The change order is focused on privately-owned containers due to the return of the container to the customer (Bodin et al., 2000; Hauge et al., 2014; Raucq et al., 2019; Wy et al., 2013).

2.4.1. Container Reuse in Other Fields

The waste industry is not the only sector focused on container reuse. A similar perspective can be found in the container drayage literature. For example, Fazi et al., 2023 studied the multi-trip container drayage problem and highlighted the importance of efficient reuse of empty containers, which is known as street-turning. Similarly, as with the reuse option in the skip container VRP, empty containers released by one shipper can be used directly by another shipper instead of being returned to a depot first. Their paper introduces six shipper types that resemble the order types defined in skip container operations. Furthermore, Zhang et al., 2020 notes that most research assumes a sufficient supply of empty containers, whereas it is more realistic to model this as limited. Under such conditions, reuse can have a larger impact, by reducing unnecessary depot

trips. This can also reduce costs, where at the moment, despite the short distances, the cost of container drayage represents 25-40% of the entire transportation chain (Zhang et al., 2020).

These operational challenges can also be seen on a global level in liner shipping. Repositioning empty containers is also costly here, with around 16 billion USD, about 15% of total container fleet management costs, spent each year on moving empty containers (Lee & Song, 2017). Studies on liner shipping service networks show improved efficiency when repositioning of empty containers is included. Shintani et al. (2007) demonstrate that integrating empty container movements into network planning helps avoid suboptimal routing when loaded and empty flows are handled separately, and Meng and Wang (2011) showed significant cost reductions when repositioning is taken into account in network design. These findings underline that, whether in waste collection, drayage, or maritime logistics, efficient reuse of empty containers between customers can reduce transportation time and lower operating costs.

2.5. Vehicle Routing Problems

As discussed earlier, the skip container VRP is a specific variant of the VRP, so it is useful to also explore the VRP in general, which this section will do in more detail. The VRP is an extensively studied problem in operations research and logistics. It was first introduced by Dantzig and Ramser, 1959 as 'The Truck Dispatching Problem'. The essence of the VRP is to design a set of minimum cost routes for a fleet of vehicles serving multiple customers. These routes typically start and end at a central depot (Laporte, 2009). Among the various types of VRPs, the PDP is the most relevant for this research. In the context of waste collection, the important locations are: depots, customers, storage locations, and disposal facilities. Furthermore, vehicle capacity is limited. Since services require both the delivery of an empty skip and the pickup of a full skip, the problem can be classified as a PDP.

2.5.1. Pickup and Delivery Problem

The PDP is a class of VRPs in which each request specifies how to transport an item from a pickup location to a delivery location. It is one of the standard one-to-one routing models and is commonly studied both with and without time windows (Dumas et al., 1991; Savelsbergh & Sol, 1995). The PDP has been studied a lot in research because it can be applied to many real-world cases, such as parcel service, ride-sharing and freight logistics. The core complexity of the PDP lies in the precedence constraint that a pickup must occur before its corresponding delivery, as well as the need to satisfy time windows, which will be discussed in detail later (Savelsbergh & Sol, 1995). The PDP is relevant for skip container operations, where precedence relations exist between pickup and delivery operations involving full and empty skips. In this sense, the skip container VRP can be seen as a special case of PDP (Raucq et al., 2019). This view is already reflected in early RR VRP literature. Bodin et al. (2000) notes that, in the absence of trip type 2, the RR VRP can be regarded as a PDP.

The PDP has been expanded to include transfer options (PDPT) (Cortés et al., 2010). This model relaxes the constraint that a single vehicle must complete both the pickup and delivery of a request, allowing items to be transferred between vehicles. This increases operational flexibility and can reduce total travel distance. Similarly, the multi-pickup and delivery problem with time windows (MPDPTW) has been introduced, where a service consists of multiple pickups followed by a single delivery (Naccache et al., 2018). In the context of waste collection, this could involve transporting two containers from different customers to the same disposal facility. The second trip type defined by Bodin et al. (2000), which involves delivering an empty container and collecting a full one within a single service that may include intermediate locations, already points towards such an MPDP, even though it is not explicitly modelled as such a model.

A recent development in the field has been the direct modelling of the skip container VRP as a PDP variant. Wøhlk and Laporte (2022) introduced the skip PDP (SPDP), which models the transport of skips between disposal facilities. The paper distinguishes between a fixed-return variant, where the empty skip is returned to its origin, and a flexible-return variant, which allows for the empty skip to be returned to any compatible location. The SPDP formulations incorporate realistic skip operational characteristics, such as destination depending on container content and multiple

facility types. Building on this framework, Belenguer et al. (2024) focused specifically on the fixed-return case, formalised as the SPDP-R, and proposed a strengthened mathematical formulation with tighter constraints. The SPDP can therefore be interpreted as a skip-specific PDP formulation within the broader RR VRP class.

2.5.2. Time Windows

A common constraint in VRP research is the specification of time windows (see Table 2.1). A time window specifies a time range, defined by an earliest and latest time, during which a service must be performed at a specific location, such as a customer's site, a depot, a storage location or a disposal facility (Benjamin & Beasley, 2010; Wy et al., 2013). The time windows can be defined as strict, which means that they must be complied with, or soft, where violation is allowed but a penalty cost follows (Archetti et al., 2025; Braekers et al., 2016). In skip container VRP, time windows are almost always treated as strict constraints (Benjamin & Beasley, 2010; Wy et al., 2013). This is because the time windows correspond to fixed facility opening hours that cannot be adjusted.

Including time windows can increase the complexity of the model. The model must now also track the arrival and departure schedules at the different locations. This creates interdependencies: the feasibility of arriving at a customer at a certain time depends on the arrival/ departure times at all the preceding locations on the route (Kim et al., 2006). In contrast, time windows can also serve to reduce the solution space. By specifying a particular interval, certain otherwise valid routes may be excluded as they become infeasible. As a result, the inclusion of time window constraints narrows the set of feasible solutions, thus reducing the search space for a solution (Dumas et al., 1991).

2.5.3. Multiple Locations

Another common extension of the VRP is the inclusion of multiple depots or locations. This is relevant in waste collection, where depots, storage locations, and disposal facilities often operate across several locations (Baldacci et al., 2006). Introducing multiple disposal facilities helps to more accurately represent real-world operations. Since waste management companies often handle different types of waste, it becomes necessary to include different facilities to accommodate these variations (Raucq et al., 2019). In addition to depots and disposal facilities, several skip container VRP variants explicitly model container storage locations, where empty containers are stored and retrieved during routes (Raucq et al., 2019; Wy et al., 2013). These storage locations introduce further routing decisions, as vehicles may need to visit them to load or unload empty containers depending on the service type. Adding more locations complicates the problem, as the model must decide which facility to use (Kim et al., 2006). This choice is based on the current position of the vehicle, the facility that is within opening hours (time windows), and the remainder of the route (Benjamin & Beasley, 2010).

2.6. Metaheuristics for VRPs

The VRP is an NP-hard problem. This is a category of computation problems for which finding an exact and optimal solution is not feasible in a reasonable amount of time, because the problem increases in the number of options (Archetti et al., 2005). They state that a skip container VRP with a capacity bigger than two is NP-hard even under strict conditions. Therefore, metaheuristics are more suitable for solving the problem in a reasonable time (Fuellerer et al., 2010). A review of the VRP literature published between 2009 and mid-2015, which reviewed almost 300 articles, found that 71.25% of the surveyed articles use metaheuristics. These approaches are widely recognised as efficient for many NP-hard optimisation problems (Braekers et al., 2016).

Therefore, an overview of scoping in VRPs is useful. Various constraints have been mentioned in this chapter, but they are not all equally represented in the literature. Braekers et al. (2016) show that around 38% of the surveyed VRP papers consider time windows, which are mainly applied at customer locations, making them the most commonly studied real-life constraint. In contrast, more complex constraints such as multiple depots and precedence relations are less frequently used, appearing only in 11% and 8.56% of the reviewed studies, respectively.

Building on this work, Elshaer and Awad (2020) specifically focuses on VRPs solved by metaheuristics and classifies the applied methods into single-solution based and population-based metaheuristics. Single-solution based methods start from an initial solution and iteratively improve it by exploring its neighbourhood, following a trajectory through the search space. Well-known examples include TS, variable neighbourhood search (VNS), and iterated local search (ILS) (Boussaïd et al., 2013). In contrast, population-based metaheuristics operate on a set of solutions simultaneously, promoting diversification through interactions between multiple possible solutions.

The review of Elshaer and Awad (2020) shows that single-solution based metaheuristics dominate the VRP literature with 64% of the papers classified in this category. Within the single-solution based, TS has been applied the most (30%), followed by VNS (23%), and LNS (15%). This pattern suggests that trajectory-based metaheuristics are well-suited to routing problems, as routes can be gradually improved through local changes while keeping constraints satisfied.

2.6.1. Solutions in Skip Container and RR VRPs

To review the solution methods applied in skip container and RR VRP research, Table 2.1 presents an overview of the selected papers. The table does not aim to cover all aspects mentioned in the papers, but only those relevant to the scope of this research. A brief explanation of the terminology used is provided below the table to support interpretation.

Table 2.1: Skip Container and RR VRP related papers overview (ordered on publication date).

Reference	Max vehicle Capacity	Stacking	Reuse	Locations	Constraints	Solution Method(s)
Bodin et al. (2000)	1	No	No	SD, SF		PEM, SA, DA, TI^2
Archetti and Speranza (2004)	1	No	Reuse	SD, MF	TW, MC	Nearest-neighbourhood
Archetti et al. (2005)	2	No	No	SD, SF		Exact
Baldacci et al. (2006)	1	No	No	SD, MF, MY		Exact
Benjamin and Beasley (2010)	1	No	No	SD, MF	TW	Tabu search & neighbourhood search
Wy et al. (2013)	1	No	No	SD, MF, MY	TW, MC	Large neighbourhood search
Hauge et al. (2014)	8	Yes	Reuse	SD, MF, MY	TW, MC	Tabu search & column generation
Aringhieri et al. (2018)	1	No	No	SD, MF	MC	Neighbourhood-based
Li et al. (2018)	1	No	Relocate	SD, MF, MY	TW	MILP & Savings & local search
Raucq et al. (2019)	2	No	No	MD, MF, MY	TW, MC	Column generation & heuristic pricing
Wøhlk and Laporte (2022)	2	No	Flexible return	MD, MF		Variable neighbourhood search
Belenguer et al. (2024)	2	No	Flexible return	SD, MF		MILP & Branch-and-cut
This research	Variable	Yes	Reuse	SD, MF, MY	MC	Metaheuristic

Description of abbreviations and terms used in Table 2.1.
textitReuse refers to the direct reuse of an emptied container by another customer without returning it to a depot, *relocate* denotes the repositioning of empty containers between locations to balance availability, and *flexible return* allows an emptied container to be returned to a compatible alternative location rather than to its original customer.

Constraints

- SD = single depot
- MD = multiple depots
- SF = single disposal facility
- MF = multiple disposal facilities
- SY = single container storage yard
- MY = multiple container storage yards
- TW = customer time windows
- MC = multiple container types

Heuristics

- Decomposition Algorithm (DA)
- Partial Enumeration Method (PEM)
- Simple trip insertion/trip improvement procedure (TI^2)
- Savings Algorithm (SA)

It is also evident from Table 2.1, that the complexity of the skip container VRP is linked to the chosen solution method. When reviewing the methods used, it becomes clear that papers rely-

ing on exact methods typically address simplified problem variants that do not include aspects such as stacking or container reuse. When these features are considered, they are usually integrated as part of a hybrid optimisation approach, as is the case in Li et al. (2018). As soon as more realistic aspects are introduced, the literature consistently shifts towards hybrid approaches or metaheuristics. An example is the paper of Hauge et al. (2014), where stacking rules, time windows, and multi-container vehicles are introduced. To handle these constraints, the paper uses a hybrid column generation approach with TS. This is consistent with the broader VRP literature, in which metaheuristics are the dominant solution approach. The metaheuristics listed in Table 2.1 all belong to the category of single-solution based metaheuristics. Moreover, the selected methods correspond to the neighbourhood-based metaheuristics most frequently identified in the surveys by Braekers et al. (2016) and Elshaer and Awad (2020), including TS, VNS, and LNS. Therefore, the next section provides a more detailed discussion of neighbourhood-based metaheuristics and their structure.

2.6.2. Neighbourhood-based Metaheuristics

Because neighbourhood-based metaheuristics are the most commonly used methods in this scope, the following section focuses on these approaches in more detail. Several shared characteristics of these metaheuristics are discussed, including the construction of the initial solution and the operators used to explore the solution space.

Initial Solution

Neighbourhood-based metaheuristics start from an initial solution. To obtain that initial solution, an algorithm or heuristic needs to be applied. Below in Table 2.2 is an overview of some of the papers mentioned in Table 2.1 and mentioned in the review paper Elshaer and Awad (2020) to give an overview of the method used.

Table 2.2: Overview of initial solutions used in neighbourhood-based metaheuristics.

Paper	Model type	Initial solution
Benjamin and Beasley (2010)	Waste collection VRP	Sequential greedy route construction
Wy et al. (2013)	RoRo VRP	Constructive heuristic with a candidate list
Dominguez et al. (2016)	2L-VRP	Savings heuristic
François et al. (2016)	Multi trip VRP	Greedy heuristic
Aringhieri et al. (2018)	RoRo VRP	Savings heuristic
Li et al. (2018)	RoRo VRP	Savings heuristic
Wøhlk and Laporte (2022)	Skip PDP	Greedy constructive heuristic

The initial solution methods in Table 2.2 are all constructive heuristics. Constructive heuristics build a solution sequentially by adding one customer at a time. One of the earliest constructive heuristics is the nearest neighbour heuristic, which extends a route by repeatedly selecting the closest unvisited customer (Rosenkrantz et al., 1977). On the contrary, insertion heuristics start from a partial route and iteratively insert customers at the position that minimises additional travel cost (Potvin & Rousseau, 1993). Moreover, a basic greedy insertion heuristic selects in each iteration the request–route combination with the smallest increase in objective value (Ropke & Pisinger, 2006). Despite their computational efficiency, purely greedy selection rules consider only immediate cost increases and may therefore postpone structurally difficult insertions.

To address this limitation, regret-k insertion heuristics incorporate limited look-ahead by selecting the request for which the difference between the best and second best insertion positions is largest, thereby prioritising critical requests (Ropke & Pisinger, 2006). Another classical constructive approach is the savings heuristic of Clarke and Wright (1964), which starts with one route per customer and iteratively merges routes according to the largest distance savings (Clarke & Wright, 1964). Due to their simplicity, low computational effort, and ability to quickly generate feasible solutions, nearest neighbour, insertion-based, regret-based, and savings heuristics are widely used as initialisation procedures in vehicle routing metaheuristics (Laporte, 2009).

Operators

After constructing the initial solution, multiple operators can be applied in order to improve the route solution. Operators define the neighbourhood structure and therefore determine how the search explores the solution space. In classical neighbourhood search for VRPs, operators are typically divided into intra-route and inter-route moves. Intra-route operators modify the sequence of customers within a single route. Well-known examples include 2-opt and 3-opt exchanges, where sequences of nodes are reversed, and Or-opt, which relocates short sequences of consecutive customers within the same route (Liu et al., 2023). These operators are computationally efficient and effective in reducing travel distance while maintaining feasibility with respect to capacity and time window constraints. Inter-route operators transfer customers between routes. Typical examples are relocate, exchange, and 2-opt, which swaps route segments between two routes (Liu et al., 2023).

In addition to these small local moves, larger neighbourhood operators have been introduced. Instead of modifying only a few arcs, a subset of customers is removed from the solution and reinserted using a constructive heuristic. For example, in LNS this enables more substantial changes and to escape local optima (Ropke & Pisinger, 2006). Insertion strategies range from greedy rules, which insert customers at the position with minimal additional cost, to regret-based rules that incorporate limited look-ahead by prioritising customers whose postponement would cause the largest future cost increase (Ropke & Pisinger, 2006).

A recent systematic review by Voigt (2025) classifies and ranks the operators used in adaptive LNS (ALNS) to gain inside in operators used for VRPs. The study identifies 57 distinct removal operators and 42 insertion operators and shows that performance differs substantially between operator classes. In particular, sequence-based removal operators, which remove consecutive customers from the current solution, are found to be among the most effective removal strategies. The two most commonly used removal operators are random customers and worst-cost customers. Insertion heuristics can be divided into two categories: sequential and parallel. The sequential builds one route at a time, while parallel heuristics construct several routes (Ropke & Pisinger, 2006). For insertion, regret-based operators generally outperform purely greedy rules. The review further emphasises that a balanced combination of diversification orientated operators (e.g., random removal) and intensification orientated operators (e.g., worst-cost or regret-based insertion) leads to more robust performance.

For complex VRP variants such as the skip container VRP, operators must respect additional structural constraints, including pickup-delivery precedence, vehicle capacities, stacking feasibility, and potential reuse dependencies. Small intra- and inter-route operators are effective for incremental improvements of route sequences, while larger removal reinsertion heuristics enable more extensive changes when routes are strongly interdependent. Consequently, the literature advocates combining local search moves with more disruptive operators to balance intensification and diversification within the search process.

2.7. Conclusion

The skip container VRP is a waste collection problem in which vehicles transport entire containers between depots, customers, storage yards, and disposal facilities. Although all skip containers are transported as whole units, an important distinction exists between large RoRo skips and smaller chain-lift skips. The literature has predominantly focused on RoRo systems, where vehicle capacity is typically limited to one container (or two with a trailer), and operations are closely tied to the rolling-loading mechanism. In contrast, chain-lift skips allow stacking, thereby introducing different operational possibilities, particularly regarding vehicle capacity and route design. While recent studies have incorporated more realistic features such as multiple container types and the synchronisation of loaded and empty flows, they largely continue to rely on assumptions derived from RoRo operations. As a result, two key gaps remain: first, stacking is generally simplified to a numerical capacity constraint, neglecting physical organisation and compatibility; second, the modelling of container reuse beyond standard order types remains limited.

The literature further indicates that increasing realism, through features such as stacking, multiple

container types, and complex precedence relations, significantly increases computational complexity. Exact methods are therefore mainly restricted to simplified problem variants, while more realistic formulations rely on metaheuristics. In particular, single-solution and neighbourhood-based metaheuristics such as TS, VNS, and LNS are widely used for their ability to handle complex constraints. Consequently, the literature suggests that metaheuristic approaches provide a suitable framework for extending skip container VRPs to better capture the operational characteristics of chain-lift containers.

3

Problem Description

This chapter outlines the scope of the problem. More background information is provided in Chapter 2. After the problem is defined, the scope of the benchmark model will be discussed, followed by a visual to illustrate it further. Furthermore, the same will be done for the scope of the extended metaheuristic model. Lastly, the assumption made in this research will be summarised.

3.1. Problem Definition

The problem considered in this thesis is a VRP focused on skip container operations within a single working day. Vehicles transport skip containers between a depot, customers, storage locations, and disposal facilities. Routes consist of sequences of predefined order types that determine container pickup and delivery actions. Vehicles must account for capacity constraints, including the number of containers, stacking restrictions, and, later on also nesting compatibility between container types. The routing order must align with the precedence relationship between pickup and delivery activities. Furthermore, in the extended model, the service times depend on the order type and whether the container is full. The objective is to find a feasible route that minimises the total travel time.

3.1.1. Scope Benchmark Model

The benchmark problem defines a simplified reference version of the skip container VRP. It contains a set of modelling assumptions that will serve as the reference model for the extended model later. A visual showing how the elements can connect in a single customer visit is shown in Figure 3.1.

- One day of modelling
- Four routing orders
- Perform orders simultaneously
- One container type
- All company owned containers
- Unlimited container availability
- Unlimited container storage at locations
- Max. number of stacked containers
- One (un)loading time
- One depot
- One storage location
- Multiple customer locations
- One disposal facility
- Fixed number of vehicles

Figure 3.1: Visual of the benchmark for a pickup order.

The four core order types, visuals can be found in Appendix F:

1. Pickup: pickup a container at a customer, empty it at a disposal facility, and transport it to a storage location
2. Delivery: pickup an empty container at a storage location and deliver it to a customer
3. Change: pickup a container at a customer, empty it at a disposal facility, and return it to the customer
4. Swap: pickup an empty container at a storage location and deliver it to a customer, then pickup another container at the customer, empty it at a disposal facility, and transport it to a storage location

The elements listed above define the scope of the benchmark model. To maintain focus on the core routing structure, only the four fundamental order types mentioned in section 2.4 are considered. Each route starts and ends at the depot. Within a route, orders can be performed simultaneously as long as they adhere to the precedence order. To prevent the model from shifting towards inventory management, container availability is assumed to be unlimited and stored at the storage locations. Furthermore, stacking limitations are defined solely by the maximum number of stacked containers, with a separate definition of the maximum number of full containers. By restricting the stacking logic at this stage, the contribution of more detailed stacking constraints can be explicitly evaluated when they are introduced in the extended model. Additional limitations, imposed to keep the benchmark as simple as possible, include assuming a single handling time for all order types and a single storage location and disposal facility. Lastly, a fixed fleet of vehicles is available with the objective of minimising the number of vehicles used.

3.1.2. Scope Extended Model

The extended model builds upon the benchmark by incorporating additional aspects that better reflect the real world. The elements marked in bold indicate additions or modifications relative to the benchmark model and are highlighted in green in Figure 3.2.

- One day of modelling
- Four routing order **+ reuse possibility**
- Perform orders simultaneously
- **Two container types**
- All company owned containers
- Unlimited container availability
- Unlimited storage capacity containers at the locations
- **More detailed stacking**
  - **Fixed number of stacking spots**
  - **Max. height stacked containers**
  - **Nesting rules**
  - **Full container on top**
- **Different (un)loading times**
- One depot
- **Two storage locations**
- Multiple customer locations
- **Two disposal facilities**
- Fixed number of vehicles

Figure 3.2: Visual of the extended model elements for a sequenced visit

To enable more efficient routing, container reuse is incorporated into the model. Reuse allows an emptied container to be delivered directly to another customer rather than being returned to a storage location first. To support this, the model tracks the individual location of each container, enabling reuse decisions at the container level. After a container is emptied at a disposal facility, the model determines whether it can be reused and, if so, to which customer it can be delivered next instead of being transported to a storage location. When an order consists of multiple containers, partial reuse is also possible. Since the model includes multiple container types, reuse is only allowed between compatible container types.

In addition, a more detailed representation of stacking is introduced. Stacking is not treated as a separate optimisation problem, but is checked through feasibility rules. Routes generated by the metaheuristic algorithm are accepted only if a feasible stacking configuration exists throughout the entire route. The level of detail is increased by introducing a fixed number of stacking positions, nesting rules between different container sizes, and explicitly dedicating the top row to full containers, ensuring that only the top containers can be full. These stacking constraints also affect reuse decisions: reuse is no longer accepted solely because it reduces routing cost, but only if it remains physically feasible. Furthermore, all containers remain company-owned and have unlimited availability.

To further increase realism in the extended model, service times are differentiated based on whether containers are loaded or unloaded and whether they are empty or full. In addition, multiple disposal facilities and storage locations are introduced to better reflect real-world operational settings.

3.1.3. Additional Assumptions

In addition to what has already been discussed above, further assumptions have been made in this research, which are summarised below.

- Customer requests are assumed to be known beforehand and no dynamic or real-time orders are considered.
- Travel times between all locations are assumed to be deterministic and known in advance based on a static distance matrix.
- Traffic congestion, stochastic delays, and uncertainty in service times are not considered.
- Driver related regulations, such as mandatory breaks, working hour legislation, and shift limitations are not included in the model.
- Vehicles are assumed to start and end at predefined depot locations.
- The fleet is assumed to be homogeneous with respect to the loading structure and operational capabilities.
- Maximum weight restrictions of the vehicle are not considered.
- Eligibility for disposal facilities and storage locations is predefined and assumed to remain fixed.

- When no valid location is specified in the input data, all locations of the corresponding type are assumed to be feasible alternatives.
- The vehicle loading configuration is represented by discrete loading positions, where containers can only occupy predefined loading spots.
- The model includes both small and large container sizes. Small containers can be stacked on top of large containers, while large containers cannot be stacked on top of small containers. Furthermore, no stacking restrictions are imposed between containers of the same size category. However, reuse is only allowed between containers of the exact same container type. As a result, containers belonging to different small container types cannot be reused interchangeably.
- Loading and unloading operations are assumed to be feasible whenever sufficient loading positions are available. Detailed physical handling constraints are therefore not explicitly modelled. Consequently, no additional handling time is considered when containers positioned on top of another container must temporarily be moved in order to access a container located at the bottom of the stack.

4

Methodology

This chapter presents the methodology for modelling and solving the skip container VRP introduced in the previous chapter. Two solution approaches are developed: an exact optimisation approach and a metaheuristic approach.

First, a benchmark model is formulated as an MILP model. This benchmark represents standard skip container operations under simplified assumptions and is used as a reference model for evaluating the extended formulation. In particular, container reuse is not included and stacking is represented in a simplified manner. The MILP model is implemented in `Python` and solved using the `Gurobi` solver. Second, an extended routing model is developed that incorporates container reuse and detailed stacking constraints. To solve this model, a metaheuristic algorithm is developed. The conceptual structure of the algorithm is illustrated using a flowchart, and the main algorithmic steps are presented using pseudocode.

The chapter is structured as follows. section 4.1 introduces the benchmark MILP formulation, including the sets, parameters, decision variables, objective function, and constraints. section 4.2 presents the design of the metaheuristic algorithm and discusses the different algorithmic components. Finally, the selected KPIs are discussed in section 4.3.

4.1. Mathematical Model

This section presents the mathematical model formulated for the benchmark model. The problem consists of a set of nodes N, including the depot, customer locations, storage locations, and a disposal facility. Storage, disposal facilities, and return customer locations are modelled using dummy nodes to allow multiple visits while preserving routing feasibility. The dummy customer nodes are introduced for change orders to represent the return visit to the same physical customer location. Therefore, the node set is formulated as

$$N = \{D\} \cup C \cup C^{ret} \cup S' \cup F',$$

where node D denotes the depot, C the customer nodes, C^{ret} the customer return dummy nodes, S' the storage dummy nodes, and F' the disposal facility dummy nodes. Although dummy nodes may correspond to the same physical location as their original node, they are treated as separate routing nodes in the network. Routing decisions are defined on all pairs of nodes $i, j \in N$.

The customer demand is represented by a set of order requests R. Each request $r \in R$ is con-

nected to a customer $c \in C$, is characterised by an order type

$$\tau_r \in O \text{ (with } O = \textit{delivery, pickup, swap, change}),$$

and specifies a container demand d_r. The order type τ determines both the set of components,

$$P = \textit{delivery}^P,\ \textit{delivery}^D,\ \textit{empty}^P,\ \textit{empty}^D,\ \textit{dropoff},\ \textit{return},$$

that must be performed and the sequence in which these components are executed. Some components, such as delivery and emptying, consist of two handling moments that occur at different locations, corresponding to pickup and dropoff activities.

An overview of the components P associated with each order type τ is provided below in Equation 4.1 and in the text described above the matrix. The parameter $\delta_{\tau p}$ indicates whether component p is required for order type τ. The parameter Δ_p, defined in Equation 4.2, specifies the changes in load associated with each component per container. To also track the number of full containers, there is a separate parameter, Equation 4.3, which tracks the changes of the full container load. Positive values correspond to pickup operations that increases the vehicle load, negative values represent delivery or drop-off operations that decrease the load, and the value zero indicates components that do not affect the vehicle load.

- Delivery: **pickup** of an empty container and **delivery** of this container to acustomer.
- Pickup: **pickup** of a full container from a customer to a disposal facility, to **empty** it there. After that **dropoff** at a storage location.
- Swap: **pickup** of an empty container and **delivery** of this container to a customer. Pickup of a full container from a customer to a disposal facility, to **empty** there. After that **dropoff** at a storage location.
- Change: **pickup** of full container from customer to disposal facility, to **empty** there. After that **return** the container to the customer.

$\delta_{\tau p} =$

$Type \downarrow \ \backslash \ Component \rightarrow$	delivery^P	delivery^D	empty^P	empty^D	dropoff	return
delivery	1	1	0	0	0	0
pickup	0	0	1	1	1	0
swap	1	1	1	1	1	0
change	0	0	1	1	0	1

(4.1)

$$\Delta_p^{tot} = \begin{cases} +1, & \text{if } p \in \{\text{delivery}^P,\ \text{empty}^P\} \\ -1, & \text{if } p \in \{\text{delivery}^D,\ \text{dropoff},\ \text{return}\} \\ 0, & \text{if } p = \text{empty}^D \end{cases} \tag{4.2}$$

$$\Delta_p^{full} = \begin{cases} +1, & \text{if } p = \text{empty}^P \\ -1, & \text{if } p = \text{empty}^D \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

The travel times t_{ij} between nodes are given as input parameters and are assumed to be deterministic. There is one handling time h, regardless of the activity. The total service time is obtained by multiplying the handling time by the container demand d_r. Other vehicle-related parameters include total container capacity C_k, full container capacity C_k^{full} and a maximum daily operational time T^{max}.

The variable x_{ijk} is a binary variable that indicates whether the vehicle k travels from node i to node j. The binary variable z_{ik} indicates whether node i is visited by vehicle k, while y_k specifies if vehicle k is used. The assignment of order requests to vehicles is handled by u_{rk}. The variable

v_{rpik} has the value one when vehicle k performs component p of request r at node i, thus assigning an individual order component to specific vehicles and nodes. It is required because of the dummy nodes, so components must be linked to the correct vehicle and specific dummy visits. Furthermore, the arrival time of vehicle k at node i is tracked by a_{ik}. Lastly, the variable q_{ik}^{tot} covers the total vehicle load and represents the number of containers carried by vehicle k at the moment of arrival from node i. In addition, q_{ik}^{full} tracks the number of full containers carried on a vehicle k.

Within an order request r, components must be performed at specific types of locations. The parameter $n_p \in \{C, S, F\}$ specifies whether the component p is handled at a customer, storage location, or disposal facility, and determines the corresponding set of candidate nodes $\mathcal{N}_{rp} \subseteq N$. This formulation allows a vehicle to visit the same physical location multiple times at different moments, with each visit being modelled separately.

In the benchmark, containers are considered in terms of total quantities rather than individual items. As a result, vehicle loads and handling operations are expressed in terms of container counts, which simplifies the model while keeping the essential routing and capacity characteristics of the problem. The total maximum number of containers that can be stacked is not defined as a separate parameter, as it is incorporated into the capacity of vehicle k, C_k^{tot}. If stacking is allowed and therefore can carry more containers, this will be reflected by increased vehicle capacity.

Table 4.1: Overview of the sets, parameters, and decision variables used in the skip container VRP benchmark.

Sets and indices		
N	Set of all nodes	$i, j \in N$
C	Set of customers	$c \in C$
S'	Set of dummy storage nodes	$i \in S'$
F'	Set of dummy disposal facility nodes	$i \in F'$
V	Set of vehicles	$k \in V$
R	Set of order requests	$r \in R$
O	Set of order types	$\tau \in O$
P	Set of order components	$p \in P$
Parameters		Unit
t_{ij}	Driving time from node i to j	min
C_k^{tot}	Maximum container capacity of vehicle k	# of containers
C_k^{full}	Maximum full container capacity of vehicle k	# of containers
T^{max}	Maximum daily operational time	min
V^{max}	Maximum number of vehicles k available	# of vehicles
d_r	Demand of order request r	# of containers
τ_r	Order type τ of order request r	$\tau_r \in O$
$\delta_{\tau p}$	1 if order type τ requires component p, 0 otherwise	$\{0, 1\}$
P_τ	Precedence relations between components p for order type τ in pairs (p, p')	-
n_p	Node type at which component p takes place	$n_p \in \{C, S, F\}$
Δ_p^{tot}	Load change direction per component p for all containers	$\{-1, 0, +1\}$
Δ_p^{full}	Load change direction per component p for full containers	$\{-1, 0, +1\}$
h	Handling time per container	min/container
M	Large constant	-
Decision variables		Unit
x_{ijk}	1 if vehicle k travel from node i to j, 0 otherwise	$\{0, 1\}$
z_{ik}	1 if node i is visited by vehicle k, 0 otherwise	$\{0, 1\}$
y_k	1 if vehicle k is used, 0 otherwise	$\{0, 1\}$
u_{rk}	1 if order request r is served by vehicle k, 0 otherwise	$\{0, 1\}$
v_{rpik}	1 if vehicle k performs component p of request r at node i, 0 otherwise	$\{0, 1\}$
a_{ik}	Time vehicle k arrives at node i	min
st_{ik}	Service time of vehicle k at node i	min
q_{ik}^{tot}	Total container load of vehicle k at arrival of node i	# of containers
q_{ik}^{full}	Total full container load of vehicle k at arrival of node i	# of containers

To reduce the size of the model, the service times are precomputed per request–component pair. For each request r and required component p, the handling time is s_{rp}, as defined in Equation 4.4. $\mathcal{C}(i) := \{(r,p) \mid i \in \mathcal{N}_{rp}\}$ denotes the set of all order components that can be executed at node i.

$$s_{rp} := h\, d_r \qquad \forall r \in R,\ \forall p \in P \text{ with } \delta_{\tau_{rp}} = 1 \tag{4.4}$$

While the parameters Δ_p^{tot} and Δ_p^{full} indicate the direction of the load change associated with an individual order component, the expressions Δ_{ik}^{tot} and Δ_{ik}^{full} represent the resulting net change in load when vehicle k performs one or more components during its visit to node i. These expressions depend on the assignment decisions through v_{rpik} and capture the total load effect scaled by the container demand d_r. In most order types, only a single component is performed at a given node. However, in swap orders, both the delivery of an empty container and the pickup of a full container take place at the customer location, and their combined effects are reflected here.

$$\Delta_{ik}^{tot} = \sum_{(r,p)\in\mathcal{C}(i)} \Delta_p^{tot}\, d_r\, v_{rpik} \qquad \forall i \in N,\ \forall k \in V \tag{4.5}$$

$$\Delta_{ik}^{full} = \sum_{(r,p)\in\mathcal{C}(i)} \Delta_p^{full}\, d_r\, v_{rpik} \qquad \forall i \in N,\ \forall k \in V \tag{4.6}$$

4.1.1. Objective Function

The objective function (4.7) minimises the total driving time of all vehicles. The driving time t_{ij} is determined by the routes of the vehicles through the network and depends on the routing decision variables x_{ijk}. Handling times are not included in the objective as the total handling time associated with an order request is fixed and independent of the routing decisions. However, when comparing the extended model, the service time will be added for comparison. Furthermore, vehicles may serve multiple order requests in sequence, and minimising total driving time implicitly discourages the use of additional vehicles, since deploying an extra vehicle results in additional travel to and from the depot.

$$min : Z = \sum_{k\in V} \sum_{i\in N} \sum_{j\in N} t_{ij} x_{ijk} \tag{4.7}$$

4.1.2. Constraints

Routing focused constraints

The route starts, Constraint 4.8, and ends, Constraint 4.9, at the depot. If vehicle k is used ($y_k = 1$), it leaves the depot exactly once and returns exactly once.

$$\sum_{j\in N\setminus\{D\}} x_{Djk} = y_k \qquad \forall k \in V \tag{4.8}$$

$$\sum_{i\in N\setminus\{D\}} x_{iDk} = y_k \qquad \forall k \in V \tag{4.9}$$

Moreover, additional constraints are imposed to ensure route continuity and feasibility. Constraint 4.10 enforces flow conservation at each non depot node $i \in N\setminus\{D\}$ for every vehicle $k \in V$, ensuring that the number of arcs entering node i is equal to the number of arcs leaving node i. Constraint 4.11 links the routing decision variable x_{ijk} to the node visit indicator z_{ik}, ensuring that a node i is visited by vehicle k if and only if exactly one outgoing arc is selected. Consequently, exactly one incoming arc must also be selected due to flow conservation. The one time visit restriction applies to model nodes rather than physical locations. Dummy nodes are introduced to represent additional visits to the same physical location when required, while preserving routing feasibility. As a result, a physical location may be visited multiple times within a route through

different dummy nodes, whereas each individual model node is still visited at most once. Finally, constraint 4.12 prohibits self-loops by preventing vehicle k from travelling from node i directly back to the same node.

$$\sum_{j \in N} x_{ijk} = \sum_{j \in N} x_{jik} \qquad \forall i \in N \setminus \{D\},\ \forall k \in V \tag{4.10}$$

$$\sum_{j \in N} x_{ijk} = z_{ik} \qquad \forall i \in N \setminus \{D\},\ \forall k \in V \tag{4.11}$$

$$x_{iik} = 0 \qquad \forall i \in N,\ \forall k \in V \tag{4.12}$$

Constraint 4.13 ensures consistency between routing and component assignment by enforcing that a vehicle can only perform an order component at a node if it actually visits that node.

$$v_{rpik} \leq z_{ik} \qquad \forall r \in R,\ \forall p \in P,\ \forall i \in \mathcal{N}_{rp},\ \forall k \in V \tag{4.13}$$

Order assignment and consistency constraints

The following constraints ensure a consistent and feasible assignment of order requests to vehicles. Constraint 4.14 ensures that every order request r is completed by one vehicle k. Constraint 4.15 prevents order requests r from being assigned to vehicles that are not operating by linking the assignment variable u_{rk} with the vehicle usage variable y_k.

$$\sum_{k \in V} u_{rk} = 1 \qquad \forall r \in R \tag{4.14}$$

$$u_{rk} \leq y_k \qquad \forall r \in R,\ \forall k \in V \tag{4.15}$$

Each order request consists of a set of required components p, which depend on the order type τ_r. Constraint 4.16 ensures that all required components of an order request are executed exactly once, at one node and by one vehicle. The binary variable v_{rpik} indicates whether component p of order r is handled by vehicle k at node i, and the set of feasible nodes is defined by the candidate nodes. Furthermore, to maintain consistency between order assignment and component execution, constraint 4.17 imposes that if a vehicle handles any component of an order request, then that order request must be assigned to the same vehicle.

$$\sum_{k \in V} \sum_{i \in \mathcal{N}_{rp}} v_{rpik} = 1 \qquad \forall r \in R,\ \forall p \in P :\ \delta_{\tau_r p} = 1 \tag{4.16}$$

$$v_{rpik} \leq u_{rk} \qquad \forall r \in R,\ \forall p \in P,\ \forall i \in \mathcal{N}_{rp},\ \forall k \in V \tag{4.17}$$

When a vehicle visits a dummy node, it must perform at least one component handling activity at that node. Constraint 4.18 prevents vehicles from visiting dummy nodes unnecessarily.

$$z_{ik} \leq \sum_{(r,p) \in \mathcal{C}(i)} v_{rpik} \qquad \forall i \in S' \cup F',\ \forall k \in V \tag{4.18}$$

Finally, constraint 4.19 enforces the required precedence order of the components within the same order request. If vehicle k is assigned to order r, it can only arrive at the node handling component p' after completing the service of the preceding component p, including the total service time at the node and the travel time between nodes.

$$a_{jk} \geq a_{ik} + s_{rp} + t_{ij} - M\,(2 - v_{rpik} - v_{rp'jk}) \qquad \forall r \in R,\ \forall (p,p') \in P_{\tau_r},\ \forall i \in \mathcal{N}_{rp},\ \forall j \in \mathcal{N}_{rp'},\ \forall k \in V \tag{4.19}$$

Time constraints

The following constraints ensure the temporal feasibility of the vehicle routes and enforce the daily operational time limit. Constraint 4.20 fixes the start time of each vehicle at the depot to zero. The service time at each node depends on the order in which components are handled by a vehicle at that node. Constraint 4.21 defines the node-specific service time st_{ik} as the sum of the handling times of all components executed by vehicle k at node i. If no component is handled at a node, the service time is zero.

$$a_{Dk} = 0 \qquad \forall k \in V \tag{4.20}$$

$$st_{ik} \;=\; \sum_{(r,p)\in\mathcal{C}(i)} s_{rp}\, v_{rpik} \qquad \forall i \in N,\ \forall k \in V \tag{4.21}$$

Constraint 4.22 enforces the maximum daily operating time. If vehicle k returns directly to the depot after visiting node j, the arrival time at node j, plus the corresponding service time and the travel time back to the depot, must not exceed the maximum allowable operating time $T^{\max}$.

$$a_{jk} + st_{jk} + t_{jD} \;\le\; T^{\max} + M\,(1 - x_{jDk})\,, \quad \forall j \in N \setminus \{D\},\ \forall k \in V \tag{4.22}$$

Temporal consistency along the route of each vehicle is enforced by constraints 4.23 and 4.24. If vehicle k travels directly from node i to node j, the arrival time at node j must be equal to the arrival time at node i plus the service time at node i and the travel time between the two nodes. The constraints are implemented as indicator constraints, meaning that they are only activated when arc (i, j) is selected by vehicle k (i.e., when $x_{ijk} = 1$). Both a lower and an upper bound are included to enforce equality of the arrival times. This formulation prevents artificial waiting time at nodes and ensures temporal consistency along the selected route.

Indicator constraints are used here because the time relation only matters when a vehicle actually travels from node i to node j. This allows the solver to directly activate or deactivate the constraint based on the routing decision. Big-M formulations are still used in other parts of the model where needed, but for these temporal constraints, indicator constraints provide a clearer and more stable implementation.

$$a_{jk} \ge a_{ik} + st_{ik} + t_{ij} \qquad \forall i, j \in N,\ i \ne j,\ j \ne D,\ \forall k \in V:\ x_{ijk} = 1 \tag{4.23}$$

$$a_{jk} \le a_{ik} + st_{ik} + t_{ij} \qquad \forall i, j \in N,\ i \ne j,\ j \ne D,\ \forall k \in V:\ x_{ijk} = 1 \tag{4.24}$$

Capacity constraints

The following constraints ensure that vehicle capacity limitations are respected throughout each route. Constraint 4.25 limits the total number of vehicles that may be used to the maximum available number $V^{\max}$. Each vehicle is subject to capacity limits on both the total number of containers and the number of full containers. Constraints 4.26 and 4.27 enforce non-negativity and upper bounds on the total and full container loads at every node. Constraint 4.28 ensures that the number of full containers never exceeds the total number of containers carried by a vehicle.

$$\sum_{k\in V} y_k \le V^{max} \tag{4.25}$$

$$0 \le q_{ik}^{tot} \le C_k^{tot} \qquad \forall i \in N,\ \forall k \in V \tag{4.26}$$

$$0 \le q_{ik}^{full} \le C_k^{full} \qquad \forall i \in N,\ \forall k \in V \tag{4.27}$$

$$q_{ik}^{full} \leq q_{ik}^{tot} \qquad \forall i \in N,\ \forall k \in V \tag{4.28}$$

Constraint 4.29 initialises the vehicle load at the depot to start at zero. Changes in vehicle load along the route are determined by the handling activities performed at each visited node. For every arc (i, j) crossed by vehicle k, constraints 4.30 and 4.31 update the total container load by accounting for the net load change resulting from all components executed at node i.

$$q_{Dk}^{tot} = 0,\ q_{Dk}^{full} = 0 \qquad \forall k \in V \tag{4.29}$$

$$q_{jk}^{tot} \geq q_{ik}^{tot} + \Delta_{ik}^{tot} - M(1 - x_{ijk}) \qquad \forall i, j \in N,\ i \neq j,\ \forall k \in V \tag{4.30}$$

$$q_{jk}^{tot} \leq q_{ik}^{tot} + \Delta_{ik}^{tot} + M(1 - x_{ijk}) \qquad \forall i, j \in N,\ i \neq j,\ \forall k \in V \tag{4.31}$$

Similarly, constraints 4.32 and 4.33 track the change of the load of full containers along the route.

$$q_{jk}^{full} \geq q_{ik}^{full} + \Delta_{ik}^{full} - M(1 - x_{ijk}) \qquad \forall i, j \in N,\ i \neq j,\ \forall k \in V \tag{4.32}$$

$$q_{jk}^{full} \leq q_{ik}^{full} + \Delta_{ik}^{full} + M(1 - x_{ijk}) \qquad \forall i, j \in N,\ i \neq j,\ \forall k \in V \tag{4.33}$$

Together, these constraints define the benchmark MILP model for the skip container VRP. The model captures the main routing, assignment, precedence, timing, and capacity restrictions required for standard skip container operations. Because the benchmark is formulated under simplified assumptions, it can be used to evaluate the performance of the extended model. However, including detailed stacking rules and container reuse would substantially increase the complexity of the exact formulation. Therefore, the next section introduces a metaheuristic algorithm designed to address these additional operational aspects.

4.2. Metaheuristic Algorithm

The extended formulation substantially increases the complexity of the routing problem compared to the benchmark MILP model. In particular, the reuse decision variables scale in the order of $\mathcal{O}(|R|^2|V|)$, since for each pair of compatible requests and each vehicle, a potential reuse relation must be evaluated. Furthermore, the detailed stacking representation introduces binary variables that scale with $\mathcal{O}(|V| \cdot S_k^{spot} \cdot H^{max} \cdot |CT|)$, as each vehicle position, stacking level, and container type must be explicitly tracked. As a result, the exact formulation becomes difficult to solve for realistic problem sizes within acceptable computational times.

To solve the extended model, this research applies a Large Neighbourhood Search (LNS) metaheuristic. LNS is well suited for routing problems with complex and interacting constraints because it combines diversification and intensification through a destroy-and-repair mechanism (Pisinger & Ropke, 2019). In the extended skip container VRP, feasibility depends not only on customer sequencing, but also on interactions between order components, stacking configurations, and reuse compatibility. Consequently, small local route modifications often lead to infeasible solutions or only limited improvements. The destroy-repair mechanism of LNS allows larger parts of the route structure to be reconstructed simultaneously, making it more suitable for this problem setting. Other metaheuristics commonly applied in the routing literature, as discussed in section 4.2, include TS and VNS. However, these approaches mainly rely on incremental route modifications, which are less effective when routes contain strongly dependent order components and complex feasibility interactions.

The suitability of LNS for skip container routing problems is also demonstrated in earlier RR VRP research. Wy et al. (2013) developed an LNS based heuristic for a RoRo waste collection problem with time windows. Their problem formulation includes several operational constraints relevant

to this thesis, such as multiple disposal facilities, multiple storage locations, different container sizes and service types, customer-facility eligibility restrictions, and multiple demands at a single customer location. In addition, their solution approach combines a construction heuristic with several improvement procedures and a ruin-and-recreate based search mechanism. Their computational experiments further show that feasible solutions can be generated in relatively short computational times, even for large and highly constrained problem cases (Wy et al., 2013). This demonstrates that LNS can effectively handle routing problems in which feasibility depends on multiple interacting operational constraints.

Before discussing the individual algorithmic components, the general methodological framework of the metaheuristic is first introduced in subsection 4.2.1. The development of an LNS requires several design steps to be considered. Osaba et al. (2021) propose a framework that can be used as a guide for developing a metaheuristic. The steps are as follows:

1. **Problem modelling and mathematical formulation**: focus is on the modelling and mathematical formulation of the optimisation problem
2. **Algorithmic design, solution encoding and search operators**: focus is on the design and implementation of the metaheuristic algorithm
3. **Performance assessment, comparison and replicability**: focus is on the correct evaluation of the algorithm development and on the replicability and consistency of the research
4. **Algorithmic deployment for real-world applications**: focus is on the deployment of the method in a real environment

Within this workflow, the benchmark model already provides a foundation for the mathematical framework, which must now be extended to incorporate the additional aspects discussed in subsection 3.1.2. The following sections describe the conceptual structure of the algorithm, the solution representation, and the corresponding design choices, with a primary focus on the second step of the framework proposed by Osaba et al. (2021).

4.2.1. Design Framework

In order to give more detail about the LNS model, a conceptual model of the metaheuristic will be presented. First, a flowchart will be shown in Figure 4.1, and following this, the pseudocode will be presented in Algorithm 1. After these, the different parts of the framework will be discussed in more detail, along with the design choices made.

The general flow in the flowchart is as follows. Customers are clustered (subsection 4.2.2) and used to construct initial routes (subsection 4.2.3). These routes are subsequently improved using improvement rules (subsection 4.2.4). Following this, the destroy-repair cycle begins, with operators selected based on a given probability distribution (subsection 4.2.5). The resulting candidate solution may be accepted as the current solution and/or the best solution (subsection 4.2.6). If the stopping criteria have not yet been met, the model continues the destroy–repair cycle (subsection 4.2.7).

Flowchart

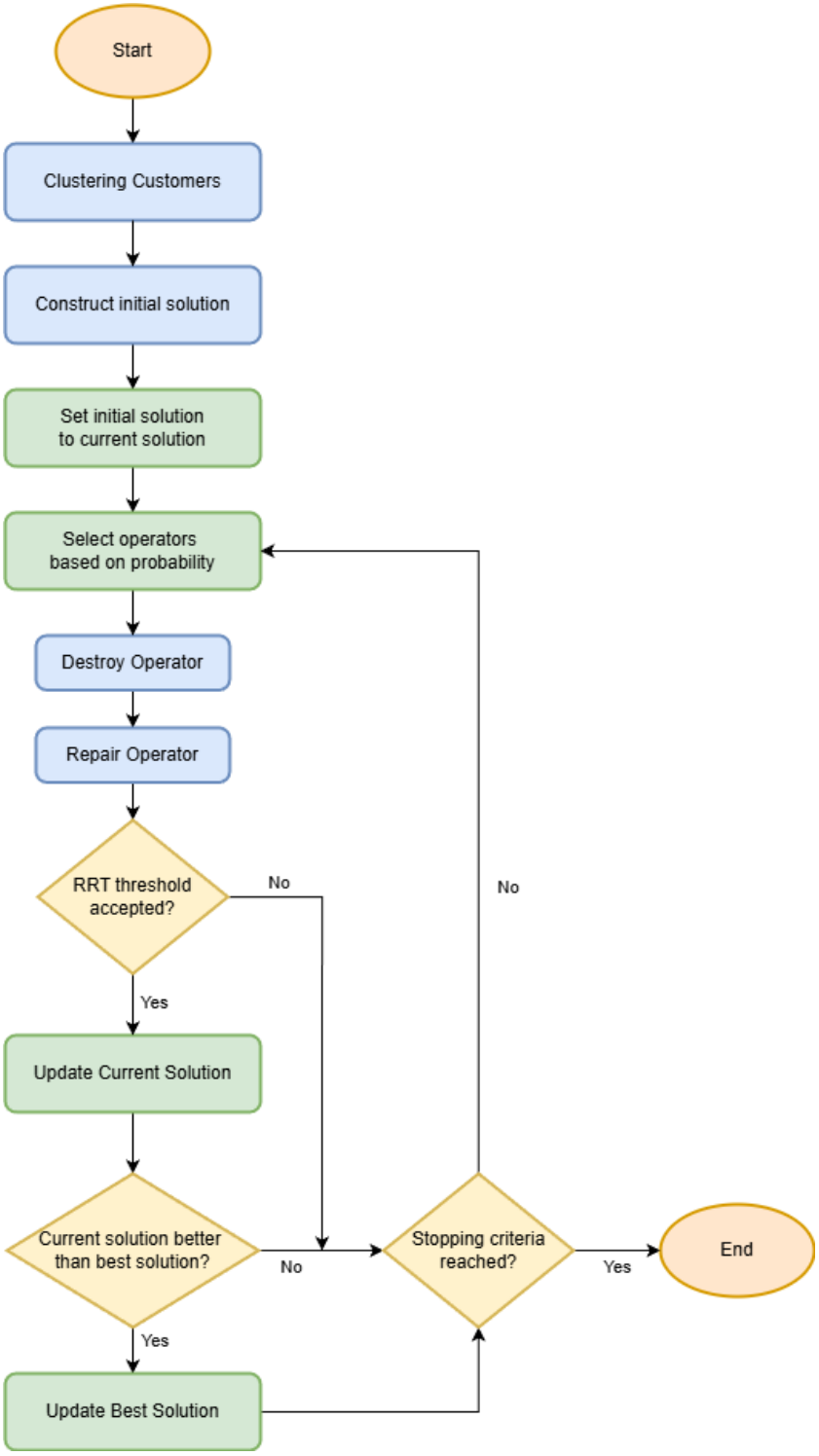


Figure 4.1: Flowchart of the LNS metaheuristic

Pseudocode

Algorithm 1 Large Neighbourhood Search with Linear RRT

Input: feasible initial solution x ▷ Uses clustering-based insertion
Input: Δ_0 (initial threshold), K (maximum iterations), R (maximum rejections)
$x^b \leftarrow x, \quad x^c \leftarrow x$
$i \leftarrow 1, \quad r \leftarrow 0$
while stopping criterion not met **do**
  $(x', feasible) \leftarrow \text{REPAIR}(\text{DESTROY}(x^c))$
  if not feasible **then**
    $r \leftarrow r + 1$
    $i \leftarrow i + 1$
    continue
  end if
  $\Delta \leftarrow \Delta_0 \left(1 - \frac{i}{K}\right)$ ▷ Linear RRT update
  if $c(x') < c(x^b)$ **then**
    $x^b \leftarrow x', \quad x^c \leftarrow x'$ ▷ Update best and current
    $r \leftarrow 0$
  else if $\frac{c(x') - c(x^b)}{c(x')} \leq \Delta$ **then**
    $x^c \leftarrow x'$ ▷ Accept candidate
    $r \leftarrow 0$
  else
    $r \leftarrow r + 1$
  end if
  $i \leftarrow i + 1$
end while
return x^b

4.2.2. Clustering

In order to combine customer locations that are geographically close, clustering is applied. The clustered customer groups are applied in different parts of the algorithm. First, clustering is used to support the construction of the initial solution, so that the algorithm starts with routes that are already geographically close. Furthermore, clustering is used in the improvement rules, as discussed later in subsection 4.2.4.

The selected clustering method is k-means clustering. This method is chosen because it is relatively simple, computationally efficient, and suited to spatial data (Shalev-Shwartz & Ben-David, 2014). In k-means, the data points are partitioned into k different clusters, where each customer is assigned to the cluster with the nearest centroid. The method, therefore, provides a straightforward way to group nearby customer locations before route construction.

In this study, the number of clusters is determined in advance as a design choice to keep the analysis simpler. Since clustering is only used as a supporting step in the LNS and not as the main optimisation objective, it is not necessary to add another layer of complexity by dynamically estimating the optimal number of clusters. The value of k is set to the minimum of either the number of vehicles or half the number of orders. In most cases, this results in k being equal to the number of vehicles, although for the smaller validation subsets, it may instead depend on the number of orders. This way, the number of clusters remains limited, while still reflecting the expected routing structure. After the initial clustering step, clusters containing fewer than three customers are merged with a nearby cluster until each cluster contains at least three customer locations. This prevents the creation of very small clusters, which negates the point of using clusters. Figure 4.2 shows an example of the Case 1B data set and how the clustering result can look like.

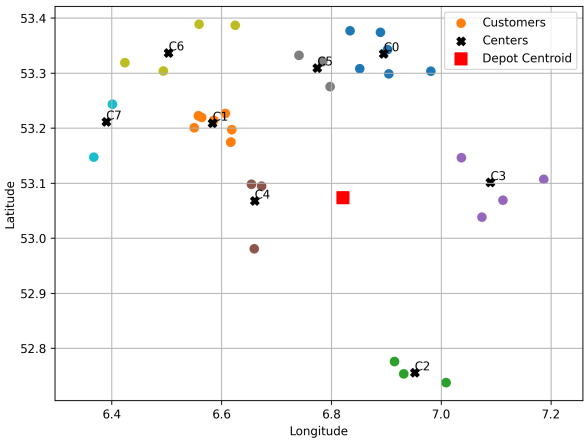


Figure 4.2: Customer clusters of the data set Case 1B.

4.2.3. Initial Solution

The initial solution for the LNS is constructed using a regret-k insertion heuristic. As discussed in subsection 2.6.2, constructive heuristics iteratively build feasible routes and are commonly used to initialise neighbourhood-based metaheuristics. Regret-based insertion introduces a limited look-ahead by selecting the request with the largest difference between its best and second-best insertion options, thereby anticipating future difficulties rather than focusing solely on immediate cost. As shown by Ropke and Pisinger (2006), this approach performs well for tightly constrained PDPs. In this study, the regret-2 variant ($k = 2$) is applied, as it provides a good balance between computational efficiency and solution quality. Larger values of k mainly increase computational effort without significant improvements. Since the initial solution only serves as a feasible starting point for the destroy-repair cycles, $k = 2$ is considered sufficient. The solution representation consists of a sequence of visited locations for each vehicle, together with the corresponding component executed at each location. Orders are inserted sequentially as groups of precedence-related components, ensuring that the operational dependencies between activities remain satisfied throughout the insertion process.

This specific implementation follows a route-based regret insertion heuristic (Qu & Bard, 2012). The order of insertion is determined using the clustering approach discussed in subsection 4.2.2. Using a sweep algorithm, the clusters are processed sequentially around the depot centroid in a counter-clockwise direction. For each cluster, the corresponding orders are evaluated using the regret insertion heuristic while continuously comparing the available end of vehicle routes. Initially, only existing routes are considered. A new route, and therefore an additional vehicle, is introduced only when no feasible insertion into the current routes can be found. Since the components are added at the end of a route, the regret evaluation compares alternative vehicle routes rather than multiple insertion positions within the same route. During insertion, the maximum route duration is temporarily extended by a buffer of 30 minutes. This additional flexibility supports the construction of a feasible initial solution and is expected to be reduced later during the improvement phase.

All candidate insertions are evaluated with respect to the feasibility constraints of the model, including vehicle capacity limits, reuse feasibility, precedence order of components, and stacking constraints where applicable. Only feasible insertions are taken into account when determining the best and second-best alternatives. The complete heuristic is described in Algorithm 2 in Appendix D.

4.2.4. Improvement Rules

An additional step is introduced after all insertions are complete, namely the improvement rules. These rules aim to smartly combine components from other orders to reduce the operational time. The use of additional improvement procedures after the construction or insertion steps is also in line with an earlier LNS based skip container routing model. Wy et al. (2013) combine

their LNS framework with several improvement algorithms, including service-type changing, inter-route improvement, intra-route improvement, and tractor-changing procedures. This shows that the search process can benefit from extra route improvement logic in addition to the main neighbourhood search.

In this research, four improvement rules are applied after the regret-2 heuristic has been executed in the initial solution, and they are also applied during the repair phase. However, rather than focusing on service types or tractor assignments as in Wy et al. (2013), the rules are designed to combine the same type of order components, eliminate unnecessary visits, and improve the sequence of components within a route. For the first three rules, there is an overlapping approach governing how components may be repositioned when an improvement is possible. These rules focus on combining customers' visits in the same cluster, as well as visits to disposal facilities and storage locations. When a combination opportunity arises, there are two possible ways in which the route may be adjusted: either at an earlier location in the route (location A) or at a later location (location B). If combining at location A results in the biggest improvement, the component to be combined is moved earlier in the route, together with its predecessors, in order to preserve precedence relationships. This scenario is illustrated in Figure 4.3, where a change order and a pickup order are combined based on the use of the same disposal facility F.

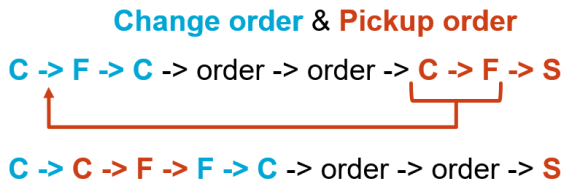


Figure 4.3: Combining a change order and a pickup order at location A.

Alternatively, the largest time reduction can be achieved by combining the operations at location B, as illustrated in Figure 4.4. In this case, the disposal facility visit F from A is inserted directly after the disposal facility visit F at B. This allows the successor component of order A to be positioned immediately afterwards.

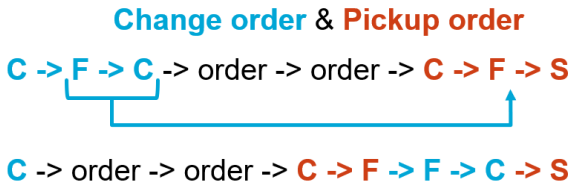


Figure 4.4: Combining a change order and a pickup order at location B.

It is also possible to combine more than two orders simultaneously. In such cases, three types of blocks are constructed: a block containing the disposal facility visits F that are candidates for combination, a block consisting of all predecessor components that must be moved to preserve feasibility, and a block of successor components that must be repositioned accordingly. These blocks are then jointly relocated within the route to evaluate potential improvements.

To limit the computational effort, the size of these combinations is restricted. For disposal facility combinations, the maximum number of components considered is bounded by the vehicle capacity for full containers. For example, if a vehicle can carry four full containers, at most four components are evaluated simultaneously for merging. A similar principle applies to storage location S, but with an important distinction. Orders are typically restricted to one or two disposal facilities, as will later be discussed in section 5.2, whereas they may be compatible with a much larger number of storage locations. Allowing all possible combinations for S would therefore lead to a combinatorial explosion. To prevent this, the model only considers storage locations that are

already visited within the route. For each such location, it identifies the orders that are allowed to visit it and evaluates combinations of at most two or three components.

Checking Reuse

The reuse rule differs from the previous rules by focusing on matching container flows between orders within a route. It identifies overlap in container types between components of type *Delivery_P* and *Dropoff*, which correspond to collecting empty containers from a storage location and returning empty containers to a storage location, respectively (see Equation 4.1).

When the container quantities match exactly, both storage visits can be skipped entirely. In cases where only partial reuse is possible, the remaining quantity is adjusted, and a storage visit is still required to handle the surplus or deficit. Additionally, if the components are not in the correct sequence (i.e. the *Dropoff* should precede the *Delivery_P*), the algorithm evaluates whether repositioning them leads to an improvement. If reuse is applied, both travel and service times are reduced. The route is re-evaluated without the unnecessary storage visits, and the service time is adjusted to reflect the reduced handling time.

4.2.5. Search Operators

The effectiveness of an LNS is strongly influenced by the selected destroy and repair operators. Within an LNS framework, destroy operators partially remove a set of orders from the solution, while repair operators reconstruct it by reinserting them. As discussed in Table 2.6.2, these operators define the neighbourhood structure and therefore determine the balance between intensification, which explores promising solution regions, and diversification, which explores new regions. Following the recommendations of Ropke and Pisinger (2006) and the systematic review of Voigt (2025), a limited but well-balanced set of complementary operators is selected. For example, the ruin-and-recreate strategy applied in the RR VRP by Wy et al. (2013) removes a random subset of customers and reinserts them using a best-insertion procedure. This confirms that even relatively simple destroy-repair combinations can be effective for complex skip container routing problems.

Destroy

Destroy operators remove a subset of requests from the current solution to define a large neighbourhood. The review paper by Voigt (2025) recommends starting with a standard set of operators and expanding only if necessary. Based on these recommendations, these destroy operators are selected:

- **Random removal** (diversification): A random subset of requests is removed from the solution (Ropke & Pisinger, 2006).
- **Worst-cost removal** (intensification): Requests that contribute the largest marginal cost to their current route are removed first (Ropke & Pisinger, 2006)

The random removal operator is based on a destroy fraction that randomly removes a percentage of the order. The destroy fraction is tuned during the parameter tuning phase (see section 5.3). This introduces a trade-off: removing a larger fraction allows for greater exploration of different solution combinations, but it also increases the number of orders that must be reinserted, thereby increasing the duration of the repair phase.

The worst-cost removal is determined by evaluating the impact of removing an entire order from a route. Specifically, the operator identifies the order whose removal yields the greatest reduction in operational time. Since orders may consist of different numbers of components, the resulting time reduction is normalised by dividing it by the number of components within the order. Based on this normalised value, orders are ranked and the worst cost is determined accordingly. The pseudocode of these operators can be found in Appendix D.

Repair

Repair operators reconstruct a feasible solution by reinserting the removed orders into the partial solution. Classical route construction methods insert customers sequentially in a position that minimises additional travel cost while maintaining feasibility (Bräysy & Gendreau, 2005). These

greedy insertion strategies are computationally efficient and primarily promote intensification, as they repeatedly exploit the best insertion positions locally.

In the context of LNS, more advanced insertion heuristics have been shown to improve overall performance. The review paper by Voigt (2025) highlights that regret-based insertion heuristics consistently outperform purely greedy strategies across many vehicle routing variants. Similar to the destroy phase, the review paper recommends two complementary repair operators as a standard basis:

- **Customer with highest position regret** - at best position (intensification) (Qu & Bard, 2012)
- **In random order** - at best position (diversification) (Lei et al., 2011)

The first operator applies a regret-based selection criterion, similar to the initial solution construction, but based on position regret rather than route regret. To limit the number of evaluated insertions, only insertion positions between orders are considered, thereby preventing insertion between components belonging to the same order. Furthermore, a precomputed internal service time is used to exclude routes that would exceed the maximum operational time when inserting the order. Possible positions across all vehicles, with operational time left, are considered together. When the position is chosen, the route is reconstructed. Furthermore, a time buffer of 30 minutes is allowed on top of the maximum operational time, consistent with the initial solution construction. After all insertions have been completed, the improvement rules described in subsection 4.2.4 are applied to further improve the solution.

The second operator shuffles the removed orders into a random sequence and inserts each order at its best feasible position across the available vehicles. The same restrictions on insertion positions and feasibility filtering based on internal service time are applied. While each insertion is locally time minimising, the randomness introduces diversification, preventing the search from repeatedly selecting similar insertion patterns and thereby improving exploration of the solution space. The pseudocode of these operators can be found in Appendix D.

Search Procedure

Instead of using an adaptive operator selection as in a full ALNS framework, a simpler approach is applied in this study. At each iteration, both the destroy and repair operators are selected with equal probability of 50%, resulting in a uniform random choice between the available operators. This choice is motivated by both methodological and practical considerations. In early-stage metaheuristic design, simple algorithmic structures are often sufficient, and increasing algorithmic complexity does not necessarily lead to improved performance. In fact, it has been shown that robust optimisation methods can be constructed using relatively simple mechanisms (Osaba et al., 2021).

Furthermore, randomness in the operator selection process naturally contributes to diversification, which is an important aspect of effective large neighbourhood search. Diversification can, for example, be achieved through random removal strategies, while similar effects also arise when operators are selected uniformly at random (Voigt, 2025). A dynamic weighting procedure was also tested, but this did not lead to an improvement in solution quality. In addition, an extra destroy operator based on the least presence of a cluster within a route was evaluated. However, this mainly increased the computational time without improving the objective value. Therefore, the fixed 50-50 chance is applied.

Finally, adaptive operator selection mechanisms are not guaranteed to improve performance and may depend heavily on parameter tuning and operator configuration. It has been noted that relying solely on adaptive selection can lead to poor performance if the initial operator set is not well chosen (Voigt, 2025). In contrast, a uniform selection strategy avoids these issues and remains effective when combined with well-designed operators.

4.2.6. Acceptance Criterion

Besides the destroy and repair operators, the acceptance criterion is another important component of an LNS, as it determines whether newly generated solutions are accepted during the search process. The simplest acceptance strategy is to only accept improving solutions, which

is also referred to as hill climbing (HC) (Shaw, 1997). However, this approach is likely to get stuck in local optima, as it does not allow for any worsening moves. This is also observed by Santini et al. (2018), who show that both HC and random walk (RW), where all new solutions are accepted, consistently perform worse than more advanced acceptance criteria within an (A)LNS framework.

To allow the search to explore beyond local optima, a record-to-record travel (RRT) acceptance criterion is applied. Originally introduced by Dueck (1993), RRT evaluates candidate solutions relative to the best solution found so far. At each iteration, the destroy and repair operators generate a candidate solution x' from the current solution x. The acceptance criterion then determines whether x' replaces the current solution. In line with Santini et al. (2018), a relative acceptance formulation is used. A candidate solution x' is accepted if

$$\frac{c(x') - c(x^b)}{c(x')} \leq \Delta_i, \tag{4.34}$$

where x^b denotes the best solution found so far and Δ_i is the deviation threshold at iteration i. This formulation follows the relative deviation variant described by Santini et al. (2018), where the deviation is normalised by $c(x')$. As a result, the acceptance decision does not depend on the absolute scale of the objective function, which is beneficial when solving data sets with different characteristics. In this study, a linear RRT variant is employed, where the deviation threshold decreases linearly over time according to

$$\Delta_i = \Delta_0 \left(1 - \frac{i}{K}\right), \tag{4.35}$$

where Δ_0 is the initial threshold and K is the maximum number of iterations. Consequently, Δ_i decreases from Δ_0 to 0, allowing larger deviations in early iterations to promote diversification, while gradually enforcing stricter acceptance to intensify the search. Furthermore, improving solutions are always accepted, i.e., if $c(x') < c(x^b)$, the candidate solution replaces both the current solution and the best solution found so far. Infeasible solutions are rejected. The pseudocode for this process can be found in Appendix D.

Compared to simulated annealing (SA), which accepts worse solutions with a probability depending on a temperature parameter, RRT applies a deterministic threshold-based rule. Santini et al. (2018) show that RRT performs competitively across multiple problems and is never statistically dominated by alternative acceptance criteria. However, for CVPRs, RRT outperforms SA. Therefore, RRT is chosen over SA, despite being a commonly used method.

4.2.7. Stopping Criterion

Lastly, the stopping criterion determines when the search process is stopped. In LNS, it is common to combine multiple stopping conditions to balance solution quality and computational effort. In this research, three criteria are applied: an iteration limit, a time limit, and a rejection limit based on stagnation. The iteration limit controls the total search effort, the time limit ensures practical runtimes, and the rejection-based criterion stops the search when no improvement is observed during a sequence of iterations. The criteria are defined as follows:

- **Maximum number of iterations:** The algorithm is stopped after a predefined number of destroy-repair cycles. This is one of the most commonly applied stopping rules in LNS literature (Ropke & Pisinger, 2006; Shaw, 1997).
- **Maximum computational time:** The search is stopped once a predefined time limit is reached. This criterion is widely used in large-scale vehicle routing problems (Pisinger & Ropke, 2019).
- **Maximum number of sequential rejections:** Candidate solutions are evaluated using the acceptance criterion described in subsection 4.2.6. Each rejected candidate solution, including infeasible solutions, increases a rejection counter. Whenever a candidate solution is

accepted, the counter is reset to zero. The search stops once a predefined number of consecutive rejections has been reached (Wy et al., 2013).

4.3. Key Performance Indicators

To evaluate and compare the proposed routing models, a set of KPIs is defined. In VRP research, model performance is typically reviewed using a combination of operational objectives, utilisation measures, and algorithmic indicators. Braekers et al. (2016) shows that a wide range of objective functions are used in the literature, with routing cost measures based on travel time or distance, vehicles related measures, and problem-specific indicators being the most common categories. This indicates that a single KPI is not sufficient and that multiple indicators are required to evaluate routing models in a comprehensive manner (Braekers et al., 2016).

Therefore, the KPIs selected for this research are grouped into three performance categories: operational efficiency, utilisation of vehicles and containers, and algorithmic performance. As this research introduces container stacking and direct container reuse, additional problem-specific KPIs are included to check whether these modelling extensions result in practical improvements. An overview of the selected KPIs is presented in Table 4.2. Each KPI is discussed in detail below.

Table 4.2: Overview KPIs.

KPI	Unit	Type
Total operational time	minutes	Objective
Number of vehicles used	count	Operational
Container reuse rate	%	Operational
Average vehicle load per arc	count	Operational
Average stacking utilisation	%	Operational
Travel time per container	minutes	Operational
Orders per hour	count	Operational
Number of iterations	count	Algorithmic
Computational time	seconds/minutes	Algorithmic

The **total operational time** is defined as the sum of driving time and handling time at the different locations for the different vehicles. Because the duration a vehicle is in operation directly and significantly impacts operational costs, this indicator is used as a primary performance measure.

The **number of vehicles** required to serve all orders is included as a KPI to evaluate fleet utilisation. This indicator is directly related to fixed operational costs and provides insight into the efficiency with which the available fleet is used. Furthermore, this has also been applied in RR VRP as well as drayage studies (Bodin et al., 2000; Zhang et al., 2020).

The **container reuse rate** measures the percentage of orders in which an empty container is directly reused without being returned to a depot. This KPI captures how frequently (partial) reuse is applied. Reducing unnecessary empty container movements is an identified gap in existing routing models and is therefore an important KPI.

The **average vehicle load per arc** represents the number of containers carried by a vehicle while driving between locations. An increase in this value indicates that vehicles transport more containers during their routes, which can suggest a more effective use of available capacity and potentially more efficient routing decisions enabled by stacking.

The **average stacking utilisation** measures the extent to which the available stacking capacity is used relative to its maximum capacity. When evaluated alongside the average vehicle load per arc, this KPI helps determine whether higher vehicle loads are achieved through improved use of stacking capacity or simply through structural changes in the routes.

The **travel time per container** measures the total operational time divided by the total number of containers. The purpose of this KPI is to evaluate routing efficiency from a container-level per-

spective, independent of vehicle-level performance. In the context of reuse, it indicates whether containers are, on average, moved with less travel when depot returns are avoided. When interpreted together with the container reuse rate, this KPI supports the assessment of whether increased reuse leads to efficient container movements rather than additional transport.

The **orders per hour** KPI is considered an important performance indicator by AMCS. Defining the exact start and end time of individual orders is not straightforward. Therefore, this metric is based on the total operational time relative to the number of orders. The number of orders refers to the original set of customer requests prior to any preprocessing steps, such as merging orders or splitting swap orders (see section 5.1). This KPI provides a clear and comparable measure of routing efficiency across different data sets. Ultimately, the primary objective is to serve customers, and this indicator offers direct insight into how efficiently that objective is achieved.

Finally, the last two KPIs are algorithmically focused: **computational time**, which measures the runtime required to obtain a solution, and **number of iterations**, which indicates the number of iterations needed for the runs. These are standard KPIs widely applied across a broad range of metaheuristics. Computational time also helps compare the practical applicability of the benchmark model with the LNS algorithm.

4.3.1. Verification and Validation

To ensure the correctness and reliability of the proposed models, both the verification and validation procedures will be performed.

Verification focuses on confirming that both the mathematical formulation and the LNS implementation are correctly translated into code. First, all generated solutions are systematically checked for feasibility with respect to capacity limits, stacking constraints, nesting compatibility, precedence relations, and reuse possibilities. In addition, the LNS solutions on small data sets are compared with the corresponding MILP benchmark solutions. This comparison is used both to verify the consistency of the implementation and to evaluate the solution quality of the LNS algorithm.

Validation evaluates whether the behaviour of the model and algorithm is consistent with expected operational patterns. Rather than relying on a single test, validation is performed throughout Chapter 6. The computational results are analysed to verify whether changes in input characteristics, such as demand levels, stacking flexibility, and reuse options, lead to logical and explainable effects on operational performance.

Overall, this chapter presented the methodology for modelling and solving the skip container routing problem. First, the benchmark MILP formulation was introduced to provide a structured mathematical representation of the problem and to serve as a reference for evaluating the metaheuristic approach. Afterwards, the design of the LNS algorithm was discussed, including the construction of the initial solution, the operators, the acceptance criterion, and the stopping conditions. Together, these components form the basis for the computational experiments and result analysis presented in the following chapters.

5

Experimental Setup

This chapter outlines the preceding steps required before running and evaluating the results. The process begins with data preprocessing, as described in section 5.1. Next, the experimental design and testing setup are discussed in section 5.2. Finally, the parameter tuning process and the resulting choices are presented in section 5.3. Once these steps are completed, the setup is ready for running the experiments, which will be covered in the next chapter.

5.1. Data Preprocessing

Benchmark model

Before using the data, several preprocessing steps were applied. First, order requests from the same customer and having the same order type were combined. In addition, the data sets contain multiple candidate locations for vehicle depots, storage locations, and disposal facilities. Therefore, three representative locations were manually selected for the analysis. A maximum of two vehicles was included in the model to limit the solution space. Since the selected locations are geographically close, most test cases can be serviced by a single vehicle, with a second vehicle available for larger data sets. For the dummy variables, the maximum number of dummies was determined based on the order types included. For example, a swap order can use a storage location at most twice: once to pick up a container at the start of the order and once to return a container at the end. Consequently, when a swap order is included in the model, two dummy storage location nodes are added. Lastly, the allowed optimality gap was set at 5% to reduce the computational time.

LNS algorithm

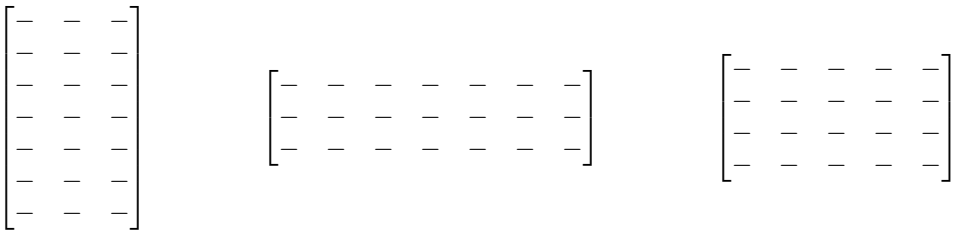
For the LNS algorithm, several adjustments similar to the benchmark model, such as order merging, are applied. Due to the larger number of orders, merging is constrained by vehicle capacity, which is dynamically determined by the selected configuration (stacking spots and stacking height), to ensure feasibility. Additionally, swap orders are split into two separate orders to increase flexibility. Unlike the optimisation model, these two sets of components are not required to be executed by the same vehicle. Each swap order is divided into a swap_delivery (first two components) and a swap_pickup (the remaining three components). This allows independent assignment, improving routing flexibility and potentially solution quality. However, collected containers must still be handled in a single visit, no further splitting is allowed.

Finally, manual adjustments were made to the data sets. For example, in the Case 8 data set,

some orders were originally assigned to disposal facilities in Belgium or Germany. To keep transportation within one country, they were reassigned to the nearest disposal facility.

5.2. Experimental Design LNS

This section describes the experimental setup used to in the proposed algorithm. In addition to the parameters that will be tuned during the computational experiments, several fixed modelling and algorithmic parameters are defined. These settings are selected to reflect realistic operational conditions while maintaining computational tractability. First, the vehicle configuration is discussed. As highlighted in the limitations of the benchmark model (see subsection 6.1.3), the transportation of full containers forms an important operational bottleneck within the benchmark formulation. To research the extent of this effect within the LNS algorithm, different vehicle configurations will be applied. Since the top row represents the capacity for full containers, changing this dimension directly affects the number of full containers that can be transported. The vehicle configuration is defined as the number of stacking spots multiplied by the stacking height. The data sets originally use a configuration of 3×7 is applied, therefore, this configuration is also used as a baseline. To provide a contrast, the inverse configuration, 7×3, is considered. In addition, a third configuration of 5×4 is introduced to represent a middle ground. Although this results in a total capacity of one container less, it is not expected to have a significant impact on the final results. A visual representation of these vehicle configurations is provided below.



Furthermore, some temporal values are fixed. The total operational time of a day is 540 minutes (9 hours). Moreover, the (un)load time of an empty container is 3 minutes and the (un)load time of a full container is set to 10 minutes.

Data Sets LNS

To further evaluate the LNS algorithm, a larger set of data sets is used. Table 5.1 presents the occurrence of the different order types within these data sets, while Table 5.2 summarises the number of vehicles, depots, disposal facilities, and storage locations, as well as the average number of allowed facilities per order component. Both tables describe the raw input data, this is before the merging of orders and the splitting of swap orders into separate delivery and pickup orders. In Appendix C, maps of all data sets are provided to give a visual representation of the geographical distribution of the locations.

In addition, two manually created test data sets are included in which only one container type is available. These cases are used to further assess the impact of container reuse. These data sets are selected because they contain relatively few change orders, as reuse cannot be applied in such cases. Moreover, they differ in the average number of orders, allowing the experiments to capture different operational scenarios.

Table 5.1: Overview of data set characteristics.

Data set	Total orders	Pickup	Delivery	Swap	Change	Total containers	% Small containers	% Large containers	Container types
Case 1A	30	1	1	17	11	42	81%	19%	8
Case 1B	40	10	10	19	1	48	98%	2%	11
Case 2A	78	11	4	22	41	82	73%	27%	21
Case 2B	70	13	15	35	7	70	96%	4%	16
Case 3	64	9	9	38	8	76	99%	1%	20
Case 4	57	13	9	29	6	58	100%	0%	14
Case 5	38	3	3	17	15	43	84%	16%	15
Case 6	91	7	7	37	40	109	95%	5%	30
Case 7	28	0	1	1	26	31	100%	0%	8
Case 8	29	0	0	4	25	31	87%	13%	11
Test case Case 3	64	9	9	38	8	76	100%	0%	1
Test case Case 4	57	13	9	29	6	58	100%	0%	1

Table 5.2: Overview of vehicle and location characteristics per data set.

Data set	Vehicles available	Vehicles depots	Disposal facilities	Avg allowed disposal	Storage locations	Avg allowed storage
Case 1A	11	3	9	1.0	2	2.0
Case 1B	9	2	5	1.0	2	2.0
Case 2A	14	3	16	1.0	10	6.5
Case 2B	11	3	11	1.0	10	9.1
Case 3	13	2	8	1.0	16	11.6
Case 4	9	2	8	1.0	10	8.9
Case 5	12	2	12	1.0	10	8.3
Case 6	24	7	13	1.0	10	6.3
Case 7	6	1	10	1.0	11	11.0
Case 8	6	4	11	1.0	12	8.6
Test case Case 3	13	2	8	1.0	16	11.6
Test case Case 4	9	2	8	1.0	10	8.9

Hardware and Software Specifications

The benchmark model was run locally on a laptop equipped with an AMD Ryzen 5 3500U with Radeon Vega Mobile Graphics processor (4 physical cores, 8 logical processors), using up to 8 threads. The benchmark model used Gurobi Optimizer 13.0.0 and Python 3.13.0.

The LNS runs were executed on a SLURM-managed compute server, allowing all experiments to be completed in a single batch without requiring the laptop to remain active throughout the process. The server is equipped with two Intel Xeon Gold 5218 processors running at 2.30 GHz. For each experiment, 4 CPU cores and 8 GB of RAM were allocated. The LNS algorithm used Python 3.10.4.

5.3. Parameter Tuning

The first parameter considered in parameter tuning is Delta0, which represents the initial value in RRT, i.e., how much worse a candidate solution is still accepted as the current solution (see subsection 4.2.6). In addition, the maximum number of iterations and maximum number of rejections are explored. Lastly, the destroy fraction is also varied. Randomness also influences the model; rather than fixing this, it is treated as a tunable parameter.

Each parameter is discussed separately below. No complete factorial exploration is conducted, which means that not all possible combinations are tested. Instead, the parameter tuning begins with Delta0 while keeping the other parameters at their average values. This setup significantly reduces the number of runs and limits computational time.

Three data sets are selected for parameter tuning: Case 1B, Case 3, and Case 7. These are chosen

to reflect a range of characteristics. Case 1B has an average number of orders and relatively few change orders. In contrast, Case 7 is a smaller data set with many change orders. Case 3, the second-largest data set, is included due to its high number of swap orders. The vehicle configuration used is 5×4, because it is the average considering all the configurations that are used to generate the results. The time for (un)loading an empty container is 3 minutes, and for a full container it is 10 minutes. The maximum runtime is set to 900 seconds (15 minutes).

Delta0

The settings for the runs are as follows:

- Delta0 = 0.10, 0.15, 0.20
- Max iterations = 1000
- Max rejections = 100
- Destroy fraction = 0.3

The results in Table 5.3 summarise the average performance across the delta0 tested values, while Figure F.6 in Appendix F visualises the individual runs as a Pareto front. Overall, the *average seconds per iteration* remain consistent across all three delta0 settings, indicating that no configuration is significantly faster or slower. Similarly, the *average total time* shows only a minor variation between the values.

In some cases, delta0 = 0.1 results in the lowest *total time*, whereas for the Case 3 data set, the best value is obtained with delta0 = 0.2. Since no single delta0 value consistently outperforms the others in all metrics, both the *average total time* and the *average number of iterations* are considered when selecting the most suitable value. For Case 1B, delta0 = 0.1 and delta0 = 0.2 result in the same total time, but delta0 = 0.1 achieves this in 150 fewer iterations. A similar pattern can be observed for Case 7, where delta0 = 0.2 and delta0 = 0.15 perform similarly but slightly worse than delta0 = 0.1, while also requiring around 60 additional iterations. The Case 3 data set shows a slightly different trend, with delta0 = 0.2 giving the best overall result. However, delta0 = 0.1 requires only around 0.5% more time while converging in more than 100 less iterations. Therefore, delta0 = 0.1 is selected, as it consistently achieves the best, or near-best, performance while requiring the fewest iterations, thus reducing runtime.

Table 5.3: Performance results for different data sets and delta0 values

delta0	data set	avg total time	std total time	avg iterations	avg runtime (min)	avg seconds per iteration
0.1	Case 1B	2286	57.3	599	1.35	0.135
0.15	Case 1B	2295	26.4	690	1.53	0.133
0.2	Case 1B	2286	56.0	756	1.69	0.134
		2289	**46.6**	**682**	**1.52**	**0.134**
0.1	Case 3	4550	53.8	686	5.53	0.484
0.15	Case 3	4533	63.7	810	6.67	0.494
0.2	Case 3	4524	54.1	802	6.70	0.501
		4536	**57.2**	**766**	**6.30**	**0.493**
0.1	Case 7	1291	7.3	917	0.33	0.022
0.15	Case 7	1292	8.3	980	0.35	0.021
0.2	Case 7	1298	5.5	972	0.34	0.021
		1294	**7.0**	**956**	**0.34**	**0.021**

Maximum Rejections

The settings for the runs are as follows:

- Max rejections = 100, 150, 200
- Delta0 = 0.1 (fixed)
- Max iterations = 1000
- Destroy fraction = 0.3

The *average seconds per iteration* are again almost identical, thus not a decisive factor. When considering the *standard deviation total time*, a different parameter value performs best for each data set. The same applies to the *average total time*, despite the fact that different values perform the best, the differences are also minimal.

However, when examining the *average number of iterations*, a considerable variation between runs can still be observed, similar to the earlier experiment on Delta0. Only in the case of Case 1B was this variation reduced. Overall, the *average total time* values differ only marginally between the tested settings. For Case 7, the difference is even limited to approximately two minutes. For both Case 3 and Case 7, the setting of 100 iterations performs best. Although this is not the case for Case 1B, the value of 100 still outperforms the other settings in terms of the *average number of iterations*.

Across all data sets, increasing the *maximum number of rejections* does not consistently improve solution quality, while it does increase computational effort. Therefore, a value of 100 is selected as a balanced trade-off between solution quality and runtime.

Table 5.4: Performance results for different data sets and max rejection values

max rejections	data set	avg total time	std total time	avg iterations	avg runtime (min)	avg seconds per iteration
100	Case 1B	2280	53.3	665	1.51	0.136
150	Case 1B	2266	59.1	739	1.64	0.133
200	Case 1B	2264	55.5	729	1.63	0.134
		2270	**56.0**	**711**	**1.59**	**0.134**
100	Case 3	4527	86.0	750	6.25	0.500
150	Case 3	4547	63.9	724	5.94	0.493
200	Case 3	4557	85.1	873	7.01	0.481
		4544	**78.3**	**782**	**6.40**	**0.492**
100	Case 7	1293	10.9	898	0.32	0.021
150	Case 7	1293	14.1	958	0.34	0.021
200	Case 7	1295	8.3	982	0.35	0.021
		1294	**11.1**	**946**	**0.34**	**0.021**

Maximum Iterations

The settings for the runs are as follows:

- Max iterations = 750, 1000, 1250
- Delta0 = 0.1 (fixed)
- Max rejections = 100 (fixed)
- Destroy fraction = 0.3

It is immediately noticeable that the *average number of iterations* remains well below the specified maximum in all scenarios. Among the tested data sets, Case 7 consistently comes closest to the maximum allowed *number of iterations*, whereas Case 1B and Case 3 finish substantially earlier.

Increasing the maximum number of iterations from 1000 to 1250 results in only marginal improvements in the quality of the solution. For Case 1B, the difference in *average total time* is approximately one minute, while for Case 7 the improvement is around four minutes. Nevertheless, this is the first parameter setting in which one value consistently outperforms the others across all data sets in terms of *average total time*. Although these improvements are limited, the setting of 1250 iterations still achieves the best overall performance.

Therefore, the *maximum number of iterations* is set to 1250. The higher limit provides additional flexibility and prevents the search process from stopping prematurely in cases where further improvements may still be possible beyond 1000 iterations. Although the algorithm is not expected to consistently reach 1250 iterations, but the higher limit allows the model to continue improving when it is beneficial.

Table 5.5: Performance results for different data sets and max iteration values

max iterations	data set	avg total time	std total time	avg iterations	avg runtime (min)	avg seconds per iteration
750	Case 1B	2288	46.9	424	0.93	0.1323
1000	Case 1B	2262	53.8	658	1.48	0.1350
1250	Case 1B	2261	37.3	691	1.56	0.1355
		2270	**46.0**	**591**	**1.32**	**0.1343**
750	Case 3	4553	46.3	532	4.36	0.4917
1000	Case 3	4574	73.9	709	5.82	0.4925
1250	Case 3	4535	50.5	718	5.70	0.4768
		4554	**56.9**	**653**	**5.29**	**0.4870**
750	Case 7	1296	9.7	713	0.25	0.0211
1000	Case 7	1294	5.9	885	0.31	0.0209
1250	Case 7	1290	7.9	1135	0.41	0.0214
		1293	**7.83**	**911**	**0.32**	**0.0211**

Destroy Fraction

The settings for the runs are as follows:

- Destroy fraction = 0.2, 0.3, 0.4, 0.5
- Delta0 = 0.1 (fixed)
- Max rejections = 100 (fixed)
- Max iterations = 1250 (fixed)

The last parameter that is tuned is the destroy fraction. The value of 0.5 was also considered because this setting was applied during the verification and validation of the LNS algorithm, see subsection 6.2.1. When examining the *average total time*, a different destroy fraction performs best for each data set. However, similar to the previous experiments, the differences between the tested settings remain relatively small. The same pattern can be observed for the *average number of iterations*, where each data set again shows a different setting with the lowest iteration count.

The *average seconds per iteration* remain nearly identical for all destroy fraction values, indicating that the computational effort per iteration is largely unaffected by the selected setting. Furthermore, when considering the *standard deviation of the total time*, the value of 0.5 never performs best. Since larger destroy fractions remove a larger part of the current solution, this may unnecessarily disrupt good solution structures, especially for the larger and more complex data sets.

Overall, the destroy fraction of 0.3 provides the most balanced performance across all data sets. For Case 1B, it achieves the lowest runtime, while for Case 7 the runtime is nearly identical to the

best-performing setting. Although Case 3 performs slightly better with larger destroy fractions, the improvements remain limited. Therefore, a destroy fraction of 0.3 is selected as a balanced middle ground.

Table 5.6: Performance results for different data sets and destroy fraction values

destroy fraction	data set	avg total time	std total time	avg iterations	avg runtime (min)	avg seconds per iteration
0.2	Case 1B	2272	57.8	716	1.62	0.136
0.3	Case 1B	2280	31.8	579	1.29	0.133
0.4	Case 1B	2282	52.3	814	1.83	0.134
0.5	Case 1B	2261	51.7	622	1.41	0.135
		2274	**48.4**	**683**	**1.54**	**0.135**
0.2	Case 3	4507	93.1	732	6.03	0.494
0.3	Case 3	4484	91.8	856	6.99	0.490
0.4	Case 3	4489	66.0	794	6.56	0.495
0.5	Case 3	4494	148.1	692	5.72	0.496
		4494	**99.8**	**769**	**6.33**	**0.494**
0.2	Case 7	1294	7.9	1178	0.43	0.021
0.3	Case 7	1295	8.8	1144	0.41	0.021
0.4	Case 7	1291	8.1	1106	0.40	0.021
0.5	Case 7	1293	8.2	1113	0.40	0.021
		1293	**8.3**	**1135**	**0.41**	**0.022**

5.4. Summary

This chapter presented the experimental setup used for the LNS algorithm, including the preprocessing steps, data set preparation, vehicle configurations, and hardware setup. Different vehicle layouts were introduced to analyse the impact of stacking flexibility, while additional test cases are created to further evaluate container reuse under simplified conditions. Furthermore, the parameter tuning process explored the influence of Delta0, maximum rejections, maximum iterations, and destroy fraction across multiple representative data sets. Overall, the tuning results showed only relatively small differences between the tested parameter values in terms of solution quality and runtime. This indicates that the LNS algorithm performs robustly across a range of parameter settings, as no single parameter change caused large performance fluctuations. Consequently, the final parameter choices were mainly based on achieving a balanced trade-off between computational effort and solution quality rather than on substantial differences in objective values.

6

Results

This chapter presents the analysis of the results of the benchmark model and the LNS algorithm. First, the benchmark is discussed in section 6.1. The results of a few runs are shown, and one run is discussed in more detail. The LNS algorithm results are discussed in section 6.2. The chapter finishes with the sensitivity analysis in section 6.3.

6.1. Benchmark Results

6.1.1. Test Case

The test case consists of customers located in and around the city of Groningen. In total, five test cases are considered, starting with three customers and gradually increasing to seven in the final case. The customers were selected based on their geographical proximity, ensuring that they can be served by a single vehicle within one day. All locations in the data set are shown in Figure C.2 in Appendix C. Additionally, the customers were also selected based on their orders, ensuring that each order type is represented at least once. An overview of the order types and their corresponding container demand is provided in Table 6.1. A visualisation of the geographical distribution is presented in Figure 6.1. Although the figure may appear to show six customers, locations 6 and 24 are positioned almost at the same location.

The model applies a fixed handling time of 5 minutes and uses a 3×3 vehicle configuration with a maximum capacity of three full containers. Compared to the experimental setup of the LNS model in section 5.2, a smaller vehicle configuration is used because the benchmark scenarios contain a relatively limited container demand. This smaller configuration also helps to better identify potential limitations and operational bottlenecks within the benchmark model setup.

Table 6.1: Overview of benchmark test cases and incremental customer additions.

Number of Customers	Number of Orders
3 customers	13 (Change,1), 24 (Pickup,1), 24 (Delivery,1), 25 (Swap,1)
4 customers	Previous + 15 (Swap,2)
5 customers	Previous + 21 (Delivery,1)
6 customers	Previous + 26 (Swap,2)
7 customers	Previous + 6 (Swap,1)

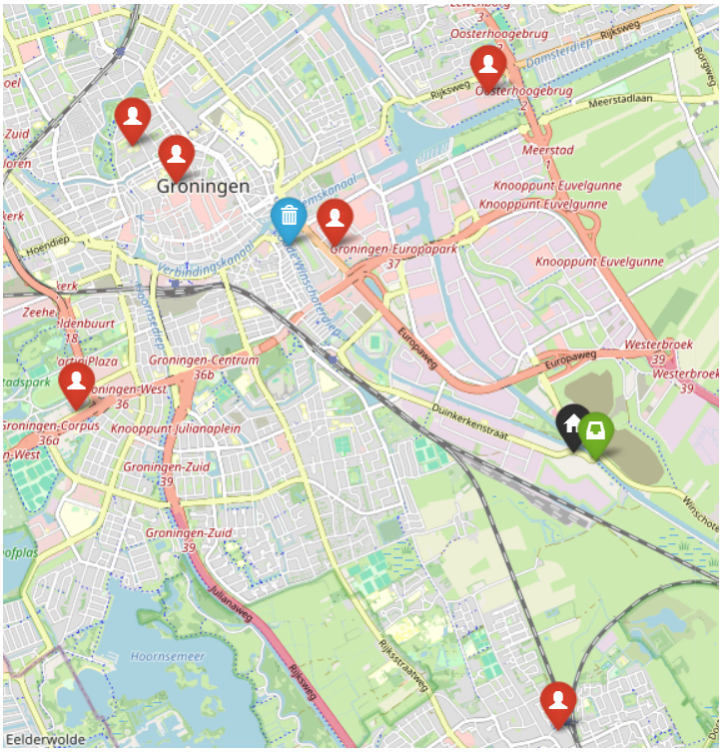


Figure 6.1: Selected locations benchmark in Groningen.
Colour coding: black – depot vehicles, green – storage locations, blue – disposal facility, red – customers.

6.1.2. Results

After running the five test cases, the corresponding KPI values are presented in Table 6.2. For the test cases with three to six customers, the results are averaged over three runs, while the case with seven customers is based on a single run due to longer computation times. In the benchmark model, the objective is to minimise driving time, as service time is assumed to be fixed. However, to later also compare with the LNS algorithm, the service time is added to the driving time to result in the total operational time. The container reuse rate is not considered here, as it falls outside the scope of the benchmark analysis, as discussed in subsection 3.1.1.

Table 6.2: Operational and computational performance for increasing numbers of customers.

KPI	Units	3 cust.	4 cust.	5 cust.	6 cust.	7 cust.
Total operational time	min	126.6	200.4	227.9	302.1	332.6
Vehicles used	#	1	1	1	1	1
Container reuse rate	%	0	0	0	0	0
Avg. vehicle load per arc	#	2.0	4.0	4.2	5.9	7.2
Avg. stacking utilisation	%	0.19	0.4	0.42	0.61	0.74
Avg. travel time per container	min	31.7	33.4	32.6	33.6	33.3
Orders per hour completed	min	1.9	1.5	1.6	1.4	1.4
Computational time	min	0.01	0.13	0.11	7.3	230.6
Optimality gap	%	0	4.4	4.9	5.0	5.0

From the results, it can be observed that the total operational time increases most significantly when a swap order with a demand of two is added. This is expected, as this order type consists of the largest number of components and therefore contributes the most to the total service time. Since orders are geographically close, adding an order with a higher demand still has a noticeable impact due to the increased service time associated with handling multiple containers.

Moreover, the average vehicle load per arc increases with the number of customers, together with the stacking utilisation. This indicates that the model continues to combine orders more efficiently as the model size grows. However, the number of orders completed per hour decreases slightly, which is also expected, as longer routes and additional driving time are required to serve more customers. The average travel time per container remains roughly constant. This suggests that the additional travel time scales proportionally with the number of containers handled. In other words, although routes become longer, the model is able to maintain a similar level of transport efficiency by combining more containers within the same trips, preventing a significant increase in travel time per individual container.

Lastly, the most significant increase can be observed in the computational time of the model. This is because each additional customer not only adds one extra decision, but also increases the number of possible routing, assignment, and sequencing combinations that the solver must evaluate. As a result, the solution space grows exponentially with instance size, making it increasingly difficult for the solver to prove optimality or find equally strong solutions in the same amount of time. In addition, the presence of multiple order components, capacity restrictions, and dummy nodes further increases the combinatorial complexity of the problem. Therefore, while the operational KPIs increase gradually, the computational effort required to solve the model increases disproportionately. This also supports the reasoning for building a metaheuristic algorithm.

Six Customers

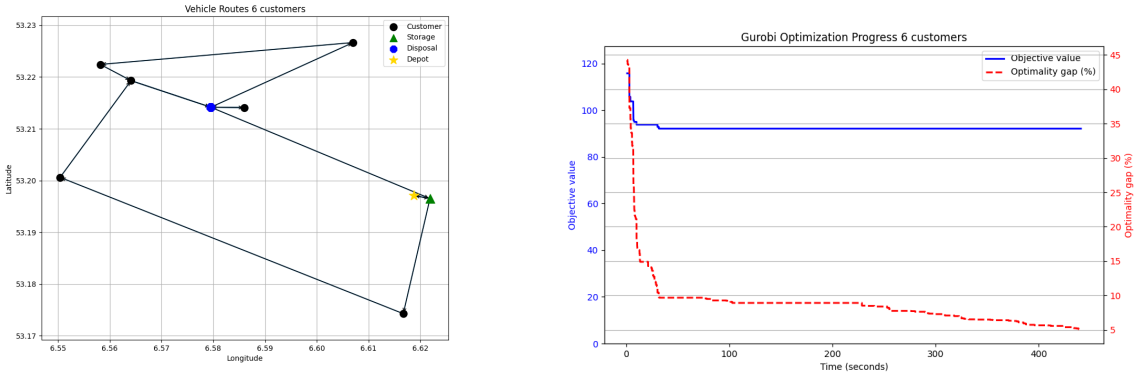
To provide a more detailed overview of the results, the scenario with six customers is further reviewed. The vehicle's time schedule is presented in Table 6.3, and the corresponding route is shown in Figure 6.2a along with the progress graph of Gurobi in Figure 6.2b. The total operational time can be split up as follows:

- **Total operational time:** 302.13 minutes
- **Total service time:** 210 minutes
- **Total driving time:** 92.13 minutes

In this scenario, the vehicle visits the storage location only at the beginning and the end of the trip, indicating that the pickup and dropoff of empty containers are combined efficiently. In contrast, the disposal facility is visited three times. Examining the vehicle load before each visit to the disposal facility shows that during the first two visits, the vehicle carries three full containers, the maximum capacity, and during the final visit, it carries the remaining two full containers. This suggests that the capacity of full containers may contribute to additional driving time, which is further examined in subsection 6.1.3.

Table 6.3: Schedule vehicle route in the benchmark with six customers (time is in minutes). D = depot, S = storage location, F = disposal facility

Arc	Dep. time	Driving time	Arr. time	Service time	Load change	Vehicle load
D -> S	0	0.45	0.45	35	7 empty ↑	7 empty
S -> 21	35.45	12.63	48.08	5	1 empty ↓	6 empty
21 -> 15	53.08	15.72	68.08	20	2 empty ↓, 2 full ↑	4 empty, 2 full
15 -> 13	88.8	10.32	99.12	5	1 full ↑	4 empty, 3 full
13 -> F	104.12	7.4	111.52	30	3 full ↓, 3 empty ↓	7 empty
F -> 26	141.52	7.2	148.72	20	2 empty ↓, 2 full ↑	5 empty, 2 full
26 -> 24	168.72	11.2	179.92	10	1 empty ↓, 1 full ↑	4 empty, 3 full
24 -> 13	189.92	2.9	192.82	5	1 empty ↓	3 empty, 3 full
13 -> F	197.82	7.4	205.22	20	2 full ↓, 2 empty ↑	5 empty, 1 full
F -> 25	225.22	3.52	228.74	10	1 empty ↓ 1 full ↑	4 empty, 2 full
25 -> F	238.74	4.52	243.26	20	2 full ↓, 2 empty ↑	6 empty
F -> S	263.26	8.43	271.69	30	6 empty ↓	-
S -> D	301.69	0.45	302.14	0	6 empty ↓	-



(a) Vehicle route of the benchmark with six customers.

(b) Process during optimisation.

Figure 6.2: Benchmark results for six customers.

6.1.3. Limitations

The main limitation of the model is the computational time. The solution time grows rapidly when a seventh customer is added. This increase is primarily caused by the structural complexity of the formulation, as the number of routing, sequencing, reuse, and stacking decisions grows substantially with the problem size. As a result, the benchmark MILP model becomes increasingly difficult to solve for larger cases, leading to a exponential increase in computational time.

There is still potential to further improve the coding structure to increase computational efficiency. However, due to time limitations, the primary focus was on ensuring that the model works correctly. Furthermore, since the model relies on a full exploration of the solution space, it is inherently computationally intensive, especially as the problem size increases.

Another limitation is that orders cannot be split. The model is formulated such that the full demand of an order must be collected or delivered in a single service activity. Consequently, pickup, change, or swap orders cannot exceed three containers, and delivery orders cannot exceed nine containers. This restriction follows from the requirement that all containers must be handled simultaneously, while the vehicle can carry a maximum of three full containers at a time. Therefore, if all full containers must be collected in one visit, the pickup quantity is limited to three. An extension of the model allowing for split pickups and deliveries would increase flexibility and enable larger demand quantities to be served. This is also in line with the limitation mentioned in Table 6.1.2, where it was shown in the route planning that the number of full containers creates a bottleneck in the number of orders that can be combined within a route.

6.2. LNS Results

The previous section discussed the operational and computational performance of the benchmark model. The following sections focus on the performance of the LNS algorithm and further evaluate the quality and reliability of the metaheuristic approach through verification and validation analyses, followed by a broader analysis of the generated results.

6.2.1. Verification and Validation

To verify and validate the results of the LNS algorithm, its results are compared with those of the benchmark model. The vehicle configurations differ slightly between the two models. In the benchmark model, no distinction is made regarding the placement of full containers, whereas in the LNS algorithm, full containers are restricted to the top row. Therefore, to provide the same capacity for empty containers, the benchmark uses a 3×3 configuration, while the LNS algorithm uses a 3×4 configuration. This ensures that both models can potentially carry 9 empty containers.

The following parameter settings are applied for the following runs. These differ slightly from the tuned settings in section 5.3, as the verification is performed on a smaller data set where a higher

destroy fraction of 50% is more suitable. This prevents the same order(s) from being removed repeatedly due to the limited number of orders in the data set.

- max iterations = 1000
- max rejections = 100
- delta0 = 0.15
- max time = 600 s
- destroy fraction = 50%

During the initial verification phase, reuse opportunities were absent due to inconsistencies in the exported swap order data, in which multiple container types were incorrectly assigned to a single swap order. After correcting the preprocessing procedure, the experiments were repeated using one container type for swap orders. Both results are retained because they provide insight into the performance of the LNS algorithm with and without reuse opportunities. The KPI results for the corrected data set are presented in Table 6.4. These represent the best results from five runs. Furthermore, the averages compared to the benchmark model can be found in Appendix E. This is followed by Table 6.5 comparing both cases with the benchmark. The gap between the two models is calculated by Equation 6.1.

$$\text{Gap} = \frac{\text{LNS} - \text{MILP}}{\text{MILP}} \times 100\% \tag{6.1}$$

Table 6.4: Operational and computational performance for increasing numbers of customers with reuse. The LNS is the best run out 5 runs. BM = Benchmark.

KPI	Units	3 cust.	3 cust.	4 cust.	4 cust.	5 cust.	5 cust.	6 cust.	6 cust.	7 cust.	7 cust.
		BM	LNS	BM	LNS	BM	LNS	BM	LNS	BM	LNS
Total operational time	min	126.6	126.6	200.4	194.2	227.9	226.6	302.1	282.4	332.6	316.8
Vehicles used	#	1	1	1	1	1	1	1	1	1	1
Container reuse rate	%	0	0	0	11.1	0	10.0	0	16.0	0	16.7
Avg. vehicle load per arc	#	2	1.8	4	2.3	4.2	3.3	5.9	3.4	7.2	2.3
Avg. stacking utilisation	%	0.2	0.1	0.4	0.2	0.4	0.3	0.6	0.3	0.7	0.2
Avg. travel time per container	min	31.7	31.7	33.4	32.4	32.6	32.4	33.6	31.4	33.3	31.7
Order per hour completed	#	1.9	1.9	1.5	1.5	1.6	1.6	1.4	1.5	1.4	1.5
Computational time	min	0.0	0.2	0.1	0.3	0.1	0.4	7.3	0.6	230.6	1.0
Optimality gap	%	0.0	-	4.4	-	4.9	-	5.0	-	5.0	-

Table 6.5: Comparison of Benchmark and LNS results regarding the total operational time.

	Number of customers	Benchmark	Best LNS	Gap	Avg. LNS	Gap
No reuse	3	126.6	126.6	+ 0.0%	126.6	+ 0.0%
	4	200.4	200.4	+ 0.0%	200.5	+ 0.0%
	5	227.9	232.4	+ 2.0%	234.6	+ 2.9%
	6	302.1	311.9	+ 3.3%	315.3	+ 4.4%
	7	332.6	351.8	+ 5.8%	351.8	+ 5.8%
	Average			+ 2.2%		+ 2.6%
With reuse	3	126.6	126.6	+ 0.0%	126.6	+ 0.0%
	4	200.4	194.2	- 3.1%	194.2	- 3.1%
	5	227.9	226.6	- 0.6%	229.2	+ 0.6%
	6	302.1	280.6	- 7.1%	282.2	- 6.6%
	7	332.6	316.8	- 4.8%	319.4	- 4.0%
	Average			- 3.1%		(absolute) 2.9%

When examining the verification results, it can be observed that the best result over the five runs shows an average deviation of less than 3% compared to the benchmark MILP model. When the

average over all five runs is considered, the deviation increases slightly in the case without reuse but remains below 3%. These results indicate that the LNS algorithm is capable of generating solutions of comparable quality within substantially shorter computational times, particularly for the larger scenarios where the benchmark model requires significantly more runtime.

When comparing the *no reuse* and *reuse* scenarios, an interesting pattern can be observed. In the 6 and 7 customer scenarios, approximately 16% container reuse is achieved. Due to the resulting reduction in operational time, the LNS algorithm even outperforms the benchmark MILP model in these cases. Furthermore, the average LNS gap is lower in the reuse scenarios, indicating that reuse has a clear influence on operational efficiency. This is also reflected by the generally lower or negative gap percentages compared to the *no reuse* scenarios, with the 3 customer scenario being the only exception.

In Figure 6.3 and Figure 6.4, the routes for the 7 customer scenarios are presented. For the LNS algorithm, more routes travel to and from the storage location, indicating that storage visits are combined less effectively than in the benchmark MILP model. This is also reflected in the KPIs, where the LNS algorithm achieves lower average vehicle load and stacking utilisation values. This behaviour may partly be explained by the storage combination improvement rules discussed in subsection 4.2.4.

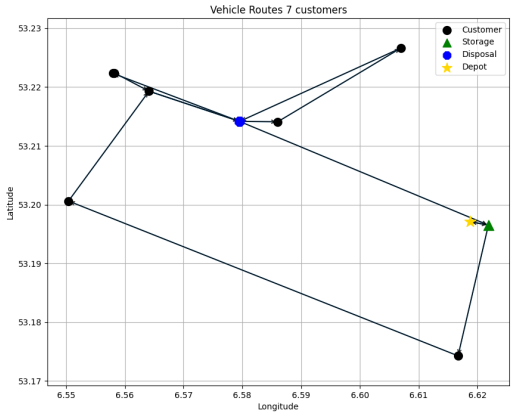


Figure 6.3: Route benchmark model

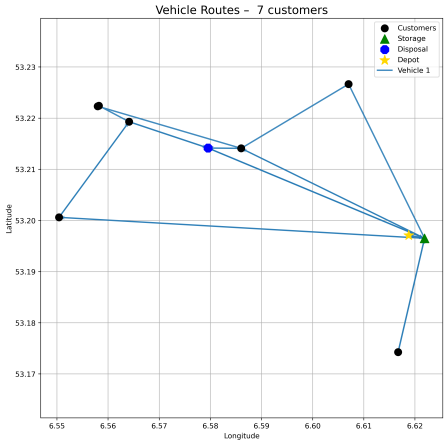


Figure 6.4: Route LNS algorithm with reuse

6.2.2. Computational Limits

To further assess the computational limits of the benchmark MILP model, additional experiments were conducted for scenarios containing 10, 11, and 12 customers. These experiments also provide insight into the scalability of the proposed LNS algorithm for larger problem scenarios. The same parameter settings as in the previous experiments were applied, and the best result over five LNS runs was considered. The scenarios were evaluated without reuse.

The results demonstrate an exponential increase in computational complexity for the benchmark MILP model as the number of customers increases. For the 10 and 11 customer scenarios, the benchmark model required 72.6 and 229.8 minutes, respectively, while still accepting optimality gaps of 30%. For the 12 customer scenario, a gap of 35% was accepted. The benchmark model reduced the optimality gap from 72.7% to 35% within a second, after which the run was finished. This explains why the computational time for the 12 customers was lower than for the 11 customers, despite the larger problem size. The computational limit of the benchmark MILP model became evident for the 13 customer scenario, for which no feasible solution was found within more than one hour of computational time.

In contrast, the LNS algorithm consistently produced feasible solutions for all tested scenarios within approximately two minutes of computational time. Furthermore, the deviation from the benchmark MILP solutions remained limited despite the substantially lower runtime. The best re-

sult across the five LNS runs showed an average deviation of 4.9% from the benchmark solutions, while the average across all runs was 6.4%. In various studies comparing heuristic approaches with MILP models, deviations of up to 10% are generally considered acceptable (Aksen et al., 2014; Özkır & Coban, 2025). These findings indicate that the LNS algorithm provides a substantially better trade-off between computational time and solution quality for larger scenarios, particularly as the benchmark MILP model approaches its computational limits. This demonstrates the practical relevance of the LNS algorithm for larger problem data sets where exact optimisation becomes computationally infeasible.

Table 6.6: Comparison of benchmark (BM) and LNS results for larger customer instances without reuse.

Customers	Benchmark total time	Best LNS total time	Gap	Avg LNS total time	Gap	BM Runtime	LNS Runtime	BM Gap
10	526.2	566.7	+7.7%	568.5	+8.0%	72.6	2.0	30%
11	555.4	591.1	+6.4%	597.6	+7.6%	229.8	2.2	30%
12	640.7	645.4	+0.7%	663.3	+3.5%	109.8	2.0	35%
Average			+4.9%		+6.4%			

6.2.3. Results Analysis

In this section, the results of the LNS runs are discussed. The complete results for all data sets are presented in Appendix E. First, the results are analysed per KPI. Afterwards, the focus shifts to the reuse test cases, including a more detailed analysis. In addition, several examples are provided to illustrate how the LNS algorithm iterates over time, along with specific cases from the generated routes.

Table 6.7: Average total operational time and vehicles used across configurations for the data sets.

	Operational time (min)			Vehicles used		
Data set	3×7	7×3	5×4	3×7	7×3	5×4
Case 1A	2594	2215	2240	6	5	5
Case 1B	2336	2310	2266	6	6	5
Case 2A	4071	3908	3915	9	9	9
Case 2B	3657	3556	3552	9	8	9
Case 3	4872	4541	4484	11	10	10
Case 4	3313	3241	3222	7	7	7
Case 5	2455	2399	2445	5	5	6
Case 6	6610	5944	6067	15	13	14
Case 7	1384	1263	1293	3	3	3
Case 8	1498	1380	1396	4	4	4
Case 3 test case	4811	4433	4450	11	10	10
Case 4 test case	3257	3222	3220	7	8	7

Table 6.7 presents the average *total operational time* for each data set for the three vehicle configurations. A clear and consistent pattern is visible across all data sets: the 3×7 configuration performs worst in every data set. Since the 3×7 and 7×3 configurations both provide a total capacity of 21 containers, this difference cannot be explained by total vehicle size alone, but rather by how the capacity is distributed between positions for full and empty containers.

For some of the larger and more operationally complex data sets, the vehicle configuration also affects the number of vehicles used. For example, in Case 3, the number of vehicles decreases from 11 under the 3×7 configuration to 10 under both the 7×3 and 5×4 configurations, while the operational time is reduced by more than 330 minutes. Case 6 shows the clearest effect, with each configuration requiring a different number of vehicles: 15 for 3×7, 13 for 7×3, and 14 for 5×4.

Case 6 also shows the largest reduction in operational time, decreasing from 6610 minutes under 3×7 to 5944 minutes under 7×3, corresponding to a reduction of approximately 650 minutes.

When comparing the 7×3 and 5×4 configurations, the differences are considerably smaller than those observed relative to the 3×7 configuration. In several cases, the difference is below 30 minutes, while for Case 2B the difference is only 4 minutes. This suggests that once sufficient capacity for transporting full containers is available, the exact balance between positions for full and empty containers becomes less decisive. Nevertheless, the 7×3 configuration achieves the lowest operational time in the majority of the data sets, indicating a slight but consistent advantage over 5×4.

Table 6.8: Average load per arc, stacking utilisation, and travel time per container comparison across configurations.

	Avg load per arc			**Avg stacking utilisation**			**Avg travel time per container**		
Data set	3×7	7×3	5×4	3×7	7×3	5×4	3×7	7×3	5×4
Case 1A	2.5	3.9	3.2	0.12	0.19	0.16	61.8	52.7	53.3
Case 1B	3.0	3.6	3.4	0.14	0.17	0.17	48.7	48.1	47.2
Case 2A	2.4	3.2	2.8	0.11	0.15	0.14	49.7	47.7	47.7
Case 2B	2.2	2.7	2.8	0.10	0.13	0.14	52.2	50.8	50.7
Case 3	2.1	2.5	2.5	0.10	0.12	0.12	64.1	59.8	59.0
Case 4	2.4	2.7	2.6	0.11	0.13	0.13	57.1	55.9	55.6
Case 5	2.5	3.1	2.8	0.12	0.15	0.14	57.1	55.8	56.9
Case 6	2.4	2.9	2.7	0.11	0.14	0.13	60.6	54.5	55.7
Case 7	2.4	4.0	3.0	0.11	0.19	0.15	44.7	40.8	41.7
Case 8	3.0	3.6	3.4	0.14	0.17	0.17	48.3	44.5	45.0
Case 3 test case	2.1	2.6	2.4	0.10	0.12	0.12	63.3	58.3	58.6
Case 4 test case	2.3	2.6	2.6	0.11	0.12	0.13	56.2	55.6	55.5

To better understand the differences in total operational time, Table 6.8 presents three additional KPIs related to vehicle utilisation: *average load per arc*, *average stacking utilisation*, and *average travel time per container*. These indicators help explain whether the observed improvements originate from carrying more containers per arc, using the available vehicle space more effectively, or reducing travel effort per transported container.

For the *average load per arc*, higher values indicate that more containers are transported during each travelled arc. The KPI generally follows the operational time results, with the 7×3 and 5×4 configurations achieving higher values than 3×7. For example, in the Case 7 data set, the value increases from 2.4 under 3×7 to 4.0 under 7×3 and 3.0 under 5×4, representing one of the largest improvements. In contrast, the differences for the Case 3 data set are less pronounced, with values of 2.1 for 3×7 and 2.5 for both alternative configurations.

Another important observation is that all average load values exceed 2.0. Compared to the literature discussed in Chapter 2, where most studies consider a container capacity of only one or two containers, these results indicate that the additional vehicle capacity is actively used in practice. If the average load had remained below 2.0, the larger vehicle configurations would not have been used extensively during the routing process.

The *average stacking utilisation* and *average travel time per container* show similar trends. For stacking utilisation, the differences are most visible in Case 1A, where the utilisation increases from 0.12 under 3×7 to 0.19 under 7×3, while 5×4 reaches 0.16. The travel time KPI shows smaller differences between configurations. For example, in the Case 4 data set, the value decreases from 57.1 minutes for 3×7 to 55.8 for 7×3 and 55.6 for 5×4. In Case 7, the KPI improves from 44.7 to 40.8 and 41.7, respectively.

These results indicate that the operational improvements are primarily achieved through transporting more containers simultaneously and using the available vehicle capacity more effectively, rather than through substantially shorter travel times per container.

Table 6.9: Orders per hour comparison across configurations for the data sets.

Data set ↓ \ **config** →	3×7 **avg**	7×3 **avg**	5×4 **avg**
Case 1A	0.69	0.81	0.80
Case 1B	1.03	1.04	1.06
Case 2A	1.15	1.20	1.20
Case 2B	1.15	1.18	1.18
Case 3	0.79	0.85	0.86
Case 4	1.03	1.06	1.06
Case 5	0.93	0.95	0.93
Case 6	0.83	0.92	0.90
Case 7	1.21	1.33	1.30
Case 8	1.16	1.26	1.25
Case 3 test case	0.81	0.87	0.86
Case 4 test case	1.05	1.06	1.06

When analysing the KPI *orders per hour* in Table 6.9, the results generally follow the operational-time trends discussed earlier. The 7×3 and 5×4 configurations achieve slightly higher service rates than 3×7, while remaining very close to one another. For example, in Case 2A the values increase from 1.15 under 3×7 to 1.20 under both alternative configurations. A comparable pattern can be observed in Case 6, where the KPI improves from 0.83 to 0.92 and 0.90, respectively. In Case 7, one of the largest relative improvements is visible, ranging from 1.21 for 3×7 to 1.33 for 7×3 and 1.30 for 5×4.

The differences between 7×3 and 5×4 remain limited across most data sets. In nearly all cases, the gap is only 0.00, 0.01, or 0.02 orders per hour, with Case 7 showing the largest difference of 0.03. This indicates that both configurations achieve a highly comparable service rate once sufficient routing flexibility is available.

Case 5 forms an exception, where the 3×7 and 5×4 configurations achieve the same value. This suggests that for some specific routing structures or demand patterns, a different balance between positions for full and empty containers can lead to less distinct performance differences between configurations.

Table 6.10: Algorithmic KPI comparison across configurations for the data sets.

	Iterations			**Runtime (min)**			**Seconds per iteration**		
Data set	3×7	7×3	5×4	3×7	7×3	5×4	3×7	7×3	5×4
Case 1A	883	859	790	0.77	0.54	0.48	0.05	0.04	0.04
Case 1B	681	606	673	1.57	1.33	1.53	0.14	0.13	0.14
Case 2A	1194	1112	1056	4.75	3.14	3.28	0.24	0.17	0.19
Case 2B	655	898	723	4.99	6.70	5.38	0.46	0.45	0.45
Case 3	725	782	799	6.53	6.37	6.57	0.54	0.49	0.49
Case 4	891	1003	995	5.07	5.21	5.27	0.34	0.31	0.32
Case 5	1070	909	1073	1.59	1.30	1.53	0.09	0.09	0.09
Case 6	863	816	812	12.30	8.26	8.78	0.85	0.61	0.65
Case 7	1125	1229	1143	0.57	0.55	0.41	0.03	0.03	0.02
Case 8	1099	1057	917	1.40	0.89	0.68	0.58	0.65	0.59
Case 3 test case	887	953	953	8.62	10.26	9.45	0.58	0.65	0.59
Case 4 test case	1085	839	947	6.10	4.36	5.01	0.34	0.31	0.32

The last KPIs that need to be discussed are the algorithmic ones: the *number of iterations* and *computational time*. Based on these, the *seconds per iteration* is also calculated in order to compare values from different data sets.

First of all, it can be noted that in the majority of the cases the 3×7 configuration has the highest

seconds per iteration time. However, in some cases, there is also no difference; this is the case for Case 5 and Case 7. It is also not surprising that the largest data set, Case 6, has the *longest seconds per iteration* time. In contrast, the data sets of Case 7, Case 8 and Case 1B, which are the smallest, require the least amount of time.

When considering the *number of iterations*, the Case 7 data set is, on average, the closest to the maximum value. However, there is no single data set that consistently reaches this limit. In contrast to the previous KPI results analysis, the number of iterations does not exhibit a clear pattern across the different vehicle configurations. No single configuration consistently results in either the lowest or the highest *number of iterations*. Each vehicle configuration achieves the lowest *number of iterations* an equal number of times, with all configurations recording the lowest value in four cases. Therefore, rather than focus solely on the vehicle configurations, it is more insightful to examine the characteristics of the individual data sets in order to understand how these influence the results and explain the observed differences.

Order Types

It is interesting to examine whether different combinations of order types affect the KPIs. In particular, the impact of swap and change orders is interesting. The main reason is that swap orders involve the largest number of routing components, namely five, whereas the other order types involve only two or three. Furthermore, change orders affect routing differently, as they are the only order type that requires a return to the same customer after the disposal facility visit. Unlike other order types, where the route can often continue to a shared storage location, the final step of a change order is tied to a fixed customer location. This makes the routing sequence more restrictive and may reduce opportunities for efficient routing. Therefore, the percentage differences presented in Table 6.11 are analysed. The vehicle configurations of 3×7 and 7×3 are compared.

When examining Table 6.11, it becomes clear that the data sets with the highest combined share of swap and change orders, namely Case 8, Case 7, Case 1A, and Case 6, also show the highest sensitivity to vehicle configuration. Case 1A provides the clearest example, with 93.3% of orders consisting of swap or change orders, resulting in a 14.6% reduction in operational time and a 56.0% increase in average load per arc. In contrast, Case 1B contains the lowest combined share of swap and change orders at 50.0% and shows only a 1.1% improvement in operational time.

When specifically considering change orders, Case 7 and Case 8 provide additional evidence that order composition may be more influential than total data set size alone. Although these data sets are among the smallest in terms of total operational time, see Table 6.7, they still show relatively large improvements between the configurations. Case 7 contains 92.9% change orders and shows an 8.7% reduction in operational time, together with a 66.7% increase in average load per arc. Similarly, Case 8 contains 86.2% change orders and shows a 7.9% improvement in operational time. This suggests that change orders increase the sensitivity to vehicle configuration. However, these findings are observational, as other data set characteristics, such as geography, container types, and facility availability, also vary simultaneously.

Table 6.11: Impact of swap and change order on configuration of 3×7 compared to 7×3.

Data set	Swap (%)	Change (%)	Swap + change (%)	Operational time improvement (%)	Load per arc improvement (%)	Vehicle reduction
Case 1A	56.7	36.7	93.3	14.6	56.0	1
Case 1B	47.5	2.5	50.0	1.1	20.0	0
Case 2A	28.2	52.6	80.8	4.0	33.3	0
Case 2B	50.0	10.0	60.0	2.8	22.7	1
Case 3	59.4	12.5	71.9	6.8	19.0	1
Case 4	50.9	10.5	61.4	2.2	12.5	0
Case 5	44.7	39.5	84.2	2.3	24.0	0
Case 6	40.7	44.0	84.6	10.1	20.8	2
Case 7	3.6	92.9	96.4	8.7	66.7	0
Case 8	13.8	86.2	100.0	7.9	20.0	0

Explore Behaviour

Before examining the routes and specific examples in more detail, it is useful to analyse the search behaviour throughout the iterations of the runs by considering how the best, current, and candidate solutions evolve over time. To do so, runs from the Case 2B data set are explored below. Case 2B showed the largest variation in the number of iterations across all data sets.

The selected runs are taken from the 7×3 configuration. Figure 6.5 shows the run where the maximum number of iterations is reached, while Figure 6.6 shows the run with the lowest number of iterations reached. Run 4 achieved a final operational time of 3415 minutes, whereas run 10 resulted in 3580 minutes.

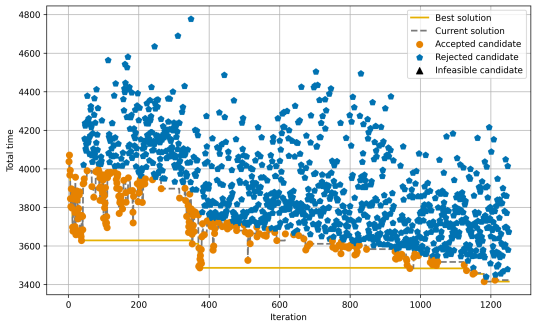


Figure 6.5: Improvement graph of Case 2B configuration 7×3 for run 4.

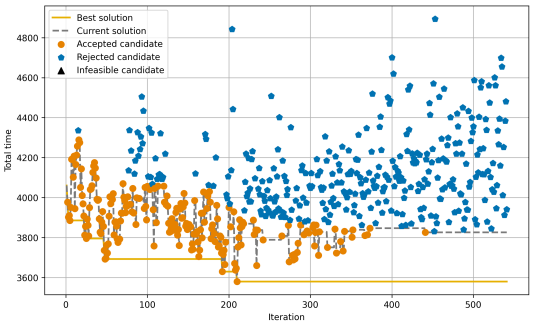


Figure 6.6: Improvement graph of Case 2B configuration 7×3 for run 10.

After the initial solution construction, both runs show a rapid improvement phase in which the LNS algorithm quickly identifies solutions with substantially lower operational times. This indicates that the destroy and repair operators are effective in exploring promising neighbourhoods during the early stages of the search. However, the search behaviour differs between the two runs after this initial improvement phase.

Run 4, in Figure 6.5, exhibits a stable search trajectory. The current solution remains closer to the incumbent throughout the run, and the accepted candidate solutions continue to improve operational time over a larger number of iterations. Furthermore, the accepted candidate solutions are concentrated around the current and best solution values, indicating a more controlled balance between diversification and intensification. This allows the search to continue refining promising route structures instead of drifting towards poorer regions of the solution space.

In contrast, run 10, shown in Figure 6.6, the search reaches a good solution relatively early in the process. However, after 250 iterations, the current solution increasingly diverges from the incumbent solution. Although many candidate solutions continue to be generated, most are rejected and the current solution remains at substantially higher operational times compared to the best solution found. This suggests that the search process becomes more exploratory and struggles to intensify around the promising region where the incumbent solution was identified. As a result, the search is unable to consistently return to similar high-quality solutions later in the run.

The differences between both runs can largely be explained by the stochastic nature of the LNS framework. Since destroy operators, repair operators, and operator selection are partly random, small differences in the early search trajectory can lead the algorithm towards different regions of the solution space. As the search progresses away from promising regions, the algorithm may struggle to return to comparable high-quality solutions later in the run. This behaviour is particularly visible in run 10, where the incumbent solution is found early, but subsequent iterations mainly explore worse neighbourhoods without recovering similar solution structures. The results, therefore, suggest that the current implementation places a relatively strong emphasis on diversification, which helps avoid premature convergence but may also reduce the ability of the search to consistently intensify around promising regions during some runs.

Route examples

The focus now shifts to several route examples from the best performing runs. The Case 1B and Case 2A data sets were selected because they differ strongly in the flexibility of feasible disposal facilities and storage locations. As shown in Table 5.2, the Case 1B data set exhibits limited variation in feasible disposal facilities, while both storage locations can, on average, be used for each order. In contrast, the Case 2A data set contains substantially more variation, where on average only 1 out of 16 disposal facilities is feasible for a given order, while approximately 6.5 out of 10 storage locations are feasible. These cases, therefore, provide insight into how differences in operational flexibility influence the resulting routes. In addition, different vehicle configurations are compared.

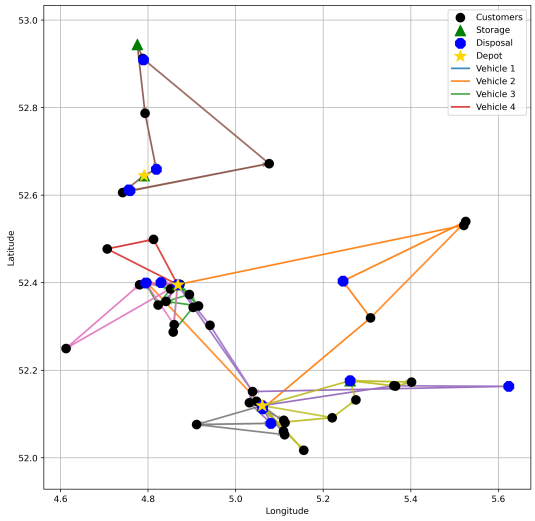


Figure 6.7: Route of Case 2A configuration 3×7 for best run 4.

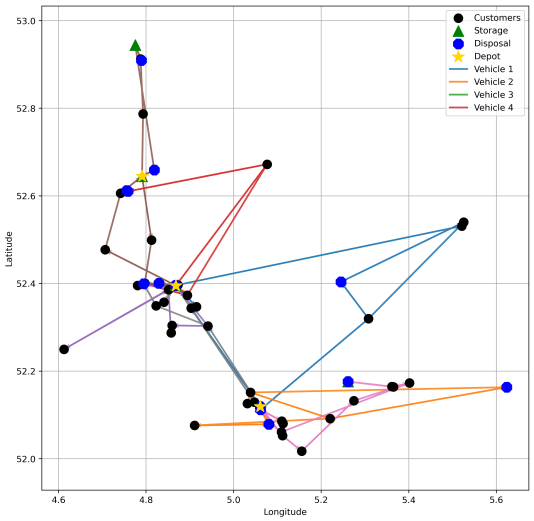


Figure 6.8: Route of Case 2A configuration 7×3 for best run 8.

To further examine the capacity bottleneck identified in the earlier sections, Table 6.12 and Table 6.13 are compared for the configurations 3×7 and 7×3. The schedules show substantially higher full container loads in the 7×3 configuration, allowing multiple orders at the same customer location to be merged within a single route. Although not all orders are executed by the same vehicle in both solutions, the results clearly illustrate how increased full container capacity enables more consolidated routing structures.

Table 6.12: Route overview of Case 2A 3×7 run 4 for vehicle 1.

From	To	Driving time (min)	Order	Customer Cluster	Order type	Δ empty	Δ full	Load empty	Load full
D 0	28	0.00	9	1	Change	0	+3	0	3
28	F 12	10.82	9	-	Change	+3	-3	3	0
F 12	28	12.18	9	1	Change	-3	0	0	0
28	28	0.00	7	1	Change	0	+2	0	2
28	F 11	0.18	7	-	Change	+2	-2	2	0
F 11	28	0.18	7	1	Change	-2	0	0	0
28	28	0.00	6	1	Change	0	+2	0	2
28	F 12	10.82	6	-	Change	+2	-2	2	0
F 12	28	12.18	6	1	Change	-2	0	0	0
28	28	0.00	8	1	Change	0	+3	0	3
28	F 11	0.18	8	-	Change	+3	-3	3	0
F 11	28	0.18	8	1	Change	-3	0	0	0
28	28	0.00	4	1	Change	0	+3	0	3
28	F 12	10.82	4	-	Change	+3	-3	3	0
F 12	28	12.18	4	1	Change	-3	0	0	0
28	28	0.00	5	1	Change	0	+3	0	3
28	F 12	10.82	5	-	Change	+3	-3	3	0
F 12	28	12.18	5	1	Change	-3	0	0	0
28	D 0	0.00	-	-	-	0	0	0	0
Total route time: 508.73 minutes									

Table 6.13: Route overview of Case 2A 7×3 run 8 for vehicle 3.

From	To	Driving time (min)	Order	Customer Cluster	Order type	Δ empty	Δ full	Load empty	Load full
D 0	28	0.00	6	1	Change	0	+5	0	5
28	F 12	10.82	6	-	Change	+5	-5	5	0
F 12	28	12.18	6	1	Change	-5	0	0	0
28	28	0.00	7	1	Pickup	0	+3	0	3
28	F 20	7.25	7	-	Pickup	+3	-3	3	0
F 20	S 11	7.47	7	-	Pickup	-3	0	0	0
S 11	28	0.18	5	1	Change	0	+7	0	7
28	F 11	0.18	5	-	Change	+7	-7	7	0
F 11	28	0.18	5	1	Change	-7	0	0	0
28	D 0	0.00	-	-	-	0	0	0	0
Total route time: 428.27 minutes									

Another important aspect is the flexibility in feasible disposal facilities and storage locations, which directly affects the routing structure. In the Case 2A 7×3 route, shown in Table 6.13, three different disposal facilities are visited. In contrast, the Case 1B route under the 5×4 configuration, shown in Table 6.14, contains a strong overlap between disposal facilities and storage locations. Several locations fulfil both activities, allowing two of the three required pickup activities to be completed during a single stop.

The schedule further shows a clear operational structure in which empty containers for delivery orders are collected at the start of the route, while the latter part mainly consists of pickup activities involving the emptying and storage of containers. Since change orders do not benefit from this type of overlap, the example highlights the operational efficiency advantages associated with delivery and pickup orders.

Table 6.14: Route overview of Case 1B 5×4 run 10 for vehicle 1. Reuse is applied for the marked components.

From	To	Driving time (min)	Order	Customer Cluster	Order type	Δ empty	Δ full	Load empty	Load full
D 0	S 0	0.00	48	-	Swap_Delivery	+1	0	1	0
S 0	S 0	0.00	44	-	Swap_Delivery	+1	0	2	0
S 0	S 0	0.00	41	-	Swap_Delivery	+1	0	3	0
S 0	37	37.42	49	2	Swap_Pickup	0	+1	3	1
37	37	0.00	48	2	Swap_Delivery	-1	0	2	1
37	35	6.62	45	2	Swap_Pickup	0	+1	2	2
35	35	0.00	44	2	Swap_Delivery	-1	0	1	2
35	33	11.93	42	2	Swap_Pickup	0	+1	1	3
33	33	0.00	41	2	Swap_Delivery	-1	0	0	3
33	F 0	43.80	45	-	Swap_Pickup *	+1	-1	1	2
F 0	S 0	0.00	45	-	Swap_Pickup	-1	0	0	2
S 0	F 0	0.00	49	-	Swap_Pickup	+1	-1	1	1
F 0	S 0	0.00	49	-	Swap_Pickup	-1	0	0	1
S 0	S 0	0.00	22	-	Swap_Delivery *	+1	0	1	1
S 0	S 0	0.00	25	-	Delivery	+1	0	2	1
S 0	22	31.73	25	3	Delivery	-1	0	1	1
22	20	22.62	23	3	Swap_Pickup	0	+1	1	2
20	20	0.00	22	3	Swap_Delivery	-1	0	0	2
20	32	16.87	40	3	Pickup	0	+1	0	3
32	19	16.02	21	3	Pickup	0	+1	0	4
19	F 0	13.05	21	-	Pickup	+1	-1	1	3
F 0	S 0	0.00	21	-	Pickup	-1	0	0	3
S 0	F 0	0.00	40	-	Pickup	+1	-1	1	2
F 0	S 0	0.00	40	-	Pickup	-1	0	0	2
S 0	F 0	0.00	23	-	Swap_Pickup	+1	-1	1	1
F 0	S 0	0.00	23	-	Swap_Pickup	-1	0	0	1
S 0	F 0	0.00	42	-	Swap_Pickup	+1	-1	1	0
F 0	S 0	0.00	42	-	Swap_Pickup	-1	0	0	0
S 0	D 0	0.00	-	-	-	0	0	0	0
Total route time: 380.05 minutes									

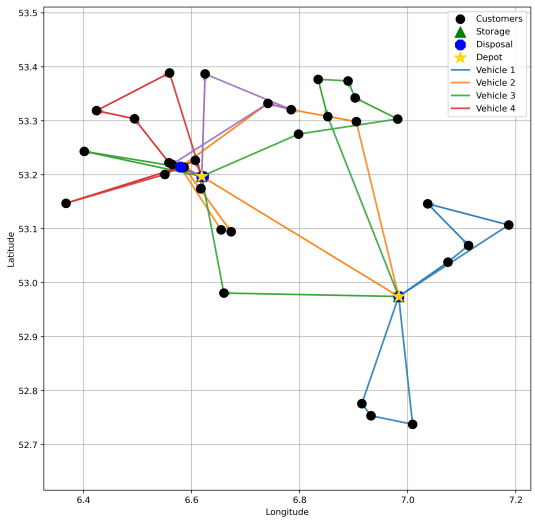


Figure 6.9: Route of Case 1B configuration 5×4 for best run 10.

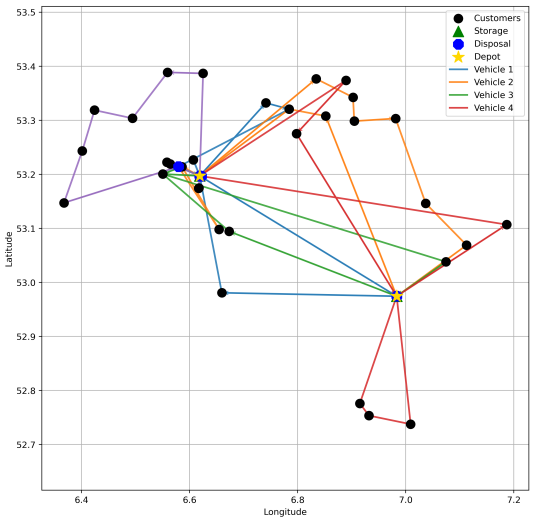


Figure 6.10: Route of Case 1B configuration 7×3 for best run 3.

6.2.4. Stacking Analysis

Stacking capacity is analysed separately in this subsection to better understand how higher vehicle capacities influence routing behaviour and operational performance.

Table 6.15: Maximum stacking values found in the best runs across all data sets for the different vehicle configurations.

Data set	3×7	7×3	5×4
Case 1A	6	10	11
Case 1B	14	9	11
Case 2A	8	9	10
Case 2B	7	8	12
Case 3	7	9	7
Case 4	6	8	7
Case 5	8	7	7
Case 6	15	10	15
Case 7	6	11	7
Case 8	8	10	10

High load values occur across all data sets, although the frequency and magnitude differ substantially between cases, as shown in Table 6.15. Data sets such as Case 6 and Case 1B reach the highest stacking values, with maximum values up to 15 and 14 containers, respectively, indicating that some routes make very intensive use of the available stacking capacity. In contrast, data sets such as Case 4 and Case 5 show lower maximum values between 6 and 8 containers, suggesting that the operational structure of these data sets requires less extensive stacking. A noticeable pattern is that the highest stacking values mainly occur in data sets containing larger shares of swap and change orders, such as Case 6, Case 1A, and Case 7. These order types require more routing components and create stronger incentives to transport multiple full containers simultaneously. At the same time, the maximum values also show that the highest capacities are not utilised consistently across all routes. Instead, the results suggest that stacking is particularly valuable for specific locally concentrated route segments where multiple compatible orders can be combined efficiently within the same vehicle route.

To explore these effects in more detail, the Case 8 and Case 7 data sets are selected. Case 8 is selected because it achieves one of the highest *average load per arc* values, while Case 7 exhibits some of the largest differences between the vehicle configurations. Since Case 1B was already analysed extensively in previous subsections, Case 8 and Case 7 provide additional insight into how stacking capacity influences route design and operational performance. For each configuration, the run with an *average load per arc* value closest to the average result is selected to ensure that the analysed routes are representative rather than exceptional cases.

Table 6.16: Run closest to the average load per arc compared with average values for Case 7 and Case 8.

Data set	**KPI** $\downarrow$ \ **config** $\rightarrow$	3×7 **repr.**	3×7 **avg**	7×3 **repr.**	7×3 **avg**	5×4 **repr.**	5×4 **avg**
Case 7	run number	run 9	-	run 2	-	run 2	-
	vehicles used	3	3	3	3	3	3
	total operational time	1386	1384	1270	1263	1280	1293
	avg load per arc	2.43	2.39	3.96	4.01	3.08	2.97
	avg stacking utilisation	0.12	0.11	0.19	0.19	0.15	0.15
Case 8	run number	run 3	-	run 2	-	run 5	-
	vehicles used	4	4	3	4	3	4
	total operational time	1503	1498	1389	1380	1416	1396
	avg load per arc	2.95	2.99	3.61	3.56	3.63	3.43
	avg stacking utilisation	0.14	0.14	0.17	0.17	0.18	0.17

The route examples in Figure 6.11 to Figure 6.13 help interpret the KPI differences reported in Table 6.16. For the Case 7 data set, all three configurations use the same number of vehicles, indicating that the operational improvements from higher stacking capacities are not achieved through fleet reduction. Instead, the primary effect is visible in the operational time, which decreases by approximately 100 minutes when comparing 3×7 with the other configurations.

This difference is also reflected in the route structures shown in the figures. In the 3×7 configuration, the routes are distributed more evenly in geographical size and route duration. In contrast, the 7×3 and 5×4 configurations contain two noticeably longer routes, while the remaining vehicle operates within a smaller geographical region. The figures, therefore, suggest that higher stacking capacities allow orders to be grouped more efficiently within the same routes.

The comparison between the 7×3 and 5×4 configurations suggests that once sufficient full container capacity is available, further differences in how that capacity is distributed across positions and stacking rows have a limited effect on operational performance. Although both configurations outperform 3×7, the differences between them remain small across most data sets, as shown in Table 6.16.

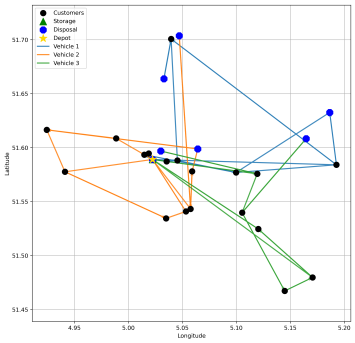


Figure 6.11: Route of Case 7 configuration 3×7 for run 9.

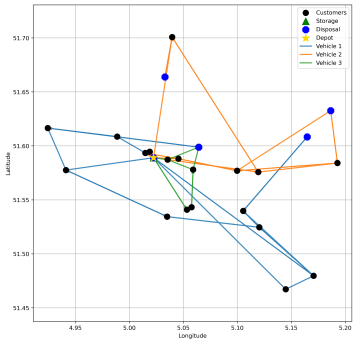


Figure 6.12: Route of Case 7 configuration 7×3 for run 2.

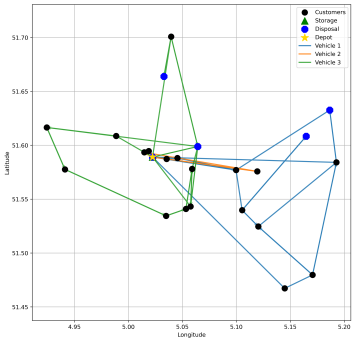


Figure 6.13: Route of Case 7 configuration 5×4 for run 2.

For the Case 8 data set, the configuration differences are also reflected in the route structures shown in Figure 6.14 to Figure 6.16. In contrast to Case 7, the 3×7 configuration requires an additional vehicle, while the fourth vehicle covers only a relatively small geographical area. This suggests that the lower full container capacity becomes restrictive for a limited number of orders that cannot be efficiently integrated into the other routes. In the 7×3 and 5×4 configurations, the routes become more geographically concentrated, with three vehicles covering the regions more efficiently. Similar to Case 7, the visual differences between these configurations remain limited, which is consistent with the relatively small operational differences reported in Table 6.16.

The Case 8 routes further illustrate that route design is not solely determined by vehicle capacity. In Figure 6.16, the first vehicle (blue) still travels towards a disposal facility in the region mainly covered by the third vehicle (green), corresponding to the cities of Maastricht and Heerlen in Limburg, shown in Figure C.10 in Appendix C. Although using one of the facilities already visited in Heerlen would appear more logical, this routing behaviour is directly determined by the feasible facilities defined in the input data.

This example shows that facility restrictions can strongly influence the resulting route structures and partially offset the operational benefits obtained from higher stacking capacities, particularly for disposal facilities where orders may have only one feasible destination.

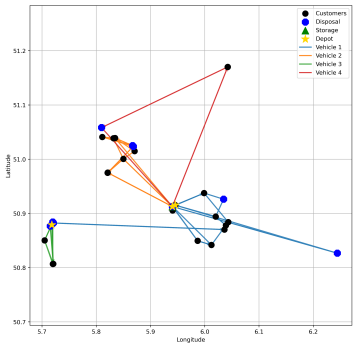


Figure 6.14: Route of Case 8 configuration 3×7 for run 3.

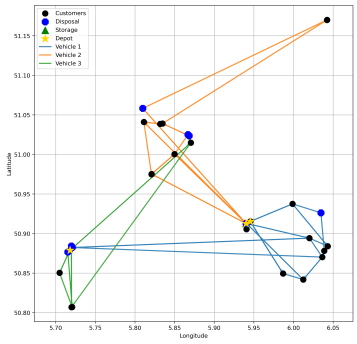


Figure 6.15: Route of Case 8 configuration 7×3 for run 2.

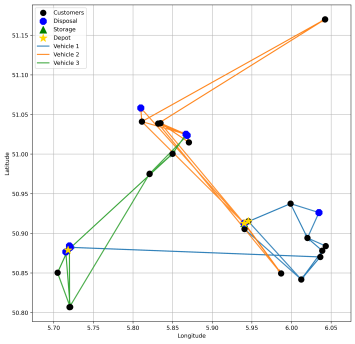


Figure 6.16: Route of Case 8 configuration 5×4 for run 5.

Vehicle Layouts

To better understand how stacking capacity is utilised within the routes, vehicle layouts are analysed at moments of high loading levels. These layouts provide additional insight into how higher capacity configurations support the route structures observed previously.

For Case 7, the previous route analysis already showed that the 7×3 configuration achieves substantially lower operational times while using the same number of vehicles as the 3×7 configuration. The layouts below help explain how this operational improvement is achieved in practice. All three vehicles in the 7×3 configuration temporarily transport six or seven full containers simultaneously, indicating that the additional full container capacity is actively used within the routes.

Case 7

$$\begin{bmatrix} F & F & F & F & F & F & - \\ - & - & - & - & - & - & - \\ - & - & - & - & - & - & - \end{bmatrix}$$
Vehicle 1

$$\begin{bmatrix} F & F & F & F & F & F & F \\ E & - & - & - & - & - & - \\ - & - & - & - & - & - & - \end{bmatrix}$$
Vehicle 2

$$\begin{bmatrix} F & F & F & F & F & F & F \\ - & - & - & - & - & - & - \\ - & - & - & - & - & - & - \end{bmatrix}$$
Vehicle 3

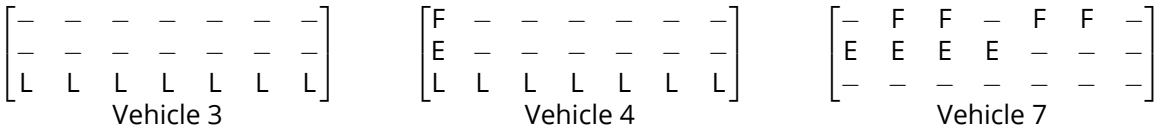
The route overview of vehicle 3 in Table 6.17 further illustrates how this loading pattern develops throughout the route. During the first part of the route, multiple change orders are consecutively collected before unloading occurs. As a result, the vehicle gradually increases its full container load from one to seven containers between nodes 38, 30, and 22. Only after reaching this peak load does unloading start at disposal facility 14, indicating that multiple change orders are intentionally clustered before unloading occurs.

Table 6.17: Route overview of Case 7 run 2 for vehicle 3.

From	To	Driving time (min)	Order	Customer Cluster	Order type	Δ empty	Δ full	Load empty	Load full
D 0	38	15.25	18	1	Change	0	+1	0	1
38	30	1.40	9	1	Change	0	+5	0	6
30	22	10.68	1	1	Change	0	+1	0	7
22	F 14	6.72	1	-	Change	+1	-1	1	6
F 14	F 14	0.00	9	-	Change	+5	-5	6	1
F 14	37	4.70	17	1	Change	0	+1	6	2
37	F 0	2.45	17	-	Change	+1	-1	7	1
F 0	F 0	0.00	18	-	Change	+1	-1	8	0
F 0	38	15.25	18	1	Change	-1	0	7	0
38	30	1.40	9	1	Change	-5	0	2	0
30	22	10.68	1	1	Change	-1	0	1	0
22	37	5.87	17	1	Change	-1	0	0	0
37	D 0	2.45	-	-	-	0	0	0	0
Total route time: 284.85 minutes									

To further examine whether similar loading patterns occur in other cases, additional layouts from Case 2A and Case 1A are analysed. Case 2A is selected because it contains the highest proportion of large containers, while Case 1A contains the highest number of containers per order before merging.

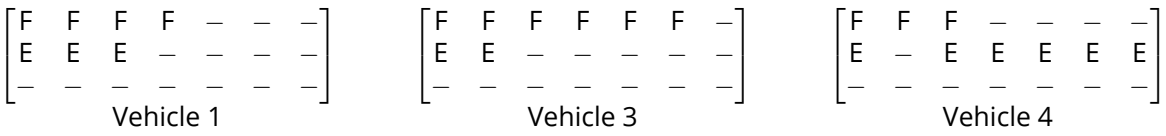
Case 2A



For Case 2A, the layouts show that relatively high loading levels can still be achieved even when large containers occupy substantial vehicle space. Vehicle 4 reaches the highest observed load among the analysed examples, with a total of nine simultaneously transported containers. In contrast, vehicle 7 mainly transports combinations of full and empty containers while leaving an entire stacking row unused.

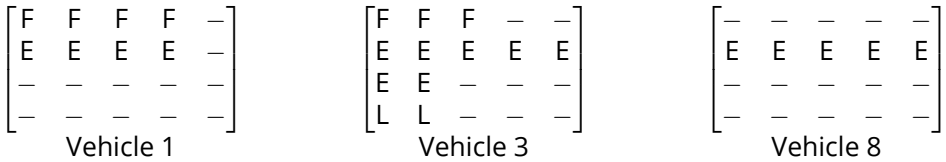
A similar pattern can be observed for Case 1A. Although certain vehicles temporarily carry relatively high combinations of full and empty containers, the layouts show clear variation between vehicles in how intensively the available stacking capacity is used.

Case 1A



Lastly, Case 2B provides an additional perspective, as it is the only data set where the 5×4 configuration achieves a higher *average load per arc* than the 7×3 configuration. The layouts below show that vehicle 3 temporarily achieves a very high loading level by simultaneously transporting full, empty, and large containers. In contrast, vehicle 8 carries at most five empty containers during its route despite the larger available capacity.

Case 2B



Across all analysed layouts, high stacking levels occur repeatedly across all data sets, but mainly during temporary peak loading moments within specific route segments rather than continuously throughout the routes. The results, therefore, indicate that high stacking utilisation is primarily driven by locally concentrated orders and temporary route clustering rather than by all vehicles operating close to their maximum capacity throughout the entire route duration.

At the same time, the layouts structurally use capacities above one or two containers, demonstrating that such assumptions are restrictive for chain-lift skip operations. However, the third stacking row often remains partially or fully unused. This suggests that, for the analysed cases, two stacking rows are sufficient for most operational situations, while additional rows mainly provide flexibility during temporary peak loads. Consequently, the results indicate that vehicle design should focus more on supporting combinations of full and empty containers within the same route than on maximising the total number of stacking rows.

6.2.5. Reuse Analysis

Lastly, the focus shifts to the additional test case variants of Case 3 and Case 4, which involve only a limited number of change orders, for which reuse is not possible. In these test cases, the distinction between container types is removed to create more opportunities for container reuse

between orders. The complete results for all configurations are presented in Appendix E, while Table 6.18 summarises the differences between the original data sets and their corresponding test case versions.

Table 6.18: Average KPI comparison between original data sets and reuse test cases.

Data set	**KPI** $\downarrow$ \ **config** $\rightarrow$	3×7	7×3	5×4
Case 3 vs test case	reuse rate (%)	+7.3	+6.5	+7.3
	operational time (min)	-61	-108	-34
	vehicles used	0	0	0
	load per arc	-0.03	+0.02	-0.14
	orders per hour	+0.01	+0.02	+0.00
Case 4 vs test case	reuse rate (%)	+10.8	+11.1	+8.9
	operational time (min)	-106	-100	-97
	vehicles used	0	0	0
	load per arc	-0.08	+0.03	+0.10
	orders per hour	+0.04	+0.03	+0.03

A first clear observation from Table 6.18 is that the reuse rate increases in every configuration. For Case 3, the average increase is 7.0%, while for Case 4 it is 10.3%. This results in a reuse percentage of around 12% for Case 3 and 17% for Case 4, full results can be found in Appendix E. This confirms that the model is able to generate additional reuse opportunities when the distinction between container types is removed. However, the increase is not large enough to reduce the average number of vehicles, which remains unchanged in all cases. The total operational time does decrease consistently, with reductions ranging between 34 and 108 minutes depending on the configuration.

The route example in Table 6.19 illustrates why additional reuse does not necessarily eliminate complete visits from the route. Most pickup components of delivery orders are already strongly clustered at the beginning of the route, while the dropoff activities are grouped later in the route. As a result, many storage locations are still visited for multiple route components even when reuse is applied. In these situations, reuse mainly reduces handling activities rather than driving time, since removing one container movement does not automatically remove an entire stop from the route.

To implement additional reuse within such a clustered route structure, dropoff activities would need to be repositioned earlier in the route before compatible delivery activities occur. However, because these activities are already efficiently combined with other route components visiting the same location, therefore, the resulting operational savings remain limited.

Table 6.19: Route overview of Case 3 test case 3×7 run 1 for vehicle 7.

From	To	Component type	Driving time (min)	Order	Customer Cluster	Order type	Δ empty	Δ full	Load empty	Load full
D 0	S 1	Delivery_P	0.60	81	-	Swap_Delivery	+1	0	1	0
S 1	S 1	Delivery_P	0.00	3	-	Swap_Delivery	+4	0	5	0
S 1	7	Delivery_D	60.28	3	5	Swap_Delivery	-4	0	1	0
7	S 7	Delivery_P	0.00	63	-	Delivery	+1	0	2	0
S 7	S 7	Delivery_P	0.00	83	-	Swap_Delivery	+1	0	3	0
S 7	64	Delivery_D	88.52	83	6	Swap_Delivery	-1	0	2	0
64	63	Delivery_D	38.65	81	6	Swap_Delivery	-1	0	1	0
63	63	Empty_P	0.00	82	6	Swap_Pickup	0	+1	1	1
63	53	Delivery_D	5.82	63	6	Delivery	-1	0	0	1
53	66	Empty_P	9.87	87	6	Pickup	0	+1	0	2
66	F 7	Empty_D	62.97	87	-	Pickup	+1	-1	1	1
F 7	S 7	Dropoff	0.00	87	-	Pickup	-1	0	0	1
S 7	F 7	Empty_D	0.00	82	-	Swap_Pickup	+1	-1	1	0
F 7	S 7	Dropoff	0.00	82	-	Swap_Pickup	-1	0	0	0
S 7	7	Empty_P	0.00	5	5	Swap_Pickup	0	+2	0	2
7	31	Empty_P	40.87	26	10	Swap_Pickup	0	+1	0	3
31	F 1	Empty_D	20.08	5	-	Swap_Pickup	+2	-2	2	1
F 1	F 1	Empty_D	0.00	26	-	Swap_Pickup	+1	-1	3	0
F 1	S 1	Dropoff	0.00	5	-	Swap_Pickup	-2	0	1	0
S 1	S 1	Dropoff	0.00	26	-	Swap_Pickup	-1	0	0	0
S 1	D 0	-	0.60	-	-	-	0	0	0	0
Total route time: 500.25 minutes										

The second observation concerns the *average load per arc*, shown in Table 6.18. The value sometimes increases and, in other cases, decreases, indicating that reuse affects route utilisation differently depending on the route structure. A decrease in *load per arc* can occur because a container no longer needs to be transported over a long distance before being delivered to the next customer. Through reuse, the container can instead remain closer to the next required location, reducing the transported load throughout the route. In contrast, an increase in *load per arc* can occur when reuse enables additional orders to be combined within the same route, resulting in more containers being transported simultaneously. The KPI therefore reflects two opposing effects of reuse: reducing unnecessary container movements while also enabling denser route combinations.

Furthermore, the route in Table 6.20 shows a route where reuse is integrated much more extensively throughout the route structure. The marked Dropoff* and Delivery_P* components and yellow rows illustrate how empty containers are reused between multiple orders within the same route.

Finally, the *orders per hour* KPI increases slightly in all cases, which is consistent with the lower operational times. Although the improvements remain relatively limited, the results consistently indicate that reuse contributes positively to routing efficiency.

Table 6.20: Route overview of Case 3 test case 7×3 run 3 for vehicle 9. Reuse is applied to the marked components.

From	To	Component type	Driving time (min)	Order	Customer Cluster	Order type	Δ empty	Δ full	Load empty	Load full
D 1	S 1	Delivery_P	0.00	56	-	Swap_Delivery	+1	0	1	0
S 1	S 1	Delivery_P	0.00	28	-	Swap_Delivery	+1	0	2	0
S 1	S 1	Delivery_P	0.00	12	-	Delivery	+1	0	3	0
S 1	S 1	Delivery_P	0.00	67	-	Swap_Delivery	+1	0	4	0
S 1	56	Delivery_D	2.65	67	2	Swap_Delivery	-1	0	3	0
56	25	Empty_P	1.68	11	2	Swap_Pickup	0	+1	3	1
25	F 1	Empty_D	4.28	11	-	Swap_Pickup	+1	-1	4	0
F 1	S 1	Dropoff*	0.00	11	-	Swap_Pickup	0	0	4	0
S 1	S 1	Delivery_P	0.00	71	-	Swap_Delivery	+1	0	5	0
S 1	58	Delivery_D	22.53	71	4	Swap_Delivery	-1	0	4	0
58	26	Delivery_D	10.80	12	4	Delivery	-1	0	3	0
26	S 1	Delivery_P*	0.00	27	-	Delivery	0	0	3	0
S 1	33	Delivery_D	19.72	27	4	Delivery	-1	0	2	0
33	50	Delivery_D	2.33	56	4	Swap_Delivery	-1	0	1	0
50	34	Delivery_D	38.28	28	1	Swap_Delivery	-1	0	0	0
34	68	Empty_P	14.28	88	1	Pickup	0	+1	0	1
68	32	Empty_P	11.38	26	1	Swap_Pickup	0	+1	0	2
32	F 1	Empty_D	48.23	26	-	Swap_Pickup	+1	-1	1	1
F 1	F 1	Empty_D	0.00	88	-	Pickup	+1	-1	2	0
F 1	S 1	Dropoff*	0.00	88	-	Pickup	0	0	2	0
S 1	S 1	Delivery_P*	0.00	41	-	Swap_Delivery	0	0	2	0
S 1	S 1	Dropoff*	0.00	26	-	Swap_Pickup	0	0	2	0
S 1	S 1	Delivery_P*	0.00	75	-	Swap_Delivery	0	0	2	0
S 1	61	Delivery_D	11.63	75	2	Swap_Delivery	-1	0	1	0
61	61	Empty_P	0.00	76	2	Swap_Pickup	0	+1	1	1
61	42	Delivery_D	6.98	41	2	Swap_Delivery	-1	0	0	1
42	F 1	Empty_D	8.20	76	-	Swap_Pickup	+1	-1	1	0
F 1	S 1	Dropoff*	0.00	76	-	Swap_Pickup	0	0	1	0
S 1	S 1	Delivery_P*	0.00	46	-	Swap_Delivery	0	0	1	0
S 1	44	Delivery_D	28.67	46	6	Swap_Delivery	-1	0	0	0
44	55	Empty_P	8.00	66	6	Swap_Pickup	0	+2	0	2
55	F 1	Empty_D	30.05	66	-	Swap_Pickup	+2	-2	2	0
F 1	S 1	Dropoff	0.00	66	-	Swap_Pickup	-2	0	0	0
S 1	D 1	-	0.00	-	-	0	0	0	0	
Total route time: 455.72 minutes										

6.3. Sensitivity Analysis

This section examines how LNS performance changes when service times are varied. The analysis focuses on the effect of changing service times, which directly influences the balance between driving time and handling time within the objective value.

Two data sets are selected for this analysis: Case 5 and Case 8. These data sets contain different combinations of order types. For both data sets, the original service times are compared with a lower scenario ($0.75\times$) and a higher scenario ($1.25\times$), both averaged over 10 runs. The complete KPI tables are included in Appendix E, while Table 6.21 summarises the average total operational time, number of vehicles used, orders per hour, and stacking utilisation.

Table 6.21: Sensitivity analysis summary for different service time levels (average results across runs).

Data set	Scenario	Best config	Total time	Vehicles	Orders/hour	Stack util.
Case 5	$0.75\times$	5×4	2099	5	1.09	0.16
Case 5	Base	7×3	2399	5	0.95	0.15
Case 5	$1.25\times$	5×4	2719	6	0.84	0.13
Case 8	$0.75\times$	7×3	1155	3	1.51	0.22
Case 8	Base	7×3	1380	4	1.26	0.17
Case 8	$1.25\times$	7×3	1598	4	1.09	0.15

The first clear observation from Table 6.21 is that shorter service times improve all main KPIs. For

both data sets, the *total operational time* decreases substantially, while productivity in terms of *orders per hour* increases. In addition, fewer vehicles are required for Case 8, with a decrease from 4 to 3 vehicles in the $0.75\times$ scenario. A similar pattern is visible for stacking utilisation, where lower service times lead to higher loading levels and more efficient combinations of orders within the same routes.

When service times increase, the opposite pattern emerges. The *total operational time* rises in all cases, while *orders per hour* decreases. For Case 5, the average vehicle usage increases to 6 vehicles in the $1.25\times$ scenario. In addition, routes can accommodate fewer orders per day and therefore offer less operational flexibility. As a result, combining additional containers becomes less beneficial and vehicle utilisation decreases. This indicates that handling activities can become a bottleneck when service times increase.

Another interesting observation is that the preferred vehicle configuration changes slightly. In the base case, the 7×3 configuration performs best for both data sets. However, for Case 5, the 5×4 configuration performs slightly better in both alternative scenarios, although the differences remain small.

Overall, the sensitivity analysis shows that the LNS algorithm responds consistently to changing service time assumptions. Lower service times lead to shorter and more productive routes, whereas higher service times increase route duration and vehicle usage while reducing stacking utilisation and operational flexibility.

6.4. Summary

This chapter analysed the benchmark model and the LNS algorithm across a range of data sets and vehicle configurations. The benchmark results demonstrated that the MILP formulation produces operationally realistic routes and captures the main routing characteristics of chain-lift skip container transportation. However, computational time grows exponentially with the problem size, confirming that the benchmark model is not suitable for larger operational scenarios and supporting the need for the LNS metaheuristic. The verification and validation results showed that the LNS algorithm generates solutions within an average deviation of less than 3% from the benchmark while requiring only a fraction of the computational time. In the reuse scenarios, the LNS even outperformed the benchmark model by successfully exploiting direct container reuse opportunities, resulting in lower total operational times for the larger validation cases.

Across all data sets, the 7×3 and 5×4 configurations consistently outperformed the 3×7 configuration. The results indicate that the primary operational bottleneck lies in the number of positions available for full containers rather than in the total stacking height. Increasing full container capacity improves the ability to combine multiple orders within the same routes, raises the average load per arc, and reduces operational times. In several larger data sets, this also leads to a reduction in the number of vehicles required. The vehicle layout analysis further showed that capacities above two containers are actively used in practice, while additional stacking rows above two rows are mainly used during peak loading moments rather than consistently throughout the routes. The order type analysis additionally showed that data sets with higher shares of swap and change orders exhibit greater sensitivity to vehicle configuration.

The reuse analysis confirmed that removing container type distinctions increases reuse opportunities and consistently reduces operational times, although the magnitude of the improvement depends on the existing route structure. In strongly clustered routes, reuse primarily reduces handling activities rather than complete route stops, resulting in limited savings in driving time. The sensitivity analysis further showed that the LNS algorithm responds consistently to changes in service time assumptions. Lower service times lead to shorter routes, higher stacking utilisation, and in some cases fewer vehicles, while higher service times increase route duration and reduce operational flexibility. In both analyses, the relative ranking of the vehicle configurations remained stable, with 3×7 performing weakest and 7×3 and 5×4 achieving comparable and consistently stronger results.

7

Conclusion and Discussion

The final chapter begins with answers to the sub-questions in order to answer the main research question in section 7.1. Subsequently, in section 7.2, the limitations of the research are discussed, followed by recommendations for future research and practical implications for AMCS.

This thesis addressed the routing of chain-lift skip containers, which can be stacked on vehicles and reused between customers without first returning to a storage location. These two operational characteristics have received limited attention in the existing literature and are often modelled with little detail. The thesis addressed this gap by incorporating stacking feasibility constraints and direct container reuse into a vehicle routing model. A benchmark MILP model was first developed as a reference solution, followed by a metaheuristic LNS algorithm capable of handling the full operational complexity at scale.

7.1. Conclusion

SQ 1. What is the current state of skip container routing in the literature?

The current state of skip container routing literature shows that the problem is well recognised as a complex variant of the VRP within waste collection. Existing studies focus on the transportation of skip containers between depots, customers, storage locations, and disposal facilities. The model can also be categorised as a PDP due to the precedence relations between empty and full skips. Over time, the literature has increased realism by incorporating aspects such as multiple locations, limited container availability, time windows, multiple container types, and the coordination of empty and loaded container flows.

However, the literature remains primarily focused on the larger RoRo container, where vehicle capacity is usually limited to one container or two when trailers are considered. As a result, the operational characteristics of the stackable chain-lift skip container are still underrepresented. In particular, stacking is often simplified to a numerical capacity limit, while detailed stacking rules, such as compatibility between container sizes and reuse possibilities, remain largely unexplored. Similar to the broader VRP literature, increasing model complexity has also shifted solution approaches away from exact methods towards neighbourhood-based metaheuristics, such as TS, VNS, and LNS.

SQ 2. Which key performance indicators are suitable for evaluating and comparing routing models?

Suitable KPIs for evaluating and comparing routing models should cover operational performance, resource use, and computational effort. Based on the literature and the objectives of this research, the most relevant KPIs are total operational time as the primary performance indicator, the number of vehicles used to evaluate vehicle usage, and orders per hour as a productivity measure. In addition, the average vehicle load per arc and the average stacking utilisation are suitable to measure how effectively vehicle capacity is used and how much stacking is applied. The travel time per container provides insight on transport efficiency from a container perspective, while the reuse rate measures the frequency with which direct reuse opportunities are exploited. Finally, computational time and number of iterations are useful algorithmic indicators to compare the practical applicability. Together, these KPIs provide a balanced overview of both operational and methodological performance.

SQ 3. How can a mixed-integer linear programming model be formulated to represent standard skip container operations?

An MILP model can be formulated to represent standard skip container operations by defining sets, parameters, decision variables, an objective function, and constraints. In the benchmark model, vehicles operate between a depot, customers, a storage location, and a disposal facility. Order requests are split up into components that are arranged in different sequences depending on the order type. Binary variables determine vehicle routing and vehicle assignment decisions, while continuous variables track arrival times and vehicle loads. The objective minimises total driving time as a benchmark measure of efficiency. Constraints ensure route continuity, correct order of execution of components, daily operating time limits, and capacity restrictions for both the total number of containers and the quantity of full containers.

SQ 4. How can a metaheuristic be conceptually designed to handle container reuse and stacking feasibility?

A metaheuristic can be conceptually designed to handle container reuse and stacking feasibility through an LNS framework that combines an initial solution, improvement rules, and an iterative destroy-repair cycle. The process starts by clustering geographically close customers and generating a feasible initial solution with a regret-2 insertion heuristic, where all insertions are checked for capacity, precedence relations, and stacking feasibility. Additional improvement rules are subsequently applied to combine compatible order components and match empty container flows between orders to look for reuse possibilities. The main loop iteratively removes and reinserts order requests using destroy and repair operators that balance diversification and intensification. Candidate solutions are evaluated with a RRT acceptance criterion, which allows controlled non-improving moves to escape local optima, while infeasible solutions are rejected. The search ends when the maximum number of iterations, the runtime limit, or a stagnation limit is reached.

SQ 5. How does the metaheuristic model perform relative to the mixed-integer linear programming model in terms of the defined key performance indicators?

The metaheuristic LNS model performs well relative to the benchmark MILP model across all defined KPIs while requiring substantially lower computational times. In the verification scenarios without reuse, both the best LNS runs and the average across all runs achieved average deviations below 3% from the benchmark solutions, indicating consistently strong solution quality. In the scenarios with reuse, the deviations were even smaller, and the LNS outperformed the benchmark in several cases by successfully exploiting direct reuse opportunities between customers, achieving reuse rates of approximately 16%. The computational experiments further demonstrated that the MILP formulation scales poorly with increasing problem size. Solution time grew from under one minute for three customers to approximately 230 minutes for seven customers, with the optimality gap reaching its maximum allowed value of 5% for the larger cases. In contrast, the LNS generated near optimal solutions for all verification scenarios within approximately one minute and remained computationally manageable for larger scenarios of 10, 11, and 12 customers, consistently producing solutions within approximately two minutes with an average deviation of 4.9% from the benchmark, whereas the benchmark required several hours while still yielding optimality gaps of 30-35%.

MQ: How can vehicle routing for skip container waste collection be optimised under container reuse and stacking feasibility constraints?

Vehicle routing for skip container waste collection can be optimised under container reuse and stacking feasibility constraints by using a solution approach that explicitly incorporates stacking compatibility, precedence relations, and direct reuse decisions during route construction and improvement. This thesis demonstrated that an LNS metaheuristic is well suited for this purpose, as it can efficiently generate high-quality routing solutions while accounting for operational constraints related to container capacities, stacking feasibility, disposal visits, and reuse opportunities. The benchmark MILP formulation provided a valid reference model for representing standard skip container operations, while the LNS framework enabled the optimisation of larger and more realistic operational scenarios within practical computational times.

The results showed that both stacking feasibility and container reuse influence operational performance. Increasing the number of available full container positions improved vehicle utilisation, reduced disposal facility visits, and lowered total operational times by up to 14.6% across the tested data sets. In addition, direct container reuse reduced unnecessary empty transport movements and storage visits, resulting in operational time reductions of 34-108 for the test cases. The experiments further demonstrated that average vehicle loads above 2.0 containers per travelled arc are actively used in practice, indicating that the common one or two container assumptions found in the literature are too restrictive for chain-lift skip operations.

Container reuse also improved routing efficiency, with increased compatibility leading to 7-11% higher reuse rates and 34-108 minutes less operational time per data set. However, the results also showed that the effectiveness of reuse depends strongly on the existing route structure, as highly consolidated routes mainly benefit through reduced handling time rather than the elimination of complete route stops. These findings demonstrate that realistic skip container routing can be substantially improved by explicitly incorporating stacking feasibility and container reuse into the optimisation process, while also showing that vehicle configurations with higher full container capacity provide the largest operational benefits.

7.2. Discussion

7.2.1. Limitations

This study also has several limitations. The first limitation relates to the quality of the data. During the analysis, some routes included relatively long driving times to disposal facilities despite the apparent availability of closer alternatives. This raises the question whether these facilities were truly mandatory for those orders or whether there are inaccuracies within the operational data. Improving the quality and consistency of the underlying data could therefore further improve routing performance.

A second limitation concerns the operational scope of the model. All analysed orders were treated as stackable chain-lift skip containers. In practice may transport chain-lift skips, closed skips, and RoRo skips simultaneously, each requiring different vehicle capacities and handling constraints. The current model does not support this mixed container setting, which limits direct applicability in operations where multiple container types are served by the same vehicles.

Additional limitations relate to the scope of the models themselves. Several practical aspects were intentionally excluded to focus on reuse and stacking feasibility. For example, time windows, driver rest periods, waiting times at facilities, and limited container availability were not included. Furthermore, storage locations were assumed to have unlimited capacity and container availability. Although these assumptions simplify the modelling process, they may limit direct applicability in a more constrained operational setting.

Finally, the LNS results are inherently stochastic and therefore subject to variation between runs. Although ten runs were performed, this may still not fully capture the variability in solution quality caused by different random seeds. The Case 2B explore behaviour analysis showed that, for several data sets, the gap between the average and best performing run was substantial. This indicates that the algorithm can occasionally converge to weaker local solutions and suggests that,

in practice, multiple runs should be executed with the best solution selected for implementation in operational planning.

7.2.2. Recommendations Future Research

Future research could extend the model by incorporating the limitations discussed in subsection 7.2.1, such as time windows, driver rest periods, limited container availability, and capacity restrictions at storage locations. Including these operational constraints would further improve the realism of the model and provide additional insight into the trade-offs between operational complexity and computational performance.

In addition, the variety of order types could be expanded further. Wy et al. (2013) discusses several additional order types that could be incorporated, such as *relocate*, where a customer container is moved between locations, and *full from depot*, where a full container is first stored at the depot before later being emptied at a disposal facility.

In addition, the current LNS framework could be expanded into an ALNS, where destroy and repair operators are selected adaptively based on recent performance rather than randomly. This would allow a large set of operators to be introduced, as the current model only uses two destroy and two repair operators. This would allow a larger and more diverse operator set to be incorporated, potentially improving the balance between diversification and intensification during the search process.

Further research could additionally focus on improving the current improvement rules. The results indicated that nearby disposal facility visits were not always combined efficiently, suggesting that additional savings opportunities remain possible. One potential improvement would be to introduce dedicated clustering strategies for disposal facilities and storage locations, allowing nearby visits to be evaluated jointly.

Moreover, the choice of RRT as an acceptance criterion was based on findings from the literature, but alternative acceptance methods, such as SA, may also be suitable for this problem and could potentially improve solution quality. Finally, experiments under stochastic travel times, dynamic order arrivals, or uncertain demand conditions would provide further insight into the robustness of the proposed approach in real operational environments.

7.2.3. Recommendations for AMCS

The results indicate that supporting stacking across two rows already provides substantial operational improvements, while extending towards larger stacking heights offers limited additional benefits in most practical routing situations. The experiments consistently showed that the primary operational bottleneck is the number of positions available for full containers rather than the overall stacking height, with the 7×3 and 5×4 configurations outperforming configurations with fewer full container positions across most data sets. This suggests that future vehicle configurations should prioritise sufficient capacity for transporting full containers while also accounting for operational feasibility and legal weight restrictions. In addition, further operational data analysis and monitoring of routing performance could help identify more optimal vehicle configurations, since the configurations analysed in this research were primarily based on values from the data sets together whit complementary ones rather than a complete data driven vehicle configuration study.

The experiments further showed that container reuse can improve routing efficiency by reducing unnecessary handling activities, storage visits, and empty transport movements. However, the results also demonstrated that part of the observed performance improvement originates from the clustering of geographically close and operationally similar activities within the same routes, rather than from reuse alone. In several scenarios, reuse eliminated handling operations without fully removing route stops. This suggests that improving the modelling of stacking feasibility, container compatibility, and route consolidation strategies may provide greater operational value than focusing exclusively on maximising reuse rates. At the same time, the relatively short computation times of approximately two minutes per planning scenario create opportunities for dynamic re-optimisation during the day, allowing routes to be updated when new orders arrive

or operational disruptions occur.

Finally, the experiments highlighted the importance of reliable operational data. Several scenarios contained unexpectedly long travel times towards disposal facilities, indicating potential inaccuracies or inconsistencies in the input data. For practical implementation, AMCS would benefit from incorporating automatic validation checks before optimisation, for example by verifying travel times, geographic consistency, and operational feasibility of orders. In addition, because the LNS algorithm is stochastic, solution quality varies between runs. Since computation times remained relatively low, practical implementations should execute multiple runs automatically and select the best-performing solution before dispatching routes. Integrating the optimisation framework directly into operational planning systems would further enable routing decisions, container availability, and vehicle capacities to be updated continuously using operational data, thereby improving both planning reliability and responsiveness.

References

Aksen, D., Kaya, O., Sibel Salman, F., & Tüncel, Ö. (2014). An adaptive large neighborhood search algorithm for a selective and periodic inventory routing problem. *European Journal of Operational Research*, *239*(2), 413–426. https://doi.org/10.1016/J.EJOR.2014.05.043

Archetti, C., Coelho, L. C., Speranza, M. G., & Vansteenwegen, P. (2025). Beyond fifty years of vehicle routing: Insights into the history and the future. https://doi.org/10.1016/j.ejor.2025.06.014

Archetti, C., Mansini, R., & Speranza, M. G. (2005). Complexity and reducibility of the skip delivery problem. *Transportation Science*, *39*(2), 182–187. https://doi.org/10.1287/trsc.1030.0084

Archetti, C., & Speranza, M. G. (2004). Vehicle routing in the 1-skip collection problem. *Journal of the Operational Research Society*, *55*(7), 717–727. https://doi.org/10.1057/palgrave.jors.2601743

Aringhieri, R., Bruglieri, M., Malucelli, F., & Nonato, M. (2018). A special vehicle routing problem arising in the optimization of waste disposal: A real case. *Transportation Science*, *52*(2), 277–279. https://doi.org/10.1287/trsc.2016.0731

Baldacci, R., Bodin, L., & Mingozzi, A. (2006). The multiple disposal facilities and multiple inventory locations rollon–rolloff vehicle routing problem. *Computers & Operations Research*, *33*(9), 2667–2702. https://doi.org/10.1016/J.COR.2005.02.023

Belenguer, J. M., Cubillos, M., & Wøhlk, S. (2024). A branch-and-cut algorithm for a skip pick-up and delivery problem. *Computers & Operations Research*, *168*, 106705. https://doi.org/10.1016/J.COR.2024.106705

Benjamin, A. M., & Beasley, J. E. (2010). Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, *37*(12), 2270–2280. https://doi.org/10.1016/J.COR.2010.03.019

Bodin, L., Mingozzi, A., Baldacci, R., & Ball, M. (2000). Rollon-Rolloff vehicle routing problem. *Transportation Science*, *34*(3), 271–288. https://doi.org/10.1287/trsc.34.3.271.12301

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82–117. https://doi.org/10.1016/J.INS.2013.02.041

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016, September). The vehicle routing problem: State of the art classification and review. https://doi.org/10.1016/j.cie.2015.12.007

Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science*, *39*(1), 104–118. https://doi.org/10.1287/trsc.1030.0056

Clarke, G., & Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, *12*(4), 568–581. https://doi.org/10.1287/opre.12.4.568

Cortés, C. E., Matamala, M., & Contardo, C. (2010). The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, *200*(3), 711–724. https://doi.org/10.1016/J.EJOR.2009.01.022

Dantzig, G., & Ramser, J. (1959). The Truck Dispatching Problem. *Management Science*, *6*(1). https://doi.org/https://doi.org/10.1287/mnsc.6.1.80

Dekker, R., Voogd, P., & Van Asperen, E. (2006). Advanced methods for container stacking. *OR Spectrum*, *28*(4), 563–586. https://doi.org/10.1007/s00291-006-0038-3

Dominguez, O., Guimarans, D., Juan, A. A., & de la Nuez, I. (2016). A Biased-Randomised Large Neighbourhood Search for the two-dimensional Vehicle Routing Problem with Backhauls.

European Journal of Operational Research, *255*(2), 442–462. https://doi.org/10.1016/J.EJOR.2016.05.002

Dueck, G. (1993). New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computational Physics*, *104*(1), 86–92. https://doi.org/10.1006/jcph.1993.1010

Dumas, Y., Desrosiers, J., & Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, *54*(1), 7–22. https://doi.org/10.1016/0377-2217(91)90319-Q

Elshaer, R., & Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers and Industrial Engineering*, *140*. https://doi.org/10.1016/j.cie.2019.106242

Eurostat. (2024, September). *Waste statistics*. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Waste_statistics

Fazi, S., Choudhary, S. K., & Dong, J. X. (2023). The multi-trip container drayage problem with synchronization for efficient empty containers re-usage. *European Journal of Operational Research*, *310*(1), 343–359. https://doi.org/10.1016/J.EJOR.2023.02.041

François, V., Arda, Y., Crama, Y., & Laporte, G. (2016). Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, *255*(2), 422–441. https://doi.org/10.1016/j.ejor.2016.04.065

Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2010). Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research*, *201*(3), 751–759. https://doi.org/10.1016/J.EJOR.2009.03.046

Golden, B. L., Assad, A. A., Wasil, E. A., Dantzig, G. B., & Ramser, J. H. (2002). *Routing Vehicles in the Real World: Applications in the Solid Waste, Beverage, Food, Dairy, and Newspaper Industries 10.1 Introduction* (tech. rep.). http://www.siam.org/journals/ojsa.php

Gunawardhana, J. A., Perera, H. N., & Thibbotuwawa, A. (2021). Rule-based dynamic container stacking to optimize yard operations at port terminals. *Maritime Transport Research*, *2*. https://doi.org/10.1016/j.martra.2021.100034

Hauge, K., Larsen, J., Lusby, R. M., & Krapper, E. (2014). A hybrid column generation approach for an industrial waste collection routing problem. *Computers & Industrial Engineering*, *71*(1), 10–20. https://doi.org/10.1016/J.CIE.2014.02.005

Henke, T., Speranza, M. G., & Wäscher, G. (2015). The multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research*, *246*(3), 730–743. https://doi.org/10.1016/J.EJOR.2015.05.020

Hess, C., Dragomir, A. G., Doerner, K. F., & Vigo, D. (2024). Waste collection routing: a survey on problems and methods. *Central European Journal of Operations Research*, *32*(2), 399–434. https://doi.org/10.1007/s10100-023-00892-y

Junqueira, L., Morabito, R., & Sato Yamashita, D. (2012). Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers and Operations Research*, *39*(1), 74–85. https://doi.org/10.1016/j.cor.2010.07.017

Kim, B. I., Kim, S., & Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, *33*(12), 3624–3642. https://doi.org/10.1016/J.COR.2005.02.045

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, *43*(4), 408–416. https://doi.org/10.1287/trsc.1090.0301

Lee, C. Y., & Song, D. P. (2017). Ocean container transport in global supply chains: Overview and research opportunities. *Transportation Research Part B: Methodological*, *95*, 442–474. https://doi.org/10.1016/J.TRB.2016.05.001

Lei, H., Laporte, G., & Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, *38*(12), 1775–1783. https://doi.org/10.1016/j.cor.2011.02.007

Li, H., Jian, X., Chang, X., & Lu, Y. (2018). The generalized rollon-rolloff vehicle routing problem and savings-based algorithm. *Transportation Research Part B: Methodological*, *113*, 1–23. https://doi.org/10.1016/J.TRB.2018.05.005

Liu, F., Lu, C., Gui, L., Zhang, Q., Tong, X., & Yuan, M. (2023). Heuristics for Vehicle Routing Problem: A Survey and Recent Advances. http://arxiv.org/abs/2303.04147

Meng, Q., & Wang, S. (2011). Liner shipping service network design with empty container repositioning. *Transportation Research Part E: Logistics and Transportation Review*, *47*(5), 695–708. https://doi.org/10.1016/J.TRE.2011.02.004

M&K. (2026). *Construction and demolition waste*. Retrieved May 21, 2026, from https://the-mkgroup.com/construction-and-demolition-waste/

Naccache, S., Côté, J. F., & Coelho, L. C. (2018). The multi-pickup and delivery problem with time windows. *European Journal of Operational Research*, *269*(1), 353–362. https://doi.org/10.1016/J.EJOR.2018.01.035

Osaba, E., Villar-Rodriguez, E., Del Ser, J., Nebro, A. J., Molina, D., LaTorre, A., Suganthan, P. N., Coello Coello, C. A., & Herrera, F. (2021). A Tutorial On the design, experimentation and application of metaheuristic algorithms to real-World optimization problems. *Swarm and Evolutionary Computation*, *64*(4), 100888. https://doi.org/10.1016/j.swevo.2021.100888

Özkır, R., & Coban, E. (2025). ALNS algorithm for load handling and AGV routing with trolleys. *Computers & Industrial Engineering*, *208*, 111334. https://doi.org/10.1016/J.CIE.2025.111334

Paquay, C., Schyns, M., & Limbourg, S. (2016). A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *International Transactions in Operational Research*, *23*(1-2), 187–213. https://doi.org/10.1111/itor.12111

Pisinger, D., & Ropke, S. (2019). Large neighborhood search. *International Series in Operations Research and Management Science*, *272*, 99–127. https://doi.org/10.1007/978-3-319-91086-4{_}4/TABLES/2

Potvin, J. Y., & Rousseau, J. M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, *66*(3), 331–340. https://doi.org/10.1016/0377-2217(93)90221-8

Qu, Y., & Bard, J. F. (2012). A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, *39*(10), 2439–2456. https://doi.org/10.1016/j.cor.2011.11.016

Raucq, J., Sörensen, K., & Cattrysse, D. (2019). Solving a real-life roll-on–roll-off waste collection problem with column generation. *Journal on Vehicle Routing Algorithms*, *2*(1-4), 41–54. https://doi.org/10.1007/s41604-019-00013-6

REMONDIS UK. (2025, August 7). *Skip hire south shields*. https://www.remondis-uk.com/skip-hire/south-shields/

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472. https://doi.org/10.1287/trsc.1050.0135

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M., II. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, *6*(3), 563–581. https://doi.org/10.1137/0206041

Sakthibala, R. K., Vasanthi, P., Hariharasudhan, C., & Partheeban, P. (2025). A critical review on recycling and reuse of construction and demolition waste materials. *Cleaner Waste Systems*, *12*, 100375. https://doi.org/10.1016/J.CLWAS.2025.100375

Santini, A., Ropke, S., Lars, ·., Hvattum, M., Magnus, L., & No, H. H. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics 2018 24:5*, *24*(5), 783–815. https://doi.org/10.1007/s10732-018-9377-x

Savelsbergh, M. W. P., & Sol, M. (1995). The General Pickup and Delivery Problem. *Transportation Science*, *29*(1), 17–29. https://doi.org/10.1287/trsc.29.1.17

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems.

Shintani, K., Imai, A., Nishimura, E., & Papadimitriou, S. (2007). The container shipping network design problem with empty container repositioning. *Transportation Research Part E: Logistics and Transportation Review*, *43*(1), 39–59. https://doi.org/10.1016/J.TRE.2005.05.003

Tao, Y., & Wang, F. (2015). An effective tabu search approach with improved loading algorithms for the 3L-CVRP. *Computers & Operations Research*, *55*, 127–140. https://doi.org/10.1016/J.COR.2013.10.017

Voigt, S. (2025, April). A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems. https://doi.org/10.1016/j.ejor.2024.05.033

Wøhlk, S., & Laporte, G. (2022). Transport of skips between recycling centers and treatment facilities. *Computers & Operations Research*, *145*, 105879. https://doi.org/10.1016/J.COR.2022.105879

Wy, J., Kim, B. I., & Kim, S. (2013). The rollon–rolloff waste collection vehicle routing problem with time windows. *European Journal of Operational Research*, *224*(3), 466–476. https://doi.org/10.1016/J.EJOR.2012.09.001

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2017). Vehicle routing strategies for pick-up and delivery service under two dimensional loading constraints. *Operational Research*, *17*(1), 115–143. https://doi.org/10.1007/s12351-015-0218-5

Zhang, R., Huang, C., & Wang, J. (2020). A novel mathematical model and a large neighborhood search algorithm for container drayage operations with multi-resource constraints. *Computers & Industrial Engineering*, *139*, 106143. https://doi.org/10.1016/J.CIE.2019.106143

A

Scientific Paper

The scientific paper is added as a PDF and starts on the next page.

Optimising Vehicle Routing for Skip Containers

A Waste Collection Case Study with Container Reuse and Stacking using LNS

Caroline Aalders
Delft University of Technology
Delft, The Netherlands

***Abstract*—Growing volumes of construction and demolition waste create increasing pressure on waste logistics systems. Skip containers are one of the main container types used for this type of waste, yet existing routing research still focuses mainly on large rollon-rolloff (RoRo) containers. As a result, the operational characteristics of smaller chain-lift skip containers remain underexplored. In particular, the possibility of stacking empty containers on a vehicle and directly reusing an empty container between customers has received limited attention in the literature. This paper addresses this gap by studying how vehicle routing for skip container waste collection can be optimised under container reuse and stacking feasibility constraints.**

First, a mixed-integer linear programming (MILP) benchmark model is developed to represent standard skip container operations under simplified assumptions. Second, an extended Large Neighbourhood Search (LNS) metaheuristic is proposed to incorporate more detailed stacking rules, direct container reuse, and multiple storage and disposal location options. The benchmark and metaheuristic models are evaluated using operational and algorithmic key performance indicators.

The validation experiments showed that the LNS algorithm generated solutions within an average deviation below 3% from the benchmark while requiring substantially shorter runtimes. Furthermore, the results show that exact optimisation becomes less practical as the number of customers increases. Across the larger experiments, vehicle configurations with limited full container capacity consistently performed worst, indicating that a primary operational bottleneck lies in the number of available positions for transporting full containers rather than in stacking height alone. In addition, reuse reduced total operational time and improved productivity mainly through reduced handling time, although the number of vehicles required remained unchanged. These findings show that detailed stacking constraints and direct container reuse can be incorporated successfully into skip container routing models and that these extensions improve the practical relevance of optimisation for chain-lift skip operations.

***Index Terms*—chain-lift skip container, skip container routing, waste collection, large neighbourhood search, container stacking, container reuse, vehicle routing problem**

I. Introduction

Construction and demolition activities generate substantial waste flows that require efficient transportation and collection systems. Within the European Union, construction and demolition waste represented approximately 38% of the total waste generated by economic activities and households in 2022 [1]. On a global scale, construction waste is estimated to account for 25% to 40% of total solid waste generation, and this share is expected to continue growing due to increasing urbanisation, infrastructure development, and population growth [2]. Consequently, waste collection and transportation operations form an increasingly important component of urban logistics systems.

Skip containers are one of the main container types used to transport construction and demolition waste [3]. Unlike residential and commercial waste collection, where waste is emptied directly into the collection vehicle, skip container operations involve transporting the container itself between customers, disposal facilities, and storage locations [4], [5]. In practice, several skip container types exist. Most routing studies focus on large rollon-rolloff (RoRo) containers, whereas this research considers smaller chain-lift skip containers that can be stacked on a vehicle. Their stackability allows multiple containers to be transported simultaneously, but also introduces additional operational constraints related to loading feasibility, stacking compatibility, and vehicle utilisation. These aspects are largely ignored in existing RoRo vehicle routing problem (RR VRP) literature [3], [5]–[7].

In addition to stacking, skip container operations may allow the direct reuse of empty containers because the containers are owned by the same company. Direct reuse refers to using an empty container collected from one customer directly at another compatible customer instead of first returning the container to a storage location. This reuse possibility reduces unnecessary storage visits and handling activities, but also introduces additional routing dependencies because the availability of a container depends on preceding operations. Although reuse has been considered in a limited number of skip container routing studies, it is generally not combined with detailed stacking feasibility constraints [6], [8].

Earlier skip container routing studies focused mainly on simplified problem variants and therefore relied primarily on exact optimisation methods [9], [10]. More recent studies have gradually incorporated additional operational realism, including multiple disposal facilities, storage locations, and multiple container types [3], [7], [11], [12]. Nevertheless, detailed stacking feasibility and direct reuse remain largely unexplored within skip container routing models.

Incorporating stacking feasibility and reuse dependencies significantly increases the complexity of the routing problem. Even simplified variants of skip container routing have been shown to be NP-hard [9]. Furthermore, exact optimisation methods become computationally impractical

as routing problems increase in size and complexity, while neighbourhood-based metaheuristics are widely applied to solve larger and more realistic VRPs [13], [14]. Therefore, this paper first develops a mixed-integer linear programming (MILP) benchmark model representing simplified skip container operations and subsequently proposes a Large Neighbourhood Search (LNS) metaheuristic to incorporate detailed stacking feasibility and direct reuse [15], [16].

The main research question is:

> *How can vehicle routing for skip container waste collection be optimised under container reuse and stacking feasibility constraints?*

This paper contributes to the skip container routing literature in three ways. First, it extends the literature beyond traditional RoRo assumptions by explicitly modelling chain-lift skip operations. Second, it develops a LNS metaheuristic that incorporates stacking feasibility and direct reuse dependencies. Third, it provides operational insight into how vehicle configuration and reuse opportunities influence routing performance.

II. Literature Review

Waste collection routing problems are commonly divided into residential, commercial, and RoRo waste collection systems [4]. In RoRo operations, the container itself is transported between customers, disposal facilities, and storage locations rather than being emptied directly into the vehicle [5], [16]. Skip container routing belongs to this category and is therefore modelled as a variant of the RoRo vehicle routing problem (RR VRP).

Existing skip container routing studies mainly focus on large RoRo containers. Most studies assume that a vehicle can transport only one container at a time, while a smaller number of studies allow up to two containers, often representing a truck and trailer combination [3], [6], [7], [9], [12]. However, chain-lift skip containers differ operationally because multiple empty containers can be stacked on a vehicle. This creates additional loading constraints related to stacking feasibility and container compatibility, which are largely ignored in existing routing models. Detailed stacking of multiple skip containers has only been considered to a limited extent in the literature, primarily under simplified operational assumptions [8].

In addition to stacking, several studies recognise that empty containers can potentially be reused directly between customers instead of first returning to a storage location [6], [8]. Direct reuse can reduce unnecessary driving and handling activities by allowing a collected empty container to immediately serve another compatible order. Nevertheless, reuse introduces additional dependencies between routing activities because the availability of a reusable container depends on preceding operations. Existing studies generally consider reuse only in simplified settings and rarely combine it with detailed stacking feasibility constraints. The feasibility of direct reuse depends on the order types involved, since some orders require an empty container while others provide one.

This research includes four main order types:

- **Pickup**: pickup a container at a customer, empty it at a disposal facility, and transport it to a storage location;
- **Delivery**: pickup an empty container at a storage location and deliver it to a customer;
- **Change**: pickup a container at a customer, empty it at a disposal facility, and return it to the customer;
- **Swap**: pickup an empty container at a storage location and deliver it to a customer, then pickup another container at the customer, empty it at a disposal facility, and transport it to a storage location.

The skip container routing literature has gradually incorporated more realistic operational characteristics over time. Earlier studies focused mainly on simplified routing structures and exact optimisation methods [9], [10]. More recent studies extended the scope by introducing multiple disposal facilities, storage locations, flexible return possibilities, and multiple container types [3], [7], [11], [12], [17]. Despite these developments, detailed stacking feasibility and direct reuse remain largely unexplored within skip container routing models.

An overview of the main characteristics of previous skip container and RR VRP studies is provided in Table I. The table shows that most studies still assume limited vehicle capacities and simplified loading structures. Furthermore, reuse is generally modelled only in simplified forms, such as flexible returns or relocation possibilities, while detailed stacking feasibility is rarely incorporated. Existing studies also increasingly shift from exact optimisation methods towards neighbourhood-based metaheuristics as operational complexity increases.
Incorporating detailed stacking feasibility and reuse dependencies substantially increases the complexity of the routing problem. Even simplified variants of skip container routing have been shown to be NP-hard [9]. Consequently, exact optimisation approaches become computationally impractical for larger operational instances. This is also reflected more broadly in vehicle routing literature, where neighbourhood-based metaheuristics are commonly applied to solve complex routing problems efficiently [13], [14]. In particular, Large Neighbourhood Search (LNS) approaches are well suited for handling large solution spaces with complex operational constraints [15].

The literature therefore points to two main research gaps. First, chain-lift skip container routing with detailed stacking feasibility remains largely unexplored. Second, direct container reuse is still underrepresented, particularly in combination with detailed loading feasibility constraints. This research addresses both gaps by explicitly modelling chain-lift skip container operations with stacking feasibility and direct reuse within a Large Neighbourhood Search framework.

TABLE I: Skip Container and RR VRP related papers overview (ordered on publication date).

Reference	Max vehicle Capacity	Stacking	Reuse	Locations	Constraints	Solution Method(s)
Bodin, Mingozzi, Baldacci, *et al.* [5]	1	No	No	SD, SF		PEM, SA, DA, TI^2
Archetti and Speranza [6]	1	No	Reuse	SD, MF	TW, MC	Nearest-neighbourhood
Archetti, Mansini, and Speranza [9]	2	No	No	SD, SF		Exact
Baldacci, Bodin, and Mingozzi [10]	1	No	No	SD, MF, MY		Exact
Benjamin and Beasley [18]	1	No	No	SD, MF	TW	Tabu search & neighbourhood search
Wy, Kim, and Kim [16]	1	No	No	SD, MF, MY	TW, MC	Large neighbourhood search
Hauge, Larsen, Lusby, *et al.* [8]	8	Yes	Reuse	SD, MF, MY	TW, MC	Tabu search & column generation
Aringhieri, Bruglieri, Malucelli, *et al.* [19]	1	No	No	SD, MF	MC	Neighbourhood based
Li, Jian, Chang, *et al.* [11]	1	No	Relocate	SD, MF, MY	TW	MILP & Savings & local search
Raucq, Sörensen, and Cattrysse [7]	2	No	No	MD, MF, MY	TW, MC	Column generation & heuristic pricing
Wøhlk and Laporte [3]	2	No	Flexible return	MD, MF		Variable neighbourhood search
Belenguer, Cubillos, and Wøhlk [12]	2	No	Flexible return	SD, MF		MILP & Branch-and-cut
This research	Variable	Yes	Reuse	SD, MF, MY	MC	Metaheuristic

SD = single depot, MD = multiple depots, SF = single facility, MF = multiple facilities, MY, multiple storage yards, TW = time windows, MC = multiple container types.

III. Problem Description

The benchmark problem defines a simplified reference version of the skip container VRP. The model considers a single planning day with a fixed fleet of vehicles operating between one depot, one storage location, multiple customer locations, and one disposal facility. Furthermore, the benchmark includes one container type, company-owned containers, unlimited container availability, unlimited storage capacity at locations, a maximum stacking height, and fixed handling times. The routing problem consists of four order types: pickup, delivery, change, and swap orders.

Fig. 1: Visual of the extended model elements for a sequenced visit

The extended model builds upon the benchmark formulation by incorporating additional operational realism. Compared to the benchmark, the extended model introduces direct container reuse, multiple container types, more detailed stacking feasibility constraints, multiple storage locations, multiple disposal facilities, and differentiated handling times. In addition, the extended model explicitly considers compatibility between stacked containers and the operational dependencies created by reuse possibilities. The additional modelling elements included in the extended model are illustrated in blue in Figure 1.

The distinction between the two model scopes is important. The benchmark model is intended to provide a simplified reference formulation for verification and comparison purposes. In contrast, the extended model aims to represent the operational complexity of practical chain-lift skip container routing more realistically. Incorporating detailed stacking feasibility and reuse dependencies substantially increases the complexity of the routing problem and therefore motivates the use of a metaheuristic approach instead of relying solely on exact optimisation methods.

IV. Methodology

A. Benchmark MILP Model

A MILP model is developed as a benchmark formulation for the skip container routing problem. The benchmark represents a simplified version of the operational problem and is primarily intended to provide high-quality reference solutions for smaller instances and to support verification and validation of the metaheuristic approach. The formulation excludes direct container reuse and models stacking only through a maximum vehicle capacity.

The benchmark model minimises the total driving time. The handling time is later added in order to compare with the LNS results. Routing decisions determine the sequence of visits between depots, customers, storage locations, and disposal

facilities while ensuring that all orders are completed exactly once. The formulation includes routing continuity constraints, vehicle capacity restrictions, disposal requirements for full containers, and operational sequencing constraints.

B. Large Neighbourhood Search Framework

The extended formulation introduces reuse decision variables that scale as $\mathcal{O}(|R|^2|V|)$ and stacking variables that scale with vehicle positions, stacking levels, and container types, making exact optimisation impractical for realistic problem sizes. LNS is therefore applied, with a destroy-repair mechanism that can handle the interacting stacking, reuse, and precedence constraints effectively. LNS has been applied successfully in earlier RR VRP research [15], [16]. The pseudocode of the main loop is presented in 1.

Initial solution and improvement rules. The initial solution is constructed using a regret-2 insertion heuristic, guided by a k-means clustering of customer locations that are evaluated cluster by cluster in a counter-clockwise sweep around the depot. Orders are inserted as groups of precedence related components, with a new vehicle introduced only when no feasible insertion into existing routes exists. After construction, four improvement rules are applied to reduce operational time by combining visits to customers in the same cluster, the same disposal facility or storage location and by identifying direct reuse opportunities within a route. These rules are also applied after every repair phase.

Solution representation. Each solution consists of a set of vehicle routes represented as ordered sequences of routing components. During each route, the algorithm tracks vehicle loads, container states, and stacking positions. Stacking feasibility constraints ensure that only compatible containers may be stacked and that only the top position within a stack may contain a full container. A candidate route is accepted only if a feasible stacking configuration exists throughout the entire route.

Search operators. Two destroy and two repair operators are each selected with equal probability at every iteration. The destroy operators are random removal and worst-cost removal. Random removal provides diversification by removing a percentage of orders at random. Worst-cost removal promotes intensification by targeting orders with the highest normalised marginal cost. On the repair side, customer with highest position regret and random order, both at the best position, are the operators. Regret-based insertion prioritises the most constrained removed order by selecting the position with the biggest difference between its best and second-best insertion options. Random order insertion shuffles removed orders and inserts each at its locally best feasible position to prevent the search from repeatedly reproducing similar route structures.

Algorithm 1 Large Neighbourhood Search with Linear RRT

Input: feasible initial solution x
Input: Δ_0, K, R
$x^b \leftarrow x$, $x^c \leftarrow x$
$i \leftarrow 1$, $r \leftarrow 0$
while stopping criterion not met **do**
  $(x', feasible) \leftarrow$ REPAIR(DESTROY(x^c))
  if not feasible **then**
    $r \leftarrow r + 1$
    $i \leftarrow i + 1$
    continue
  end if
  $\Delta \leftarrow \Delta_0 \left(1 - \frac{i}{K}\right)$
  if $c(x') < c(x^b)$ **then**
    $x^b \leftarrow x'$, $x^c \leftarrow x'$
    $r \leftarrow 0$
  else if $\frac{c(x') - c(x^b)}{c(x')} \leq \Delta$ **then**
    $x^c \leftarrow x'$
    $r \leftarrow 0$
  else
    $r \leftarrow r + 1$
  end if
  $i \leftarrow i + 1$
end while
return x^b

Acceptance criterion and stopping. Candidate solutions are evaluated using a linear record-to-record travel (RRT) criterion. A candidate x' is accepted as the new current solution if

$$\frac{c(x') - c(x^b)}{c(x')} \leq \Delta_0 \left(1 - \frac{i}{K}\right), \tag{1}$$

where x^b is the best solution found so far, Δ_0 is the initial threshold, and K is the maximum number of iterations. The linearly decreasing threshold promotes diversification early and intensification late. Improving solutions are always accepted, whereas infeasible solutions are always rejected. The search ends when one of three stopping criteria is reached: the iteration limit, the time limit, or a predefined number of consecutive rejections without improvement [16].

V. Experimental Setup

The benchmark test cases consist of five scenarios with customers located in and around Groningen, increasing from three to seven customers. Orders are selected such that each order type is represented at least once. Before solving, order requests from the same customer with the same order type are combined, and a 5% optimality gap is permitted to limit computational time.

The LNS experiments use ten operational data sets together with two additional test cases containing only one container type to isolate the effect of reuse. Before solving, compatible orders are merged subject to vehicle capacity constraints, and

TABLE II: Overview of data set characteristics.

Data set	Total orders	Pickup	Delivery	Swap	Change	Total containers	Container types
Case 1A	30	1	1	17	11	42	8
Case 1B	40	10	10	19	1	48	11
Case 2A	78	11	4	22	41	82	21
Case 2B	70	13	15	35	7	70	16
Case 3	64	9	9	38	8	76	20
Case 4	57	13	9	29	6	58	14
Case 5	38	3	3	17	15	43	15
Case 6	91	7	7	37	40	109	30
Case 7	28	0	1	1	26	31	8
Case 8	29	0	0	4	25	31	11
Test case Case 3	64	9	9	38	8	76	1
Test case Case 4	57	13	9	29	6	58	1

swap orders are decomposed into separate delivery and pickup orders to increase routing flexibility. Table II summarises the order type composition of the data sets, while Table III summarises the available vehicles and facility counts. The data sets vary from 28 to 91 orders, from 1 to 7 depots, from 5 to 16 disposal facilities, and from 2 to 16 storage locations, thereby representing a broad range of operational conditions.

TABLE III: Vehicle and facility counts per data set.

Data set	Vehicles	Depots	Disposal	Storage
Case 1A	11	3	9	2
Case 1B	9	2	5	2
Case 2A	14	3	16	10
Case 2B	11	3	11	10
Case 3	13	2	8	16
Case 4	9	2	8	10
Case 5	12	2	12	10
Case 6	24	7	13	10
Case 7	6	1	10	11
Case 8	6	4	11	12
Test case Case 3	13	2	8	16
Test case Case 4	9	2	8	10

The LNS experiments compare three vehicle configurations: 3×7, 7×3, and 5×4. The notation refers to the number of stacking spots multiplied by stacking height, and the configurations are chosen specifically to test the effect of full container versus empty container capacity. Besides the vehicle configuration, service times are also fixed in the base experiments. The total operational time of a day is set at 540 minutes, while the loading and unloading time of an empty container is 3 minutes and for a full container 10 minutes.

VI. Results Analysis

A. Benchmark versus LNS

The benchmark results show that total operational time increases gradually as the number of customers increases, whereas computational complexity grows exponentially faster. This confirms that exact optimisation becomes less suitable for larger operational cases. Furthermore, the benchmark scenarios show that vehicles are frequently forced to visit disposal facilities because only a limited number of full containers can be transported simultaneously. As the routes become larger and more routing activities must be combined efficiently within the same route, this restriction increasingly acts as an operational bottleneck.

TABLE IV: Comparison between benchmark and LNS results regarding total operational time.

Number of customers	Benchmark	Best LNS	Gap	Avg. LNS	Gap
Without reuse					
3	126.6	126.6	+0.0%	126.6	+0.0%
4	200.4	200.4	+0.0%	200.5	+0.0%
5	227.9	232.4	+2.0%	234.6	+2.9%
6	302.1	311.9	+3.3%	315.3	+4.4%
7	332.6	351.8	+5.8%	351.8	+5.8%
Average			+2.2%		+2.6%
With reuse					
3	126.6	126.6	+0.0%	126.6	+0.0%
4	200.4	194.2	-3.1%	194.2	-3.1%
5	227.9	226.6	-0.6%	229.2	+0.6%
6	302.1	280.6	-7.1%	282.2	-6.6%
7	332.6	316.8	-4.8%	319.4	-4.0%
Average			-3.1%		(absolute) 2.9%

The verification experiments show that the LNS generates solutions close to the benchmark within substantially shorter runtimes. In the reuse scenarios, the metaheuristic even outperforms the benchmark because the benchmark formulation does not incorporate direct reuse possibilities. Across the validation runs, the average gap between the benchmark and the best LNS solution equals 2.2% for the scenarios without reuse. When reuse is enabled, the LNS achieves an average improvement of 3.1% relative to the benchmark. The average absolute deviation across all LNS runs remains limited to 2.9%, as shown in Table IV.

These results demonstrate that the LNS provides solutions of sufficient quality relative to the benchmark while simultaneously incorporating the additional operational realism of stacking feasibility and direct container reuse. The relatively limited deviations therefore indicate that the metaheuristic can effectively approximate the benchmark

solutions while remaining applicable to larger operational scenarios.

B. LNS Results

Table V presents the average total operational time and number of vehicles used for each data set across the three vehicle configurations. A clear and consistent pattern is visible throughout the experiments: the 3×7 configuration has the highest operational time for every data set. Since the 3×7 and 7×3 configurations both provide a total capacity of 21 containers, this difference cannot be explained by total vehicle capacity alone. Instead, the results indicate that the operational performance depends primarily on how the available capacity is distributed between positions for full and empty containers.

TABLE V: Average total operational time and vehicles used across configurations for the data sets.

	Operational time (min)			Vehicles used		
Data set	3×7	7×3	5×4	3×7	7×3	5×4
Case 1A	2594	2215	2240	6	5	5
Case 1B	2336	2310	2266	6	6	5
Case 2A	4071	3908	3915	9	9	9
Case 2B	3657	3556	3552	9	8	9
Case 3	4872	4541	4484	11	10	10
Case 4	3313	3241	3222	7	7	7
Case 5	2455	2399	2445	5	5	6
Case 6	6610	5944	6067	15	13	14
Case 7	1384	1263	1293	3	3	3
Case 8	1498	1380	1396	4	4	4
Case 3 test case	4811	4433	4450	11	10	10
Case 4 test case	3257	3222	3220	7	8	7

The results show that limited capacity for transporting full containers frequently result in additional disposal visits, which increases routing inefficiency and total operational time. As a result, the 7×3 and 5×4 configurations consistently outperform the 3×7 configuration across all data sets. This effect becomes particularly visible in the larger and more operationally complex cases. For example, in Case 3, the number of vehicles decreases from 11 under the 3×7 configuration to 10 under both the 7×3 and 5×4 configurations, while operational time decreases by more than 330 minutes.The differences between the 7×3 and 5×4 configurations are considerably smaller than the differences relative to the 3×7 configuration. In several data sets, the operational time difference remains below 30 minutes. This suggests that once sufficient positions for transporting full containers are available, further increases in stacking height provide only limited additional operational benefit. Nevertheless, the 7×3 configuration achieves the lowest operational time in most data sets, indicating a small but consistent advantage over the 5×4 configuration.

1) Order Types: The composition of order types also appeared to influence routing performance. In particular, swap and change orders showed a stronger sensitivity to the vehicle configuration. Swap orders involve the largest number of routing components, while change orders are operationally restrictive because they require a return to the same customer after visiting a disposal facility. Unlike other order types, where routes can often continue towards a shared storage location, the final movement of a change order is tied to a fixed customer location, thereby reducing routing flexibility.

The results indicate that data sets with a high share of swap and change orders generally achieved larger improvements when using the 7×3 configuration instead of the 3×7 configuration. Case 1A showed the clearest effect, with 93.3% of orders consisting of swap or change orders, resulting in a 14.6% reduction in operational time. Similarly, Case 7 and Case 8 contained predominantly change orders and achieved operational time reductions of 8.7% and 7.9%, respectively, despite being among the smaller data sets. In contrast, Case 1B contained the lowest combined share of swap and change orders and showed only a limited improvement of 1.1%. These findings suggest that change and swap orders increase the sensitivity to vehicle configuration, although the observations remain data set dependent because other characteristics, such as geography and facility availability, also differ between cases.

2) Explore Behaviour: To further examine the behaviour of the LNS algorithm throughout the search process, two runs from the Case 2B data set are analysed. Case 2B showed the largest variation in the number of iterations across all data sets. The selected runs are taken from the 7×3 configuration. Figure 2 shows the run with the highest number of iterations, while Figure 3 shows the run with the lowest number of iterations. Run 4 achieved a final operational time of 3415 minutes, whereas run 10 resulted in 3580 minutes. After the initial solution construction, both runs show a rapid improvement phase in which the LNS algorithm quickly identifies substantially better solutions. This indicates that the destroy and repair operators are effective in exploring promising neighbourhoods during the early stages of the search.

Despite the similar initial improvement phase, both runs exhibit different search behaviour afterwards. Run 4 shows a relatively stable search trajectory, where the current solution remains closer to the incumbent solution and accepted candidate solutions continue to improve operational time over a larger number of iterations. In contrast, run 10 reaches a good solution relatively early, but afterwards the current solution increasingly diverges from the incumbent solution and many candidate solutions are rejected. This suggests that the search becomes more exploratory and struggles to intensify around promising regions of the solution space. The differences between both runs can largely be explained by

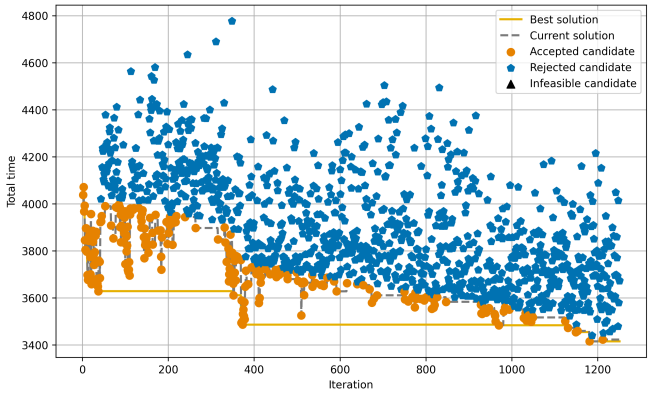


Fig. 2: Improvement graph of Case 2B configuration 7×3 for run 4.

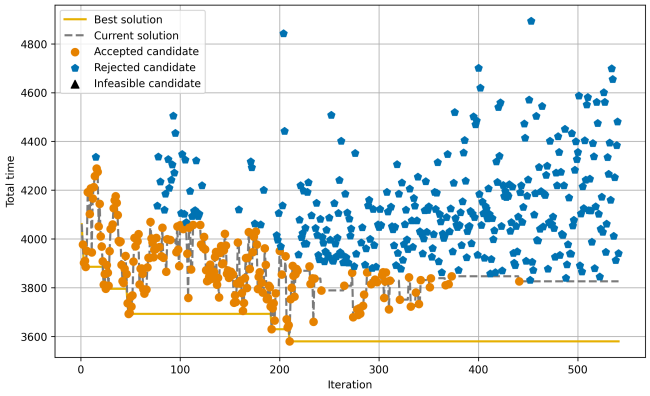


Fig. 3: Improvement graph of Case 2B configuration 7×3 for run 10.

the stochastic nature of the LNS framework, where small differences in the early search trajectory may guide the algorithm towards different regions of the solution space. The results therefore suggest that the current implementation places a relatively strong emphasis on diversification, which helps avoid premature convergence but may reduce the ability to consistently intensify around promising regions during some runs.

3) Route Examples: To further illustrate the routing behaviour of the LNS algorithm, the Case 2A data set is analysed for the 7×3 configuration. The route overview in Table VI shows that the higher full container capacity allows multiple orders at the same customer location to be combined within a single route. This results in higher vehicle utilisation and fewer disposal facility visits within the route structure.

The example further shows that the vehicle transports up to seven full containers simultaneously, which would not be feasible under the 3×7 configuration. The route therefore illustrates how additional full container positions improve consolidation possibilities and reduce operational time. This supports the earlier finding that the number of available positions for full containers forms the main operational bottleneck within the routing problem.

C. Stacking Analysis

The stacking results show that high load values occur across all data sets, although the magnitude differs considerably between cases. Case 6 and Case 1B reach the highest stacking values, with maximum loads up to 15 and 14 containers respectively, while Case 4 and Case 5 remain between 6 and 8 containers. A noticeable pattern is that the highest stacking values mainly occur in data sets containing larger shares of swap and change orders, such as Case 6, Case 1A, and Case 7. These order types involve more routing components and therefore create stronger incentives to transport multiple full containers simultaneously. At the same time, the results indicate that the highest capacities are not utilised consistently throughout all routes, but mainly during locally concentrated route segments where multiple compatible orders can be combined efficiently.

The comparison between the 7×3 and 5×4 configurations further suggests that once sufficient positions for transporting full containers are available, the precise distribution between stacking height and stacking rows has only a limited effect on operational performance. Although both configurations consistently outperform the 3×7 configuration, the operational differences between them remain relatively small across most data sets. This indicates that the primary operational benefit originates from increasing the number of positions available for transporting full containers rather than from maximising stacking height itself.

1) Vehicle Layouts: To further examine how stacking capacity is utilised within the routes, two representative vehicle layouts are analysed at moments of high loading levels. The Case 7 layout below shows that the 7×3 configuration can temporarily transport up to seven full containers simultaneously. This supports the earlier observation that higher full container capacity enables multiple change orders to be clustered before unloading occurs at a disposal facility.

The Case 2B layout further shows that high utilisation is not limited to full containers alone. The vehicle simultaneously transports full, empty, and large containers within the same route arc, illustrating the operational flexibility created by stacking compatibility. At the same time, both layouts show that some stacking rows remain partially unused, indicating that operational improvements mainly originate from combining different container flows efficiently rather than from maximising the total number of stacking rows throughout the entire route.

TABLE VI: Route overview of Case 2A 7×3 run 8 for vehicle 3.

From	To	Driving time (min)	Order	Customer Cluster	Order type	Δ empty	Δ full	Load empty	Load full
D 0	28	0.00	6	1	Change	0	+5	0	5
28	F 12	10.82	6	-	Change	+5	-5	5	0
F 12	28	12.18	6	1	Change	-5	0	0	0
28	28	0.00	7	1	Pickup	0	+3	0	3
28	F 20	7.25	7	-	Pickup	+3	-3	3	0
F 20	S 11	7.47	7	-	Pickup	-3	0	0	0
S 11	28	0.18	5	1	Change	0	+7	0	7
28	F 11	0.18	5	-	Change	+7	-7	7	0
F 11	28	0.18	5	1	Change	-7	0	0	0
28	D 0	0.00	-	-	-	0	0	0	0
Total route time: 428.27 minutes									

$$\begin{bmatrix} F & F & F & F & F & F & F \\ E & – & – & – & – & – & – \\ – & – & – & – & – & – & – \end{bmatrix}$$

Case 7 Vehicle 2

$$\begin{bmatrix} F & F & F & – & – \\ E & E & E & E & E \\ E & E & – & – & – \\ L & L & – & – & – \end{bmatrix}$$

Case 2B Vehicle 3

D. Reuse Analysis

Additional test cases for Case 3 and Case 4 were analysed in which the distinction between container types was removed to create more opportunities for direct container reuse. The results in Table VII show that the reuse rate increases consistently across all configurations. For Case 3, the average increase equals 7.0%, while Case 4 achieves an average increase of 10.3%. At the same time, total operational time decreases in every configuration, with reductions ranging between 34 and 108 minutes.

TABLE VII: Average KPI comparison between original data sets and reuse test cases.

Data set	KPI ↓ \ config →	3×7	7×3	5×4
Case 3 vs testcase	reuse rate (%)	+7.3	+6.5	+7.3
	operational time (min)	-61	-108	-34
	vehicles used	0	0	0
	load per arc	-0.03	+0.02	-0.14
	orders per hour	+0.01	+0.02	+0.00
Case 4 vs testcase	reuse rate (%)	+10.8	+11.1	+8.9
	operational time (min)	-106	-100	-97
	vehicles used	0	0	0
	load per arc	-0.08	+0.03	+0.10
	orders per hour	+0.04	+0.03	+0.03

Despite these improvements, the average number of vehicles remains unchanged in all cases. This indicates that reuse primarily improves routing efficiency through reduced handling activities and more efficient route structures rather than through fleet size reductions. The load per arc KPI shows both small increases and decreases, suggesting that reuse affects route utilisation differently depending on the route structure. In some cases, reuse enables additional orders to be combined within the same route, while in other cases it mainly reduces unnecessary container movements without substantially changing the route loading structure. Overall, the results consistently show that direct reuse contributes positively to routing efficiency, although the operational improvements remain relatively limited.

E. Sensitivity Analysis

A sensitivity analysis was performed to examine how the routing results change when service times vary. Two data sets, Case 5 and Case 8, were analysed using lower service times ($0.75\times$), the base scenario, and higher service times ($1.25\times$). The results in Table VIII show that lower service times consistently improve the main KPIs. For both data sets, the total operational time decreases substantially, while productivity in terms of *orders per hour* increases. In addition, Case 8 requires only three vehicles in the $0.75\times$ scenario instead of four in the base case. The results also show higher stacking utilisation under lower service times, indicating that more orders can be combined efficiently within the same routes.

TABLE VIII: Sensitivity analysis summary for different service time levels (average results across runs).

Data set	Scenario	Best config	Total time	Vehicles	Orders/hour	Stack util.
Case 5	$0.75\times$	5×4	2099	5	1.09	0.16
Case 5	Base	7×3	2399	5	0.95	0.15
Case 5	$1.25\times$	5×4	2719	6	0.84	0.13
Case 8	$0.75\times$	7×3	1155	3	1.51	0.22
Case 8	Base	7×3	1380	4	1.26	0.17
Case 8	$1.25\times$	7×3	1598	4	1.09	0.15

When service times increase, the opposite pattern emerges. Total operational time rises, while productivity and stacking utilisation decrease. For Case 5, the average vehicle usage increases from five vehicles in the base case to six vehicles in the $1.25\times$ scenario. This indicates that longer handling activities reduce operational flexibility and limit the ability to combine additional containers within the same routes. Although the preferred vehicle configuration changes slightly for Case 5, where the 5×4 configuration performs marginally better in the alternative scenarios, the differences between the 7×3 and 5×4 configurations remain limited overall.

The sensitivity analysis, therefore, confirms that the LNS framework responds consistently to changing service time assumptions.

VII. Discussion

The results show that incorporating detailed stacking feasibility and direct container reuse improves the practical relevance of skip container routing models. The computational experiments demonstrated that the MILP formulation provides useful benchmark solutions for small cases, but becomes computationally impractical as the number of customers and operational constraints increases. In contrast, the LNS framework generated near optimal solutions within substantially shorter runtimes while remaining capable of handling stacking feasibility, multiple facilities, and reuse dependencies simultaneously. This confirms the suitability of neighbourhood-based metaheuristics for realistic skip container routing problems.

From an operational perspective, the experiments consistently showed that the primary bottleneck lies in the number of positions available for transporting full containers rather than in total stacking height alone. Configurations with more full container positions achieved lower operational times, fewer disposal visits, and, in some cases, lower vehicle usage. At the same time, the vehicle layout analysis showed that the highest stacking levels mainly occur during temporary peak loading moments within locally concentrated route segments rather than continuously throughout the routes. Furthermore, reuse consistently improved operational efficiency, although the benefits remained relatively limited and mainly originated from reduced handling activities rather than from eliminating complete route stops. The experiments also showed that routing performance depends strongly on data set structure, facility restrictions, and order composition, particularly for data sets containing many change orders.

Several limitations remain. The analysed scenarios are deterministic and based on single day planning scenario, meaning that uncertainty in travel times, demand, and container availability is not considered. In addition, practical constraints such as time windows, waiting times, driver regulations, and limited storage capacities were excluded from the current scope. The current model also assumes that all orders involve stackable chain-lift skip containers, whereas real operations may simultaneously involve other skip container types with different operational characteristics. Finally, the stochastic nature of the LNS framework leads to variation across runs, indicating that multiple runs should be performed in practice to obtain consistently strong solutions.

VIII. Conclusion

This paper studied how vehicle routing for skip container waste collection can be optimised under container reuse and stacking feasibility constraints. A benchmark MILP formulation was first developed as a simplified reference model, after which a LNS metaheuristic was proposed to incorporate stacking feasibility, direct reuse, multiple disposal facilities, and multiple storage locations.

The results showed that the LNS framework generates solutions close to the benchmark while requiring substantially shorter runtimes. In the reuse verification scenarios, the LNS even outperformed the benchmark by exploiting direct reuse opportunities between customers. Across the operational data sets, the 3×7 configuration consistently performed worst, indicating that limited full container capacity forms the primary operational bottleneck. Once sufficient full container positions are available, the operational differences between the 7×3 and 5×4 configurations become relatively small. The experiments further showed that high stacking levels are actively used within practical route structures, while direct reuse contributes positively to routing efficiency through reduced handling activities and more efficient route combinations.

The research, therefore, extends the skip container routing literature towards a more realistic representation of chain-lift skip operations by explicitly incorporating stacking feasibility and direct container reuse. In addition, the results provide practical insight into how vehicle configurations, order composition, and reuse opportunities influence routing performance. Future research could further extend the framework with stochastic travel times, time windows, limited container availability, or adaptive operator selection within an ALNS framework.

References

[1] Eurostat. "Waste statistics." (Sep. 2024), [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Waste_statistics.

[2] R. K. Sakthibala, P. Vasanthi, C. Hariharasudhan, and P. Partheeban, "A critical review on recycling and reuse of construction and demolition waste materials," *Cleaner Waste Systems*, vol. 12, p. 100 375, Dec. 2025, ISSN: 2772-9125. DOI: 10.1016/J.CLWAS.2025.100375. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772912525001733?.

[3] S. Wøhlk and G. Laporte, "Transport of skips between recycling centers and treatment facilities," *Computers & Operations Research*, vol. 145, p. 105 879, Sep. 2022, ISSN: 0305-0548. DOI: 10.1016/J.COR.2022.105879. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054822001502?via%3Dihub.

[4] B. L. Golden, A. A. Assad, E. A. Wasil, G. B. Dantzig, and J. H. Ramser, "Routing Vehicles in the Real World: Applications in the Solid Waste, Beverage, Food, Dairy, and Newspaper Industries 10.1 Introduction," Tech.

Rep., 2002. [Online]. Available: http://www.siam.org/journals/ojsa.php.

[5] L. Bodin, A. Mingozzi, R. Baldacci, and M. Ball, "Rollon-Rolloff vehicle routing problem," *Transportation Science*, vol. 34, no. 3, pp. 271–288, 2000, ISSN: 00411655. DOI: 10.1287/trsc.34.3.271.12301.

[6] C. Archetti and M. G. Speranza, "Vehicle routing in the 1-skip collection problem," *Journal of the Operational Research Society*, vol. 55, no. 7, pp. 717–727, 2004, ISSN: 01605682. DOI: 10.1057/palgrave.jors.2601743.

[7] J. Raucq, K. Sörensen, and D. Cattrysse, "Solving a real-life roll-on–roll-off waste collection problem with column generation," *Journal on Vehicle Routing Algorithms*, vol. 2, no. 1-4, pp. 41–54, Dec. 2019, ISSN: 2367-3591. DOI: 10.1007/s41604-019-00013-6.

[8] K. Hauge, J. Larsen, R. M. Lusby, and E. Krapper, "A hybrid column generation approach for an industrial waste collection routing problem," *Computers & Industrial Engineering*, vol. 71, no. 1, pp. 10–20, May 2014, ISSN: 0360-8352. DOI: 10.1016/J.CIE.2014.02.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835214000485.

[9] C. Archetti, R. Mansini, and M. G. Speranza, "Complexity and reducibility of the skip delivery problem," *Transportation Science*, vol. 39, no. 2, pp. 182–187, 2005, ISSN: 00411655. DOI: 10.1287/trsc.1030.0084.

[10] R. Baldacci, L. Bodin, and A. Mingozzi, "The multiple disposal facilities and multiple inventory locations rollon–rolloff vehicle routing problem," *Computers & Operations Research*, vol. 33, no. 9, pp. 2667–2702, Sep. 2006, ISSN: 0305-0548. DOI: 10.1016/J.COR.2005.02.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054805000791?via%3Dihub.

[11] H. Li, X. Jian, X. Chang, and Y. Lu, "The generalized rollon-rolloff vehicle routing problem and savings-based algorithm," *Transportation Research Part B: Methodological*, vol. 113, pp. 1–23, Jul. 2018, ISSN: 0191-2615. DOI: 10.1016/J.TRB.2018.05.005. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0191261517311177.

[12] J. M. Belenguer, M. Cubillos, and S. Wøhlk, "A branch-and-cut algorithm for a skip pick-up and delivery problem," *Computers & Operations Research*, vol. 168, p. 106 705, Aug. 2024, ISSN: 0305-0548. DOI: 10.1016/J.COR.2024.106705. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054824001771?via%3Dihub.

[13] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, *The vehicle routing problem: State of the art classification and review*, Sep. 2016. DOI: 10.1016/j.cie.2015.12.007.

[14] R. Elshaer and H. Awad, "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants," *Computers and Industrial Engineering*, vol. 140, Feb. 2020, ISSN: 03608352. DOI: 10.1016/j.cie.2019.106242.

[15] D. Pisinger and S. Ropke, "Large neighborhood search," *International Series in Operations Research and Management Science*, vol. 272, pp. 99–127, 2019, ISSN: 08848289. DOI: 10.1007/978-3-319-91086-4{_}4/TABLES/2. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-91086-4_4.

[16] J. Wy, B. I. Kim, and S. Kim, "The rollon–rolloff waste collection vehicle routing problem with time windows," *European Journal of Operational Research*, vol. 224, no. 3, pp. 466–476, Feb. 2013, ISSN: 0377-2217. DOI: 10.1016/J.EJOR.2012.09.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221712006649?via%3Dihub.

[17] C. Hess, A. G. Dragomir, K. F. Doerner, and D. Vigo, "Waste collection routing: a survey on problems and methods," *Central European Journal of Operations Research*, vol. 32, no. 2, pp. 399–434, Jun. 2024, ISSN: 16139178. DOI: 10.1007/s10100-023-00892-y.

[18] A. M. Benjamin and J. E. Beasley, "Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities," *Computers & Operations Research*, vol. 37, no. 12, pp. 2270–2280, Dec. 2010, ISSN: 0305-0548. DOI: 10.1016/J.COR.2010.03.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054810000730?via%3Dihub.

[19] R. Aringhieri, M. Bruglieri, F. Malucelli, and M. Nonato, "A special vehicle routing problem arising in the optimization of waste disposal: A real case," *Transportation Science*, vol. 52, no. 2, pp. 277–279, Mar. 2018, ISSN: 15265447. DOI: 10.1287/trsc.2016.0731.

B

Literature Overview

B.1. Search Words

These are the search words where most papers where found.

- ”VRP” AND ”review”
- ”Routing” OR ”VRP” AND ”Waste collection”
- ”Waste collection” AND ”skip”
- ”RR VRP”
- ”Rollon-Rolloff” OR ”RoRo” OR ”Roll-on-Roll-off” AND ”VRP” OR ”Vehicle routing problem”
- ”Rollon–rolloff vehicle routing problem”
- ”skip transport”
- ”bin packing problem” AND ”truck”
- ”container” AND ”stacking” AND ”terminal” OR ”yard”
- ”review” AND ”VRP” AND ”metaheuristic”
- ”review” AND ”LNS” OR ”ALNS”
- ”large neighbourhood search” OR ”LNS”
- ”operators” AND ”metaheuristic” AND ”neighbourhood”
- ”LNS” AND ”VRP” AND ”Acceptance”

B.2. Research Aspects in Previous Studies

In order of being cited in the literature review.

Table B.1: Literature review paper overview.

Literature	Waste collection	Skip Container/RR VRP	Stacking	Order types	Container reuse in other fields	PDP	Time windows	Multiple locations	(Meta)heuristics	Exact methods
This research	*	*	*	*				*	*	*
Golden et al. (2002)	*	*		*			*			
Bodin et al. (2000)	*	*		*		*			*	
Wy et al. (2013)	*	*		*			*	*	*	
Wøhlk and Laporte (2022)	*			*		*		*	*	
Archetti and Speranza (2004)	*	*					*			
Aringhieri et al. (2018)	*	*		*				*	*	
Raucq et al. (2019)	*	*		*			*	*		
Hess et al. (2024)	*	*				*			*	*
Archetti et al. (2005)		*								*
Belenguer et al. (2024)	*					*		*		*
Hauge et al. (2014)	*	*	*	*				*	*	
Li et al. (2018)	*	*		*			*	*	*	*
Henke et al. (2015)	*									
Zachariadis et al. (2017)						*			*	
Fuellerer et al. (2010)			*						*	
Tao and Wang (2015)				*					*	
Junqueira et al. (2012)			*							*
Paquay et al. (2016)			*							*
Gunawardhana et al. (2021)			*						*	
Dekker et al. (2006)			*							
Baldacci et al. (2006)	*	*		*						*
Fazi et al. (2023)				*	*		*	*		*
Zhang et al. (2020)				*	*		*		*	*
Lee and Song (2017)					*					
Shintani et al. (2007)					*				*	
Meng and Wang (2011)						*				*
Dantzig and Ramser (1959)										*
Dumas et al. (1991)						*	*	*		*
Laporte (2009)									*	*
Savelsbergh and Sol (1995)						*	*	*		*
Cortés et al. (2010)						*	*	*		*
Naccache et al. (2018)						*	*	*	*	*
Benjamin and Beasley (2010)	*	*					*	*	*	
Archetti et al. (2025)						*			*	
Braekers et al. (2016)									*	*
Kim et al. (2006)	*						*	*	*	
Elshaer and Awad (2020)									*	
Boussaïd et al. (2013)									*	
Rosenkrantz et al. (1977)									*	
Potvin and Rousseau (1993)						*	*		*	
Ropke and Pisinger (2006)						*	*	*	*	
Clarke and Wright (1964)									*	
Liu et al. (2023)								*		
Voigt (2025)									*	

C

Data Sets Overview

This appendix container the maps of the data sets. The point on the maps mean the following:

- **Red:** customer
- **Blue + bin:** disposal facility
- **Green + storage:** storage location
- **Black + house:** vehicle depot
- **Blue + arrows:** a combination between disposal facility, storage location, and/or vehicle depot

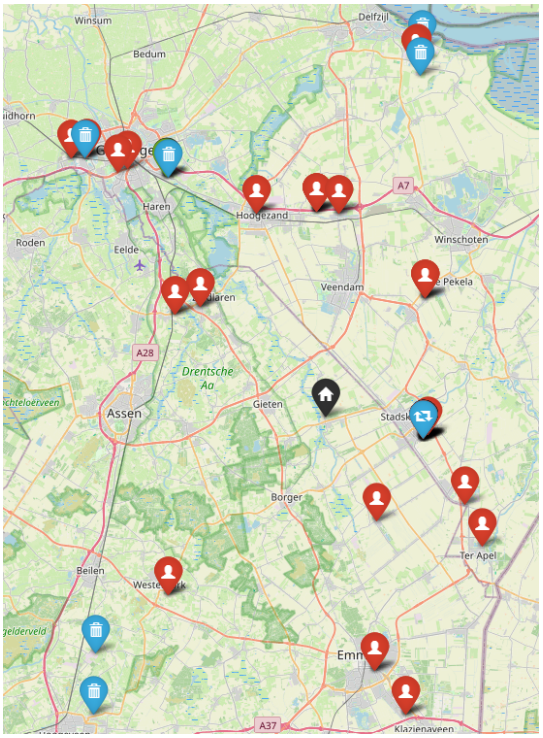


Figure C.1: Map for data set Case 1A.

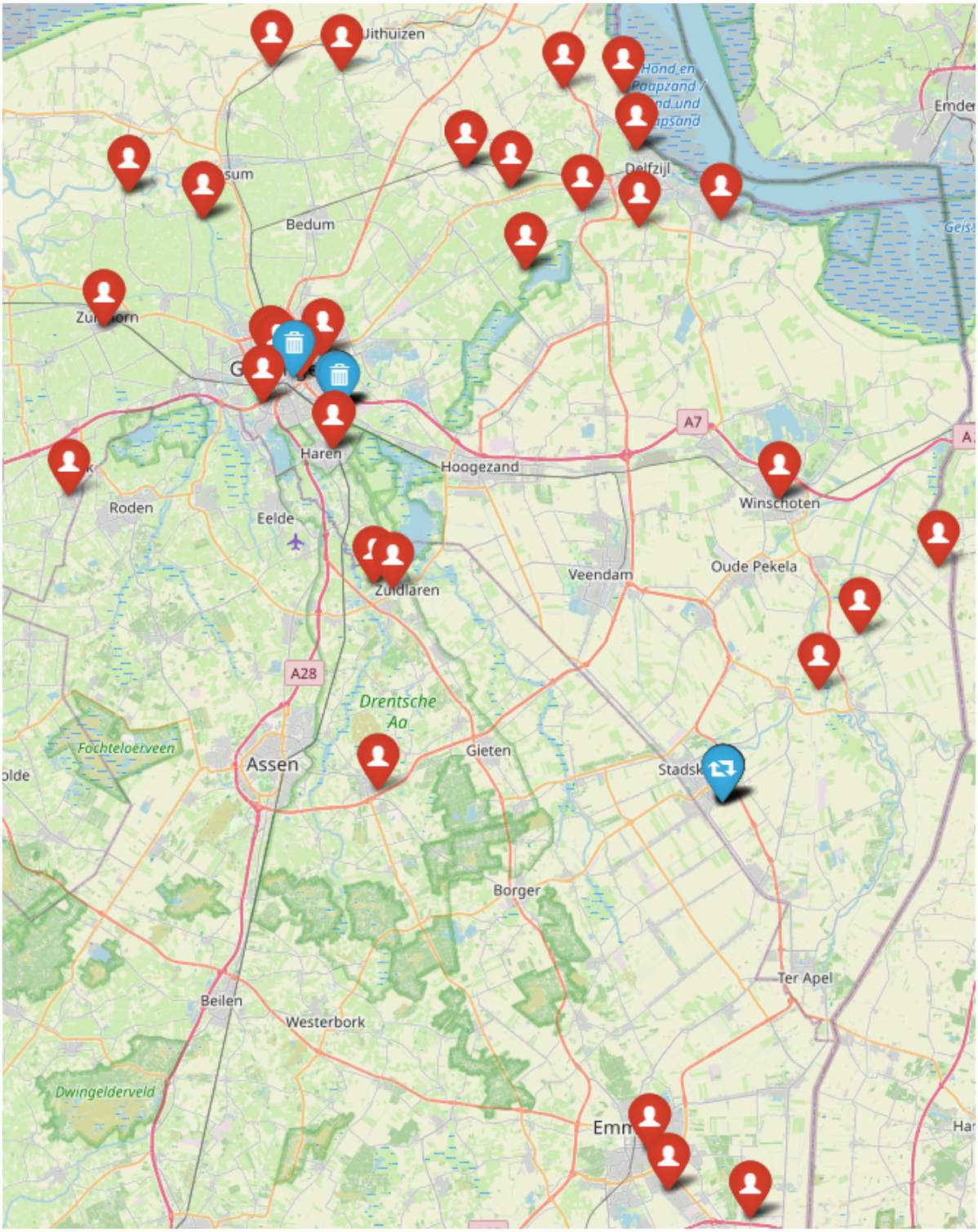


Figure C.2: Map for data set Case 1B.

Figure C.3: Map for data set Case 2A.

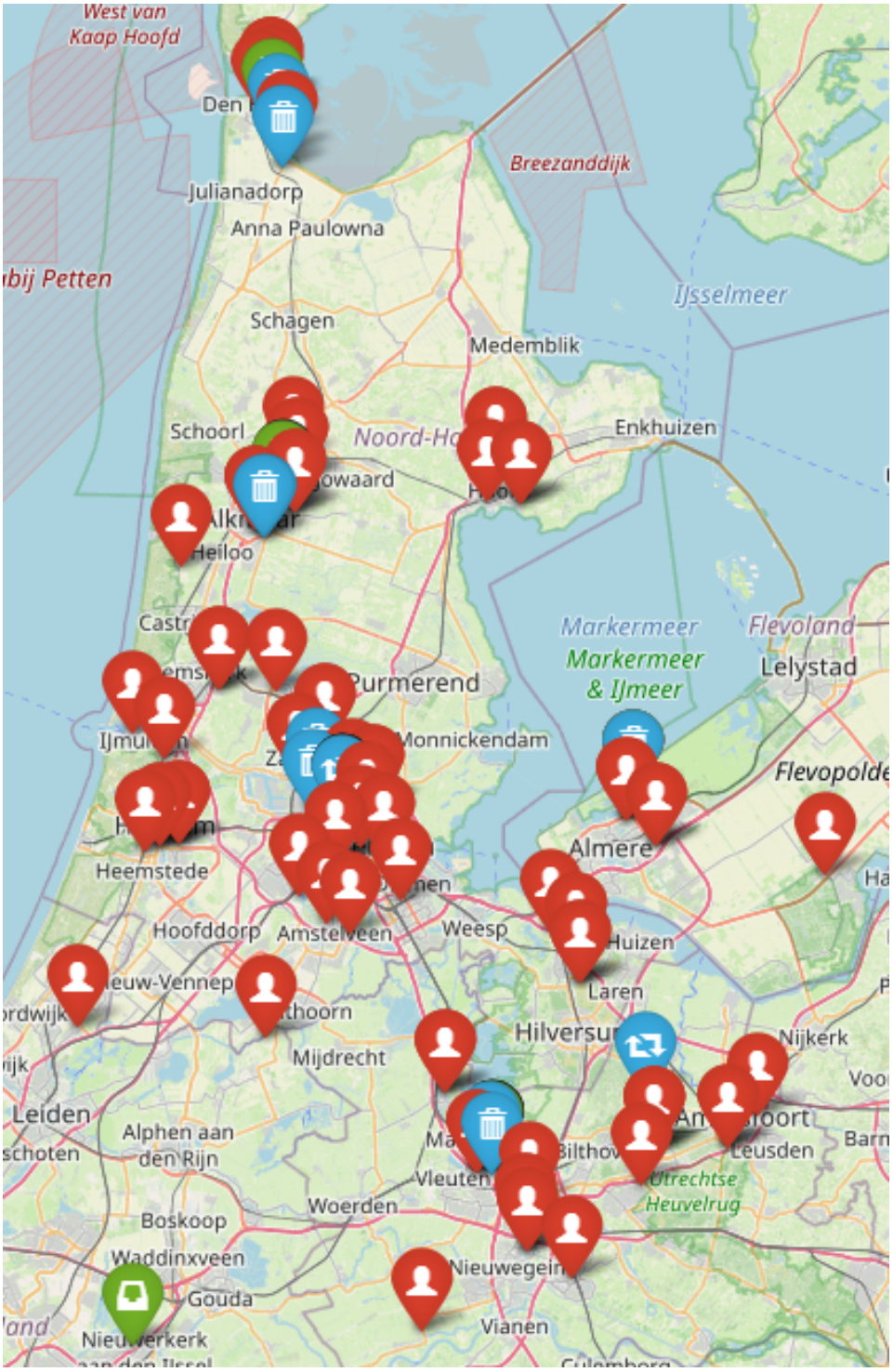


Figure C.4: Map for data set Case 2B.

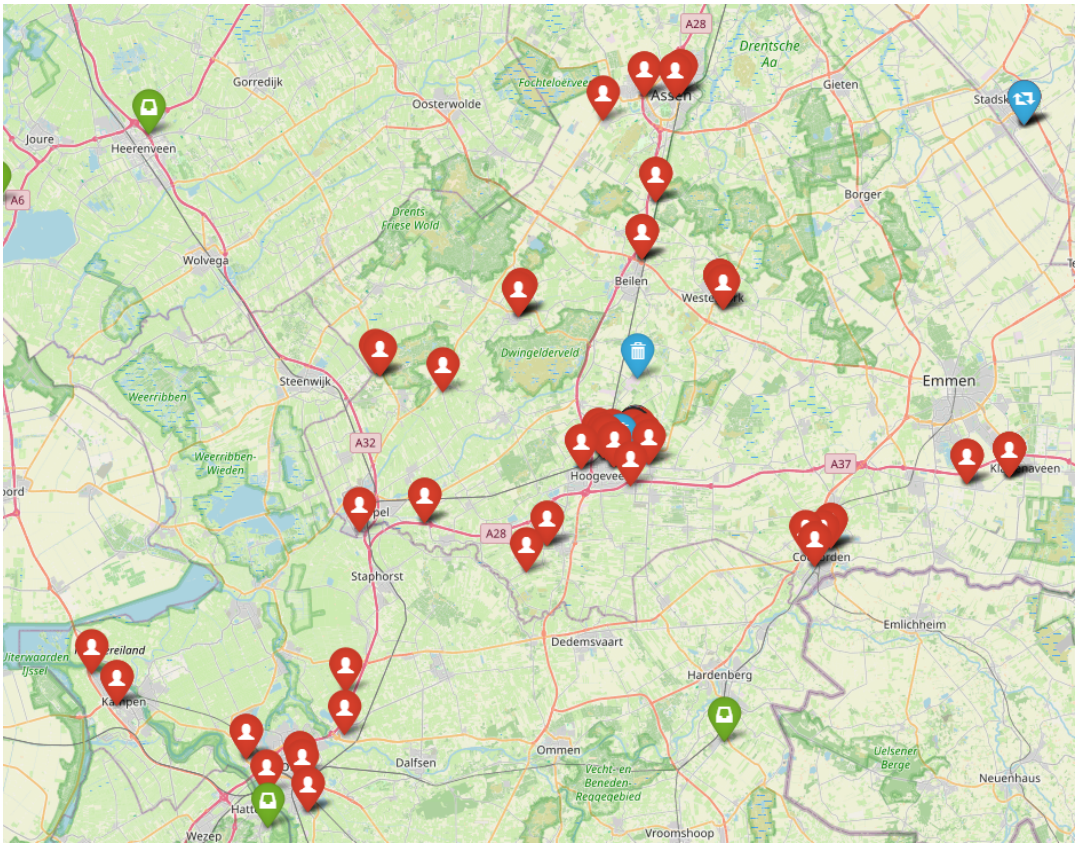


Figure C.5: Map for data set Case 3.

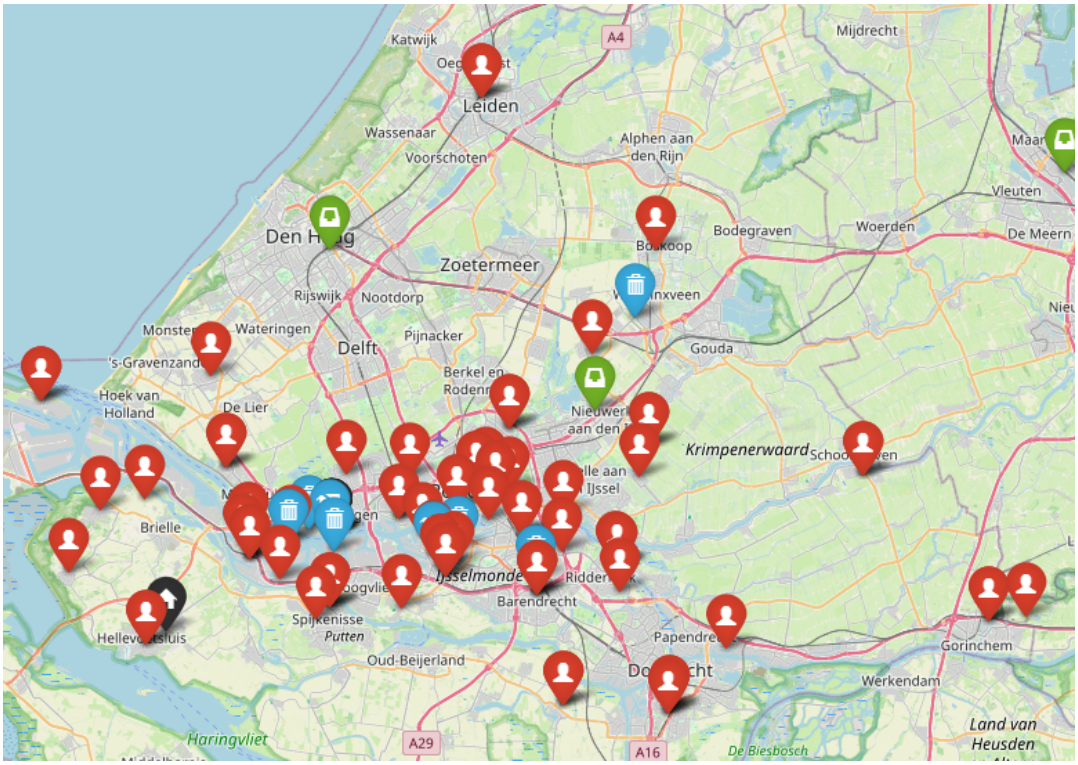


Figure C.6: Map for data set Case 4.

Figure C.7: Map for data set Case 5.

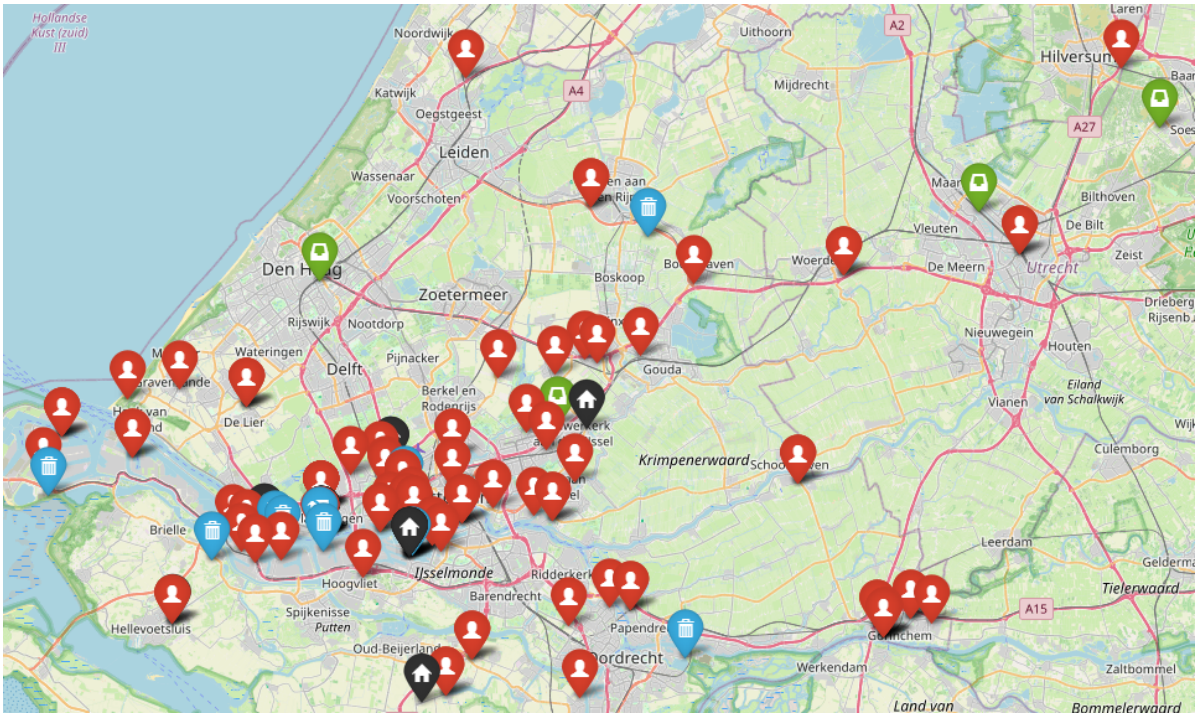


Figure C.8: Map for data set Case 6.

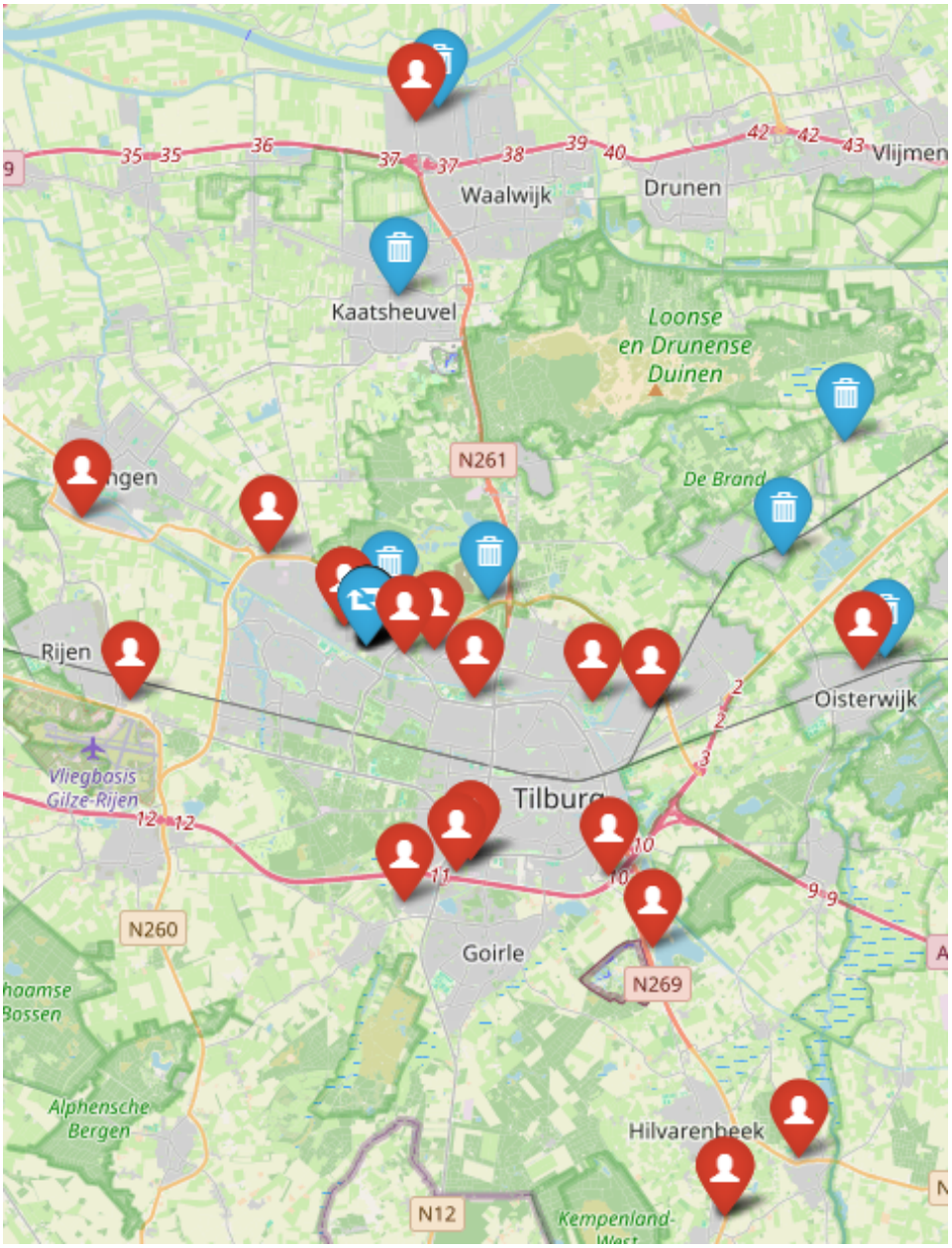


Figure C.9: Map for data set Case 7.

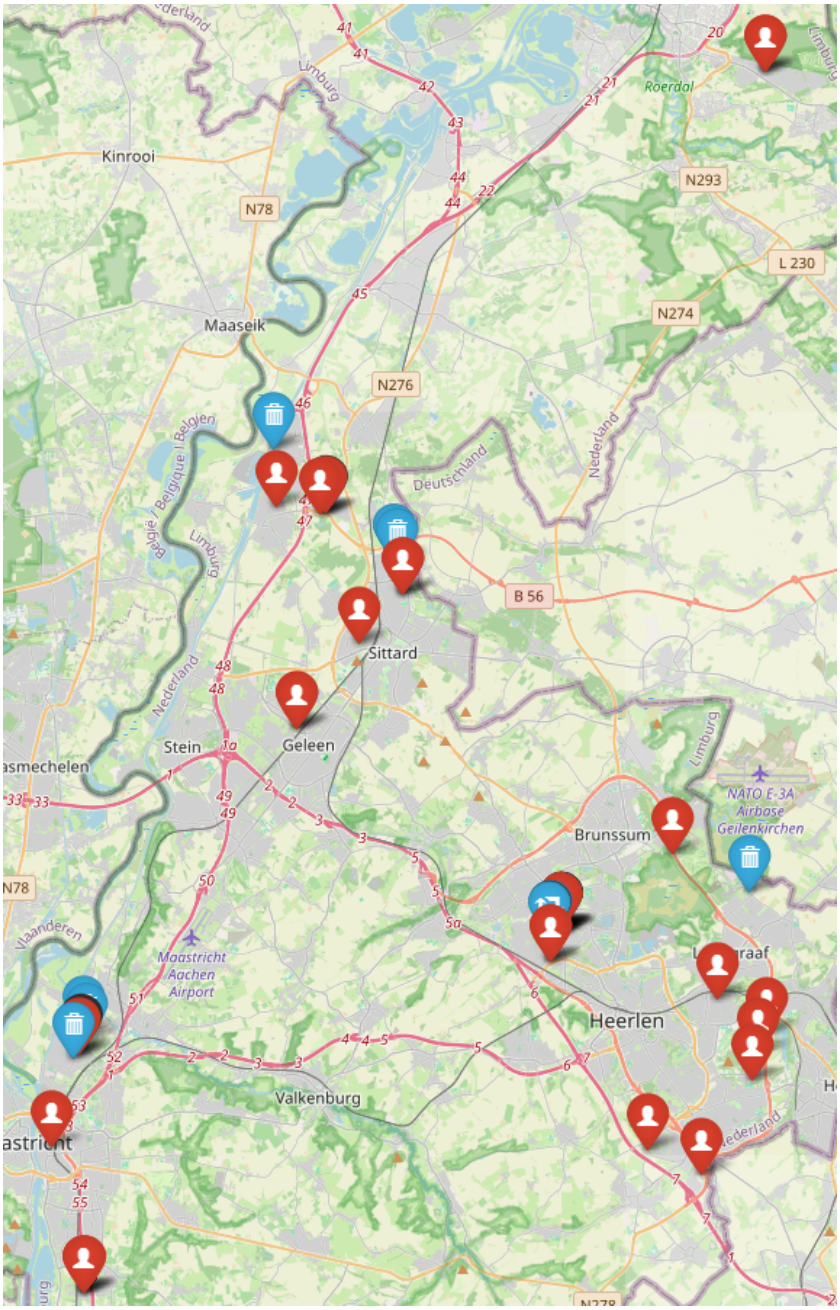


Figure C.10: Map for data set Case 8.

D

Pseudocodes

Here is an overview of pseudocodes that describe different parts of the LNS algorithm.

Algorithm 2 Cluster-Based Route Regret-2 Construction

Input: orders R, vehicles V, clustering $\mathcal{C}$
Output: initial solution x
$x \leftarrow$ empty solution
for all $C \in \mathcal{C}$ **do** ▷ Set of uninserted orders
    $U \leftarrow C$
    while $U \neq \emptyset$ **do**
        for all $r \in U$ **do**
            $c_1(r), c_2(r)$ ▷ best and second-best insertion costs
            $\rho(r) \leftarrow c_2(r) - c_1(r)$
        end for
        $r^* \leftarrow \arg\max_{r \in U} \rho(r)$
        $x \leftarrow$ insert r^* into x at cost $c_1(r^*)$
        $U \leftarrow U \setminus \{r^*\}$
    end while
end for
Apply improvement rules to x
return x

Algorithm 3 Random Removal

Input: solution x
Output: partial solution x', removed set R^{-}
$R \leftarrow$ set of orders in x
$q \leftarrow \max\{1, \lfloor \alpha |R| \rfloor\}$ ▷ α: destroy fraction
$R^{-} \leftarrow$ random subset of R with $|R^{-}| = q$
for all $r \in R^{-}$ **do**
    Remove r from x
end for
Recompute x'
return x', R^{-}

Algorithm 4 Worst-Cost Removal

Input: solution x
Output: partial solution x', removed set R^{-}
$R \leftarrow$ set of orders in x
$q \leftarrow \max\{1, \lfloor \alpha |R| \rfloor\}$ ▷ α: destroy fraction
for all $r \in R$ **do**
    $x^{-r} \leftarrow$ solution obtained by removing r from x
    $\Delta(r) \leftarrow \dfrac{c(x) - c(x^{-r})}{|C_r|}$ ▷ Normalised marginal cost
end for
$R^{-} \leftarrow$ set of q orders with largest $\Delta(r)$
for all $r \in R^{-}$ **do**
    Remove r from x
end for
Recompute x'
return x', R^{-}

Algorithm 5 Regret-Based Repair

Input: partial solution x, removed orders R^{-}
Output: repaired solution x'
$U \leftarrow R^{-}$
while $U \neq \emptyset$ **do**
    for all $r \in U$ **do**
        Determine $c_1(r)$ and $c_2(r)$
        $\rho(r) \leftarrow c_2(r) - c_1(r)$
    end for
    $r^{*} \leftarrow \arg\max_{r \in U} \rho(r)$
    $x \leftarrow$ insert r^{*} into x at cost $c_1(r^{*})$
    $U \leftarrow U \setminus \{r^{*}\}$
end while
Apply improvement rules to x
$x' \leftarrow x$
return x'

Algorithm 6 Random Repair

Input: partial solution x, removed orders R^-
Output: repaired solution x'
Shuffle R^-
for all $r \in R^-$ **do**
    Determine $c_1(r)$
    $x \leftarrow$ insert r into x at cost $c_1(r)$
end for
Apply improvement rules to x
$x' \leftarrow x$
return x'

Algorithm 7 Record-to-Record Travel Acceptance

Input: candidate solution x', current solution x^c, best solution x^b
Input: iteration i, maximum iterations K, initial threshold Δ_0
Output: updated (x^b, x^c)
if x' infeasible **then**
    return (x^b, x^c)
end if
$\Delta \leftarrow \Delta_0 \left(1 - \frac{i}{K}\right)$
if $c(x') < c(x^b)$ **then**
    $x^b \leftarrow x'$
    $x^c \leftarrow x'$
else if $\dfrac{c(x') - c(x^b)}{c(x')} \leq \Delta$ **then**
    $x^c \leftarrow x'$
end if
return (x^b, x^c)

E

Results

In this appendix, an overview of the KPIs for each data set is presented. It begins with additional results from the benchmark model alongside the validation results of the LNS algorithm. Subsequently, the results of the LNS algorithm are presented in alphabetical order, including two additional test cases for reuse, and conclude with the sensitivity analysis.

Benchmark Model

Table E.1: Operational and computational performance with reuse for increasing numbers of customers. The LNS results are averaged over 5 runs.

KPI	Units	3 cust.	3 cust.	4 cust.	4 cust.	5 cust.	5 cust.	6 cust.	6 cust.	7 cust.	7 cust.
		BM	LNS	BM	LNS	BM	LNS	BM	LNS	BM	LNS
Total operational time	min	126.6	126.6	200.4	194.2	227.9	229.2	302.1	282.2	332.6	319.4
Vehicles used	#	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Container reuse rate	%	0.0	0.0	0.0	11.0	0.0	10.0	0.0	16	0.0	15.3
Avg. vehicle load per arc	#	2.0	1.8	4.0	2.3	4.2	2.3	5.9	3.0	7.2	2.6
Avg. stacking utilisation	%	0.2	0.1	0.4	0.2	0.4	0.2	0.6	0.2	0.7	0.2
Avg. travel time per container	min	31.7	31.7	33.4	32.4	32.6	32.7	33.6	31.4	33.3	31.9
Order per hour completed	#	1.9	1.9	1.5	1.5	1.6	1.6	1.4	1.5	1.4	1.5
Computational time	min	0.0	0.2	0.1	0.4	0.1	0.6	7.3	0.6	230.6	0.9
Optimality gap	%	0.0	-	4.4	-	4.9	-	5.0	-	5.0	-

Table E.2: Operational and computational performance without reuse for increasing numbers of customers. The LNS results are averaged over 5 runs.

KPI	Units	3 cust.	3 cust.	4 cust.	4 cust.	5 cust.	5 cust.	6 cust.	6 cust.	7 cust.	7 cust.
		BM	LNS	BM	LNS	BM	LNS	BM	LNS	BM	LNS
Total operational time	min	126.6	126.6	200.4	200.5	227.9	234.6	302.1	315.3	332.6	351.8
Vehicles used	#	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Container reuse rate	%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Avg. vehicle load per arc	#	2.0	1.8	4.0	3.5	4.2	3.2	5.9	3.4	7.2	3.7
Avg. stacking utilisation	%	0.2	0.1	0.4	0.3	0.4	0.3	0.6	0.3	0.7	0.3
Avg. travel time per container	min	31.7	31.7	33.4	33.4	32.6	33.5	33.6	35.0	33.3	35.2
Order per hour completed	#	1.9	1.9	1.5	1.5	1.6	1.5	1.4	1.3	1.4	1.4
Computational time	min	0.0	0.2	0.1	0.3	0.1	0.6	7.3	0.6	230.6	1.0
Optimality gap	%	0.0	-	4.4	-	4.9	-	5.0	-	5.0	-

LNS Algorithm

Table E.3: KPI comparison across configurations for Case 1A

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 6	-	run 8	-	run 4	-
iterations	1136	883	1093	859	666	790
total operational time	2525	2594	2137	2215	2144	2240
vehicles used	6	6	5	5	5	5
container reuse rate	2.0%	5.1%	3.5%	4.1%	3.5%	4.5%
avg load per arc	2.70	2.45	4.10	3.91	4.40	3.22
avg stacking utilisation	0.13	0.12	0.20	0.19	0.22	0.16
avg travel time per container	60.12	61.76	50.87	52.74	51.04	53.32
orders per hour	0.71	0.69	0.84	0.81	0.84	0.80
runtime (min)	1.03	0.77	0.72	0.54	0.42	0.48

Table E.4: KPI comparison across configurations for Case 1B

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 10	-	run 3	-	run 10	-
iterations	731	681	1105	606	436	673
total operational time	2271	2336	2273	2310	2124	2266
vehicles used	5	6	5	6	5	5
container reuse rate	3.1%	4.4%	4.6%	3.4%	5.3%	5.0%
avg load per arc	3.63	2.97	3.95	3.62	3.59	3.36
avg stacking utilisation	0.17	0.14	0.19	0.17	0.18	0.17
avg travel time per container	47.31	48.66	47.36	48.13	44.26	47.20
orders per hour	1.06	1.03	1.06	1.04	1.13	1.06
runtime (min)	1.76	1.57	2.38	1.33	1.04	1.53

Table E.5: KPI comparison across configurations for Case 2A

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 4	-	run 8	-	run 2	-
iterations	1250	1194	1250	1112	1250	1056
total operational time	4005	4071	3835	3908	3814	3915
vehicles used	9	9	8	9	8	9
container reuse rate	2.2%	1.4%	1.2%	1.1%	1.2%	1.3%
avg load per arc	2.65	2.38	3.44	3.15	3.21	2.82
avg stacking utilisation	0.13	0.11	0.16	0.15	0.16	0.14
avg travel time per container	48.84	49.65	46.77	47.65	46.51	47.74
orders per hour	1.17	1.15	1.22	1.20	1.23	1.20
runtime (min)	4.99	4.75	3.40	3.14	4.00	3.28

Table E.6: KPI comparison across configurations for Case 2B

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 8	-	run 4	-	run 3	-
iterations	859	655	1250	898	935	723
total operational time	3520	3657	3415	3556	3462	3552
vehicles used	8	9	8	8	8	9
container reuse rate	4.7%	4.6%	3.3%	4.9%	4.7%	4.3%
avg load per arc	2.55	2.20	2.82	2.68	3.08	2.82
avg stacking utilisation	0.12	0.10	0.13	0.13	0.15	0.14
avg travel time per container	50.28	52.24	48.79	50.80	49.45	50.74
orders per hour	1.19	1.15	1.23	1.18	1.21	1.18
runtime (min)	6.71	4.99	9.67	6.70	6.94	5.38

Table E.7: KPI comparison across configurations for Case 3

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 5	-	run 3	-	run 4	-
iterations	1250	725	993	782	1250	799
total operational time	4743	4872	4318	4541	4333	4484
vehicles used	11	11	10	10	9	10
container reuse rate	6.8%	4.9%	4.3%	5.4%	2.2%	4.7%
avg load per arc	2.11	2.09	2.81	2.54	2.55	2.50
avg stacking utilisation	0.10	0.10	0.13	0.12	0.13	0.12
avg travel time per container	62.41	64.11	56.81	59.75	57.01	59.00
orders per hour	0.81	0.79	0.89	0.85	0.89	0.86
runtime (min)	11.31	6.53	7.94	6.37	10.33	6.57

Table E.8: KPI comparison across configurations for Case 4

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 3	-	run 7	-	run 6	-
iterations	608	891	1236	1003	736	995
total operational time	3276	3313	3168	3241	3152	3222
vehicles used	7	7	7	7	7	7
container reuse rate	5.6%	5.7%	5.1%	6.2%	7.2%	7.1%
avg load per arc	2.47	2.41	2.91	2.67	2.51	2.62
avg stacking utilisation	0.12	0.11	0.14	0.13	0.13	0.13
avg travel time per container	56.49	57.12	54.62	55.88	54.35	55.55
orders per hour	1.04	1.03	1.08	1.06	1.09	1.06
runtime (min)	3.52	5.07	6.89	5.21	3.86	5.27

Table E.9: KPI comparison across configurations for Case 5

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 3	-	run 8	-	run 9	-
iterations	1250	1070	958	909	1250	1073
total operational time	2392	2455	2348	2399	2361	2445
vehicles used	5	5	5	5	5	6
container reuse rate	5.2%	5.0%	1.7%	3.2%	1.7%	2.6%
avg load per arc	3.24	2.51	3.23	3.08	2.91	2.81
avg stacking utilisation	0.15	0.12	0.15	0.15	0.15	0.14
avg travel time per container	55.63	57.09	54.60	55.79	54.90	56.87
orders per hour	0.95	0.93	0.97	0.95	0.97	0.93
runtime (min)	1.92	1.59	1.44	1.30	1.80	1.53

Table E.10: KPI comparison across configurations for Case 6

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 3	-	run 2	-	run 1	-
iterations	477	863	498	816	818	812
total operational time	6479	6610	5874	5944	5972	6067
vehicles used	15	15	14	13	13	14
container reuse rate	1.8%	3.1%	4.7%	3.4%	4.2%	4.0%
avg load per arc	2.42	2.36	2.73	2.86	2.70	2.65
avg stacking utilisation	0.12	0.11	0.13	0.14	0.13	0.13
avg travel time per container	59.44	60.64	53.89	54.53	54.78	55.66
orders per hour	0.84	0.83	0.93	0.92	0.91	0.90
runtime (min)	6.32	12.30	5.09	8.26	8.95	8.78

Table E.11: KPI comparison across configurations for Case 7

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 2	-	run 5	-	run 9	-
iterations	1250	1125	1129	1229	1138	1143
total operational time	1373	1384	1247	1263	1270	1293
vehicles used	3	3	3	3	3	3
container reuse rate	0.0%	0.0%	0.0%	0.4%	0.0%	0.0%
avg load per arc	2.29	2.39	4.61	4.01	3.15	2.97
avg stacking utilisation	0.11	0.11	0.22	0.19	0.16	0.15
avg travel time per container	44.28	44.65	40.24	40.75	40.98	41.70
orders per hour	1.22	1.21	1.35	1.33	1.32	1.30
runtime (min)	0.63	0.57	0.54	0.55	0.43	0.41

Table E.12: KPI comparison across configurations for Case 8

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 5	-	run 4	-	run 4	-
iterations	1176	1099	1043	1057	1030	917
total operational time	1484	1498	1348	1380	1367	1396
vehicles used	4	4	4	4	4	4
container reuse rate	5.0%	4.5%	2.9%	3.2%	2.9%	2.3%
avg load per arc	2.78	2.99	4.44	3.56	3.73	3.43
avg stacking utilisation	0.13	0.14	0.21	0.17	0.19	0.17
avg travel time per container	47.88	48.33	43.50	44.52	44.10	45.03
orders per hour	1.17	1.16	1.29	1.26	1.27	1.25
runtime (min)	1.51	1.40	0.92	0.89	0.84	0.68

Extra Test Cases

Table E.13: KPI comparison across configurations for the test case Case 3

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 1	-	run 1	-	run 3	-
iterations	1089	887	354	953	1250	953
total operational time	4731	4811	4265	4433	4348	4450
vehicles used	11	11	11	10	9	10
container reuse rate	7.6%	12.2%	12.6%	11.9%	12.2%	12.0%
avg vehicle load per arc	2.04	2.06	2.74	2.56	2.59	2.36
avg stacking utilisation	0.10	0.10	0.13	0.12	0.13	0.12
avg travel time per container	62.25	63.31	56.12	58.33	57.20	58.56
orders per hour	0.81	0.80	0.90	0.87	0.88	0.86
runtime (min)	10.03	8.62	3.95	10.26	13.23	9.45

Table E.14: KPI comparison across configurations for the test case Case 4

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 4	-	run 4	-	run 5	-
iterations	669	1023	646	948	692	807
total operational time	3157	3207	3078	3141	3039	3125
vehicles used	8	7	7	7	7	7
container reuse rate	14.4%	16.5%	16.9%	17.3%	16.4%	16.0%
avg vehicle load per arc	2.29	2.33	2.86	2.70	3.22	2.72
avg stacking utilisation	0.11	0.11	0.14	0.13	0.16	0.14
avg travel time per container	54.43	55.30	53.06	54.16	52.40	53.88
orders per hour	1.08	1.07	1.11	1.09	1.13	1.09
runtime (min)	4.05	6.17	3.60	5.21	4.00	4.49

Sensitivity Analysis

Table E.15: KPI comparison across configurations for Case 5 ($0.75\times$ service times)

KPI ↓ \ **config** →	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 3	-	run 3	-	run 1	-
iterations	850	1046	705	987	1232	1032
total operational time	2088	2164	2030	2114	2031	2099
vehicles used	5	5	4	5	4	5
container reuse rate	1.7%	3.7%	7.8%	4.2%	6.1%	4.7%
avg load per arc	2.54	2.52	3.17	3.20	3.59	3.19
avg stacking utilisation	0.12	0.12	0.15	0.15	0.18	0.16
avg travel time per container	48.56	50.33	47.22	49.17	47.24	48.81
orders per hour	1.09	1.05	1.12	1.08	1.12	1.09
runtime (min)	1.42	1.79	1.16	1.63	1.99	1.62

Table E.16: KPI comparison across configurations for Case 5 ($1.25\times$ service times)

KPI $\downarrow$ \ **config** $\rightarrow$	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 4	-	run 10	-	run 5	-
iterations	1250	1138	1250	1070	1250	1181
total operational time	2690	2782	2676	2767	2621	2719
vehicles used	6	6	6	6	6	6
container reuse rate	6.1%	5.4%	8.7%	5.8%	4.3%	4.8%
avg load per arc	2.81	2.28	2.60	2.57	2.46	2.60
avg stacking utilisation	0.13	0.11	0.12	0.12	0.12	0.13
avg travel time per container	62.57	64.70	62.24	64.36	60.95	63.24
orders per hour	0.85	0.82	0.85	0.82	0.87	0.84
runtime (min)	1.93	1.69	1.82	1.49	1.84	1.70

Table E.17: KPI comparison across configurations for Case 8 ($0.75\times$ service times)

KPI $\downarrow$ \ **config** $\rightarrow$	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 8	-	run 2	-	run 7	-
iterations	869	872	1250	1014	1250	1038
total operational time	1264	1297	1118	1155	1149	1162
vehicles used	3	3	3	3	3	3
container reuse rate	2.5%	3.5%	2.9%	2.6%	2.9%	2.9%
avg load per arc	3.52	3.18	3.89	4.56	5.09	4.42
avg stacking utilisation	0.17	0.15	0.19	0.22	0.25	0.22
avg travel time per container	40.77	41.84	36.08	37.25	37.06	37.49
orders per hour	1.38	1.34	1.56	1.51	1.51	1.50
runtime (min)	1.19	1.19	1.37	1.04	1.20	0.96

Table E.18: KPI comparison across configurations for Case 8 ($1.25\times$ service times)

KPI $\downarrow$ \ **config** $\rightarrow$	3×7 **best**	3×7 **avg**	7×3 **best**	7×3 **avg**	5×4 **best**	5×4 **avg**
run number	run 4	-	run 8	-	run 2	-
iterations	534	1051	1018	1065	687	922
total operational time	1686	1701	1584	1598	1569	1603
vehicles used	5	4	4	4	4	4
container reuse rate	5.0%	4.2%	0.0%	2.3%	2.9%	2.9%
avg load per arc	2.65	2.77	3.77	3.23	3.82	3.14
avg stacking utilisation	0.13	0.13	0.18	0.15	0.19	0.16
avg travel time per container	54.37	54.87	51.08	51.54	50.60	51.73
orders per hour	1.03	1.02	1.10	1.09	1.11	1.09
runtime (min)	0.55	1.10	0.76	0.79	0.52	0.66

F

Figures

Figure F.1: Pickup order type (Hauge et al., 2014).

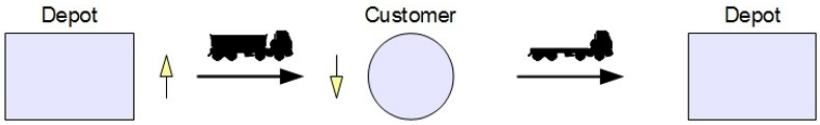


Figure F.2: Delivery order type (Hauge et al., 2014).

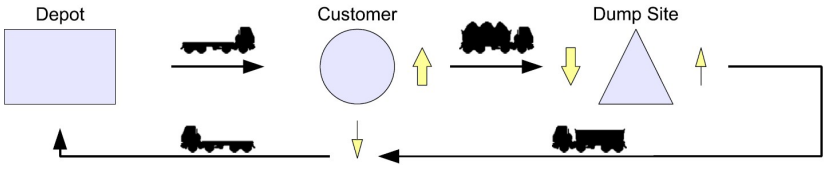


Figure F.3: Change order type (Hauge et al., 2014).

Figure F.4: Swap order type (Hauge et al., 2014).

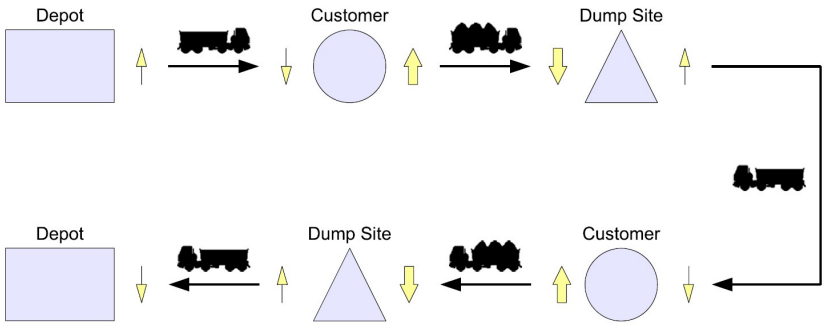


Figure F.5: Reuse applied between orders (Hauge et al., 2014).

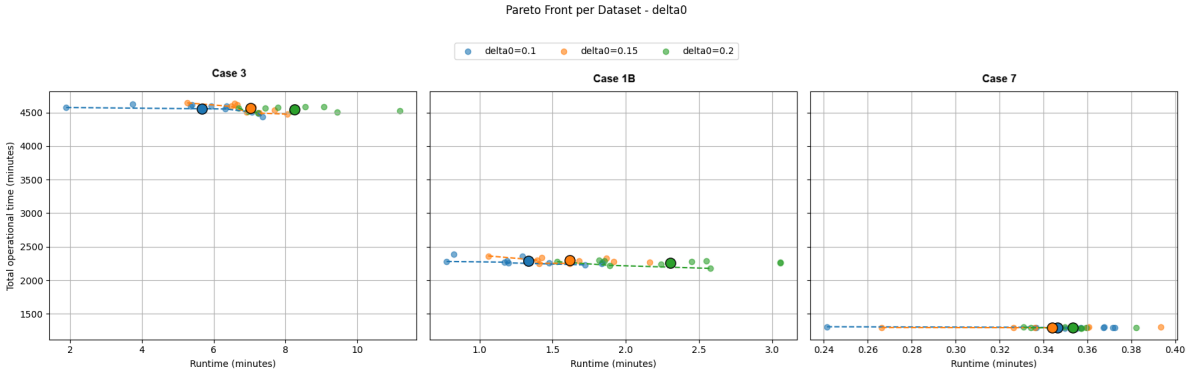


Figure F.6: Pareto front for parameter tuning of Delta0.

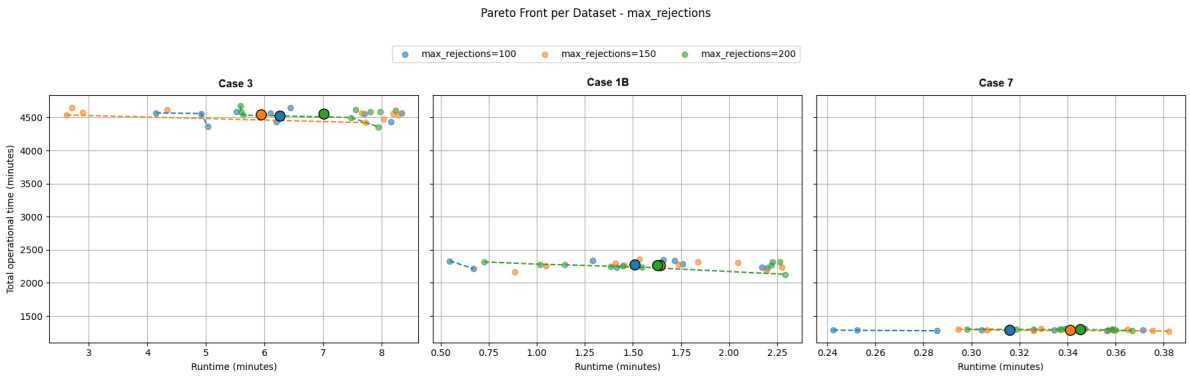


Figure F.7: Pareto front for parameter tuning of maximum rejections.

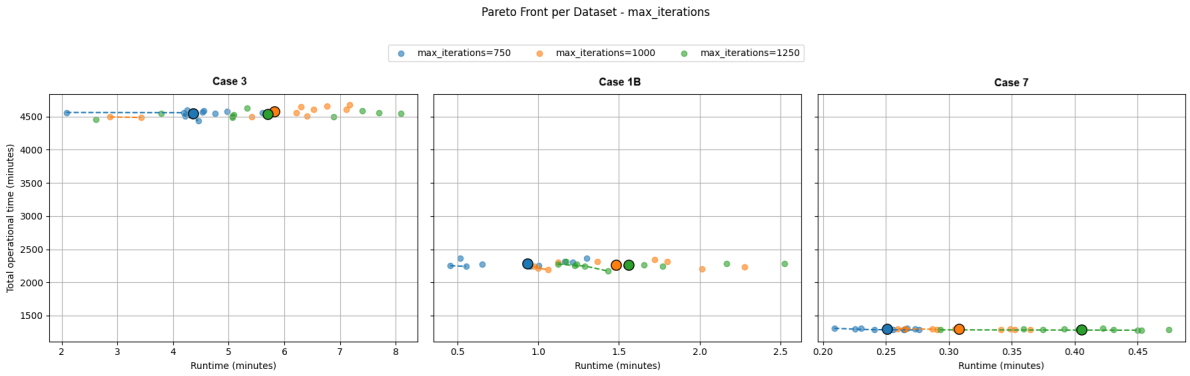


Figure F.8: Pareto front for parameter tuning of maximum iterations.

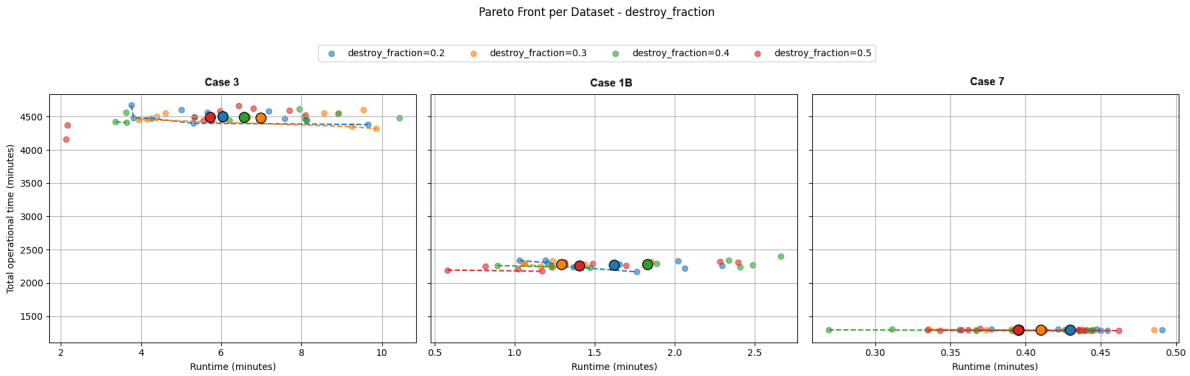


Figure F.9: Pareto front for parameter tuning of the destroy fraction.