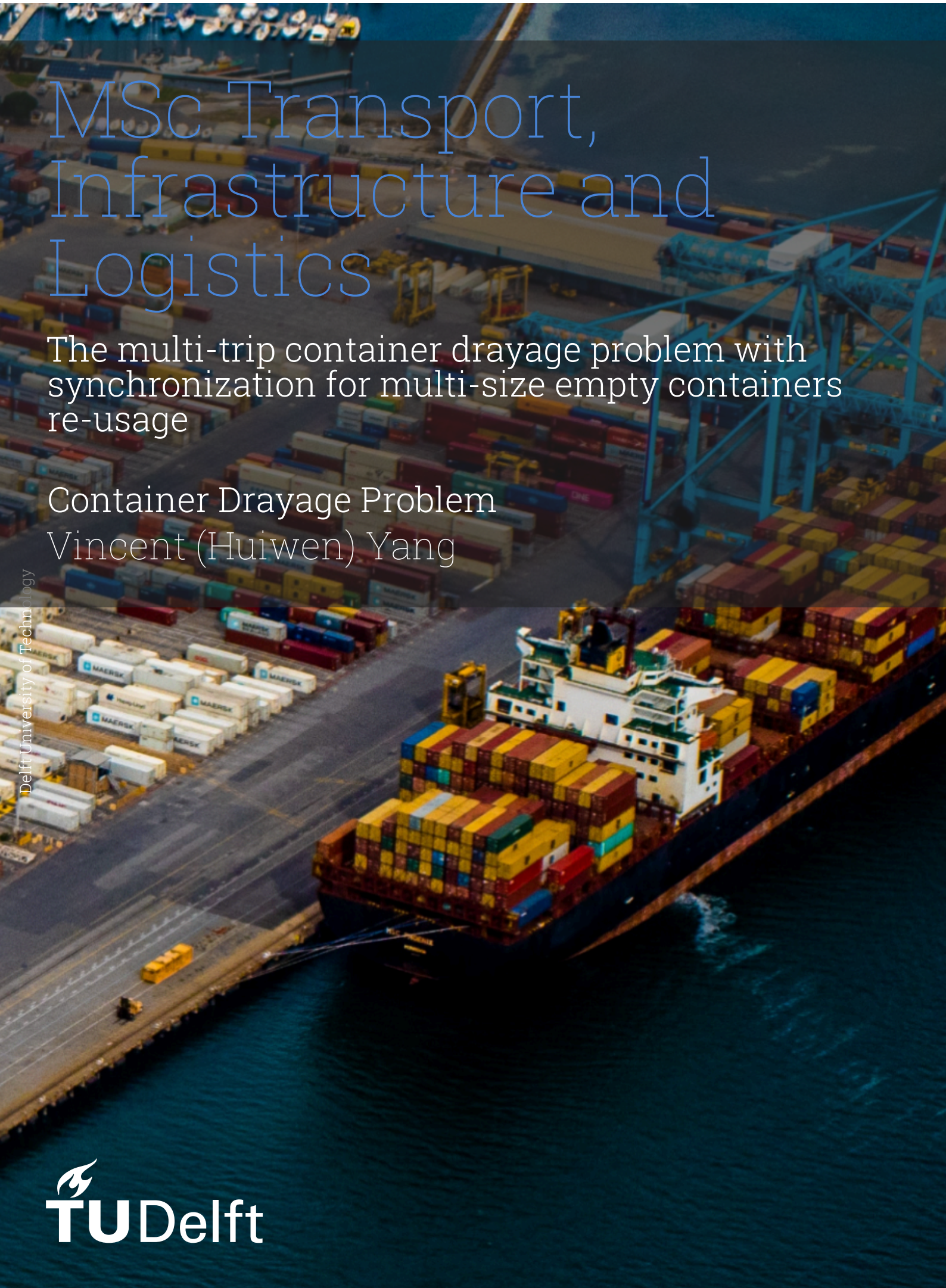

MSc Transport, Infrastructure and Logistics

The multi-trip container drayage problem with synchronization for multi-size empty containers re-usage

Container Drayage Problem

Vincent (Huiwen) Yang

Delft University of Technology

TU Delft

MSc Transport, Infrastructure and Logistics

The multi-trip container drayage problem with synchronization for multi-size empty containers re-usage

by

Vincent (Huiwen) Yang

Student Name	Student Number
Huiwen Yang	5703646

Chairman:	Lori Tavasszy
Supervisor:	Stefano Fazi, Alessandro Bombelli
Project Duration:	October, 2023 - June, 2024
Faculty:	Faculty of Civil Engineering and Geosciences, Delft

TU Delft

Abstract

This thesis investigates a container drayage problem involving terminals, depots, and shippers. Each terminal operates a homogeneous fleet of trucks that start and end at their respective terminals, carrying either one 40ft or two 20ft containers, and can make multiple trips within a single planning horizon. Terminals handle full containers and maintain a limited stock of empty containers, while depots provide additional empty containers. Shippers have specific time windows for service, and requests include sending and receiving empty or full containers to designated terminals, with street turns of empty containers possible. An Adaptive Large Neighborhood Search (ALNS) algorithm is implemented to address this problem, outperforming traditional methods like CPLEX. Extensive computational experiments validate the ALNS algorithm's efficacy, highlighting the logistical benefits of optimal depot placement and the impact of street turns, emphasizing the practical implications for container logistics.

Acknowledgements

First and foremost, I wish to express my deepest gratitude to my parents Zhiwei Yang and Yu Li, whose unwavering love and support have afforded me the privilege of a quality education. Their greatness and selflessness have been a constant source of inspiration in my life. I am profoundly grateful to Professor Lori Tavasszy for his invaluable guidance in selecting my research topic, and for elevating the conclusions of this thesis with his insightful input. Special thanks are due to Stefano Fazi and Alessandro Bombelli for their meticulous assistance and constructive feedback throughout the development of this thesis. I am also indebted to my friends, whose companionship has been a solace to me, even while being over 10,000 kilometers away from home. I extend my heartfelt appreciation to my girlfriend, Meilin Chen, for her unwavering support and encouragement throughout this journey.

Lastly, I wish that anyone reading these words finds the strength to endure solitude in times of adversity, the humility to remain grounded in times of success, and the courage to start anew when facing insurmountable challenges.

Contents

1

Introduction

The burgeoning growth of global trade has led to an unprecedented surge in containerized freight transport, predominantly facilitated by trucks in inland regions. While trucks offer unparalleled flexibility and speed in moving containers from ports to inland terminals, they also contribute to escalating environmental concerns, such as greenhouse gas emissions and air pollutants (Lee et al., 2019). Besides, Tompkins, 2022, Chief Operating Officer at Port Technology Services, argues that street turns are probably the single most impactful thing we can do to improve these industry-wide challenges. However, quantifying the potential benefits of street turns remains difficult. Existing models for optimizing truck routes in inland container transport often focus on few-constraints, such as time window and single depot, without adequately addressing the environmental externalities, such as limited empty container and multi-trip. Thus, there exists a compelling need for a more holistic optimization model that not only evaluates and enhances operational efficiency but also mitigates environmental impact.

The objective of this research thesis is to minimize the total routing cost in multi-trip container drayage operations within the landscape of container transport logistics. Upon successfully achieving this aim, we anticipate consequential results including enhanced operational efficiency, reduced environmental pollution, and a model that more closely approximates real-world conditions. To realize this objective, we introduce a novel optimization framework that cohesively integrates multi-terminal operations, resource synchronization, and multi-type container reusage. This comprehensive model not only aims to solve the primary problem of routing cost minimization but also provides a robust tool for making more informed and sustainable logistical decisions. To the best of our knowledge, this is the first study that amalgamates these diverse considerations into a singular optimization framework, thereby filling an existing gap in the academic literature. Fazi et al., 2023 addressed the characteristics of resource synchronization and multi-trip in the container drayage problem. Shiri and Huynh, 2016 and Nossack and Pesch, 2013 tackled the multi-port characteristics. Chen et al., 2021, on the other hand, incorporated the multi-type container feature into the drayage problem.

In this thesis, we address a complex container drayage problem involving multiple trips and container types within a multi-terminal network. The focus is on a network of inland terminals that manage a fleet of trucks and a stockpile of empty containers to serve a set of shippers or consignees located in the hinterlands. Containers in this network have various origins and destinations: full containers may originate from either inland terminals or shippers and are destined for consignees or back to inland terminals. Empty containers, on the other hand, can originate from empty depots, inland terminals, or consignees where they have been street-turned. Their destinations include either inland terminals, for replenishing stock for subsequent deliveries, or empty depots. The fleet of trucks is homogeneous in its capabilities, designed to accommodate either a single 40-foot container or two 20-foot containers. Within the confines of a given planning horizon, each truck is capable of executing multiple trips, provided that time constraints permit. A single trip is defined as a truck starts from an inland terminal and returns to an inland terminal after completing the task. Considering real-world trucking operations, drivers typically commence and conclude their work shifts at the same terminal. Therefore, it is stipulated that the terminal to which a truck returns must be the same as the one from which it initially

departed.

In addressing the complex container drayage problem, our study introduces an innovative approach through the implementation of the Adaptive Large Neighborhood Search (ALNS) algorithm, which is detailed in our methodology. Our implementation of ALNS is uniquely characterized by its use of a variety of destroy and repair operators, which significantly modify the solution space to promote thorough exploration and effective exploitation. The algorithm also integrates a simulated annealing mechanism as its acceptance criterion, which dynamically adapts based on performance evaluations of different neighborhood structures. This adaptability allows ALNS to effectively refine solutions through iterative enhancements, significantly improving the efficiency of solving the container drayage problem.

During our tests, computational experiments were conducted on small- and medium-scale instances using both CPLEX and the ALNS algorithm for cross-validation. Additionally, we developed real-world scenarios based on the geographical locations of terminals, creating instances with 50 and 100 points. These instances were then utilized to conduct detailed analyses on the impact of initial empty container stock, depot location, and empty container street-turns. Such studies provide insights for logistics planning and routing network configuration, aiding in the strategic management and planning within the field of transportation logistics.

The structure of this thesis is as follows. Firstly, Section 2 reviews previous studies and explicitly identifies the gap between this model and prior research. Section 3 introduces the model's setup and mathematical formulation. Section 4 verifies the model using an example solved through CPLEX. Section 5 describes the Adaptive Large Neighborhood Search (ALNS) algorithm developed to tackle medium to large-scale instances. Section 6 presents the computational results. Finally, Section 7 serves as a concluding remark reviewing the entire work.

2

Literature Review

2.1. Basic Load Drayage Problem

The main research topics about this thesis are the variants based on container drayage problem by considering multiple terminals/depots, multiple container types, resource synchronization and multi-trip. Most previous studies only focused on one or several of the settings or constraints. This thesis intends to combine all the above settings into a general model so that it can deal with more complex real-life situations. The literature review will divide into several parts based on above settings or constraints.

The Drayage Problem is a multifaceted logistical challenge in the field of transportation and supply chain management. At short-haulage container transportation's core, it revolves around efficiently moving containers from one shipper to another by trucks in the terminal region (Sinclair and Dyk, 1987). This problem can be broadly categorized into two main types: the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP). In the TSP variant, the focus is on optimizing the distribution of goods by a single vehicle, connecting multiple points in the most efficient way possible and always going back to the origin when the trip is over. This simplification is useful when dealing with scenarios where only one vehicle is available for transportation, and the primary goal is to minimize the total distance traveled or time spent. On the other hand, the VRP variant encompasses a more intricate set of challenges. In addition to finding optimal routes for multiple vehicles, it involves considerations such as vehicle allocation, capacity constraints, and the distribution of goods among multiple points. This model is particularly relevant in scenarios where a fleet of vehicles needs to be allocated efficiently to pick up and deliver goods while adhering to constraints such as vehicle capacities, time windows for deliveries, and the availability of resources.

The Container Drayage Problem is developed based on the foundations of the TSP and the VRP. Researchers and practitioners further differentiate the Drayage Problem based on several factors. Time window constraints play a crucial role, as some scenarios require goods to be picked up or delivered within specific time intervals. Resource constraints, such as limitations on vehicle capacity or other resources, add complexity to the problem. Depot or terminal considerations also come into play, with some models focusing on a single starting and ending point for transportation, while others deal with more complex scenarios involving multiple depots or terminals. Additionally, variations of the problem may consider deterministic or stochastic data.

Based on the existing research in port operations and management, primarily, there are two truck operating modes considered: the trailer stay-with mode and the trailer separation mode. The trailer stay-with mode involves a truck remaining with its trailer during the loading and unloading of containers, ensuring continuity and simplicity in operations. Conversely, the trailer separation mode offers a choice for the truck to either wait with its trailer at a customer's premise or leave without it after transporting a container (Chen et al., 2021). This thesis focuses on utilizing the trailer stay-with mode.

The optimization of the Container Drayage Problem has garnered growing attention over the past twenty years. This section will introduce how our model originated. Wang and Regan, 2002 developed a

methodology for the multiple traveling salesman problem with time window constraints (m-TSPTW) aimed at local pick-up and delivery of goods. They selected the best solution from three possibilities: considering time window constraints in route construction, relaxing time window constraints, and using discretized time windows. They discovered that smaller discretization intervals tended to yield better computational results, albeit at the cost of increased time. This model represents a basic framework for the pick-up and delivery of goods, where there is no differentiation between goods, the endpoints for vehicles are not determined, and it does not meet all customer requirements. Tjokroamidjojo et al., 2004 extends this model and takes into account the effect of uncertainty in loading, unloading, waiting, and travel times on truck/driver-to-load assignment decisions. Trucks are scattered in different cities. And the initial assembly plan is a priori, and necessary constraints are added within each decision window. The study offers useful insights and assesses several methods for calculating the value of advance load information. To account for data variability, the model applies stochastic programming approaches, most specifically chance constraint programming. However, they did not use real geographic data.Pi et al., 2006 proposes a TSP to solve the local pickup and delivery based on the nested partitioning (NP) method, a metaheuristic approach for combinatorial optimization problems. In order to improve the job satisfaction of truck drivers, each truck is returned to the initial location every day, and the truck driver's preference index is set. NP methods involve partitioning the problem space into nested subsets and iteratively searching for optimal solutions within these subsets. Despite the commendable efforts in this paper to present a comprehensive solution framework, a notable shortcoming is the absence of a more nuanced approach that incorporates both Lagrangian Relaxation methods and specialized local search algorithms. The NP algorithm outperforms the CPLEX solver in terms of solution quality. Our model was originally derived from this type of problem.

2.1.1. Container Drayage Problem with Single Terminal/Depot

This section will describe how such a model takes Single Terminal/Depot into consideration. When considering VRP or TSP in connection with single terminal/depot container pickup and delivery, Imai et al., 2007 tackled the Vehicle Routing Problem with Full Container Load (VRPFC), which focuses on the distribution of container cargo from a single terminal on a given day. Each truck has a maximum operating time limit and can operate multiple trips, but there is no time window constraint for delivering homogeneous container. The problem is considered NP-hard, and the paper proposes a Lagrangian relaxation-based heuristic to solve it. A significant limitation of the current paper lies in the oversimplification of the delivery scenario under consideration. Specifically, the model fails to account for the complexities introduced by time window constraints and resource limitations, which are critical factors in real-world applications.

Caris and Janssens, 2007 and Caris and Janssens, 2010 extend the full truckload pickup and delivery problem with time windows. The former proposes a two-phase insertion heuristic to solve it. It focuses on developing a construction heuristic and an improvement heuristic to find an initial solution and then improve it through local search. On the other hand, the latter proposes a deterministic annealing algorithm to solve the pre- and end-haulage problem. Deterministic annealing is a metaheuristic optimization technique inspired by the annealing process in metallurgy. It uses a temperature parameter to control the exploration and exploitation of the search space. However, in both papers, trucks can only be assigned to single routes. Braekers et al., 2013 extend the work by adding the allocation problem to determine the optimal repositioning of empty containers. Each truck can operate multiple trips in a single day with single depot. There is no mention of a specific limit on the number of empty containers. The paper presents a sequential and an integrated approach to solving this problem, formulating it as an asymmetric multiple vehicles Travelling Salesman Problem with Time Windows (am-TSPTW). The model setting demonstrates a lack of complexity in its assumptions concerning terminal resources and vehicle capabilities. Firstly, the assumption of unlimited empty containers available at each terminal stands in stark contrast to real-world conditions, where resource constraints often play a pivotal role. Secondly, the simplification that vehicles and containers are homogeneous, coupled with the stipulation that a vehicle can only hold one container per trip, seriously curtails the model's ability to capture the intricate dynamics of heterogeneous fleets and multi-container loading scenarios.

Fazi et al., 2023's model includes a terminal, a depot and many shippers (consignee), and considers a VRP model with multiple trips, limited fleet, time window restrictions, and 40ft empty container resource constraints. They develop a column-and-row generation algorithm embedded in a branch-and-price

framework to accurately solve this problem.

2.1.2. Container Drayage Problem with Multiple Container Type

Regarding the Container Drayage problem, papers on distinguishing types of containers have gradually emerged in the past decade. This improvement, in comparison to Fazi et al., 2023's work, also involves the integration of the concept of Multiple Terminal/Depots into our model. Based on ISO, 2020, containers are divided into 10, 20, 30, 40 and 45 feet, but currently most of the containers used in container transportation are 20 feet or 40 feet in length, with a width of around 8 feet and a height of approximately 8.5 feet.

Vidović et al., 2011 established a single terminal container drayage problem. The peculiarity of this problem is that a vehicle can transport one 40-foot container or two 20-foot containers at the same time without time window constraint. Each truck can be assigned to single route, which fulfills at most 4 requests with two deliveries and two pickups. There is only one terminal which is the origin and destination for trucks. They calculated utilities for all potential node matches. The utilities are based on the length of a single route that visits all nodes and the sum of all routes when nodes are visited separately. A 20-foot container has less weight. Then a heuristic algorithm is proposed to solve this problem. While the paper makes noteworthy contributions to the field, it does not sufficiently address the necessity of algorithmic improvements that account for real-life constraints, such as time windows and non-homogeneous fleets of vehicles. Vidović et al., 2017 extended the work by adding the time window constraint. They propose a variable neighborhood search (VNS) heuristic to solve larger instances of the problem. Chen et al., 2021 extended the work to a more general level for single terminal and single route for trucks. They propose a mixed-integer linear programming (MILP) model that can fit any size containers. They don't see trucks and containers as a scarce resource. They create two relaxed MILP models that take less time to compute and deliver answers that are nearly optimal. A hybrid heuristic approach that combines the variable neighborhood search scheme and the cheapest viable insertion mechanism is also created to address complex issues. However, a salient limitation of the paper is its neglect to correlate transportation costs with the size of the containers being moved. Compared to the model proposed by Fazi et al., 2023, we have added 40ft containers, making it no longer limited to only 20ft containers.

2.1.3. Container Drayage Problem with Synchronized Resource Constraints

Our model falls under the category of resource synchronization in the vehicle routing problem with synchronized constraints. Drexl, 2012 gave a clear definition of vehicle routing problem with synchronized constraints that more than one vehicle need to fulfill a task in terms of spatial, temporal and load aspects. They gave five types of synchronization, which are task synchronization, operation synchronization, movement synchronization, load synchronization and resource synchronization. Our model falls in the resource synchronization that trucks compete for scarce empty container in the inland terminal, because retrieving empty containers from further depots potentially incurs penalties due to extra mileage.

Paraskevopoulos et al., 2017 refines this definition by considering whether the resource is renewable. In our model, the empty container resource is renewable, but there is a certain loss, because some needs will first obtain an empty container and then export it as a full container. Zhang et al., 2011 based on the container drayage problem with time window adds the a limited number of empty containers as a constraint and each truck can be assigned a single route to delivery one 40 feet container. The movement of trucks will affect the number of hollow containers in the depot. The network has one terminal and one depot that is the origin and destination for a route. They developed an algorithm based on reactive tabu search to solve randomly generated examples. However, a limitation of this paper is that it only considers homogeneous containers and trucks, and it is a single depot/terminal. Zhang et al., 2020 extend the work by adding one type task of container, which is outbound empty container task. The rest of three tasks are inbound full container tasks, outbound full container tasks and inbound empty container tasks. In reality, only import-dominant area can have the outbound empty container tasks and only export dominant area can have the inbound empty container tasks. However, the number of empty containers in the depot at the current moment is determined by the movement of the past moment. It has no other source of empty container supplementation.

2.1.4. Container Drayage Problem with multi-trip problem

Taillard et al., 1996 first proposed the concept of multi-trip idea, which means vehicle are allowed to do multiple routes to and from the origin/destination during one working horizon. The first exact algorithm is proposed by Mingozzi et al., 2012 to solve the multi-trip vehicle routing problem, which is a very typical model. They present two set-partitioning-like formulations of the problem and study valid lower bounds based on these formulations' linear relaxations. But they did not consider the time window constraint.

In terms of container drayage problem, Bruglieri et al., 2021 propose a more realistic alternative called the Multi-trip Multi-period CDP with Release and Due Dates, which divides the planning horizon into distinct periods and allows trucks to undertake several trips within each period. The issue also takes into account customer service release and due dates, as well as contractual constraints on drivers' working hours. To model the problem, the authors present an Arc-based Integer Linear Programming formulation. However, a notable limitation is the neglect of time window constraints in the modeling of trucks engaged in multi-trip scenarios. Imai et al., 2007, Caris and Janssens, 2007, Shiri and Huynh, 2016, and Fazi et al., 2023 also incorporated the characteristic of trucks being able to perform multi-trip in their models.

2.2. Adaptive Large Neighborhood Search

This thesis introduces a novel mathematical model devised to tackle a variant of container drayage problem, a conundrum recognized for its NP-hard nature. Traditionally, computational solvers such as CPLEX have been employed to navigate through the complexity of smaller instances of this problem. However, these approaches often stumble when scaling up, as larger instances not only demand prohibitive computational resources but also suffer from extensive solution times, rendering them impractical for real-world applications where time is of the essence.

In light of these constraints, our investigation pivots towards the realm of metaheuristic algorithms, known for their adeptness at handling complex, large-scale optimization problems with a balance of precision and expedience. Among these, the Adaptive Large Neighborhood Search (ALNS) algorithm stands out for its exceptional capability to explore and exploit the vast search space inherent to the container drayage problem. The genesis of ALNS from its predecessor, LNS, introduced by Shaw, 1998 in the late 1990s. Unlike traditional local search algorithms that explore the solution space by making incremental changes to a current solution, LNS explores significantly larger neighborhoods by allowing larger, more disruptive changes. This is achieved by selectively removing and then reinserting a subset of the solution components, thereby potentially escaping local optima and exploring the solution space more effectively.

Building on the foundational principles of LNS, Ropke and Pisinger, 2006 introduced Adaptive Large Neighborhood Search in 2006, aiming to solve Pickup and Delivery Problems with Time Windows (PDPTW). An essential aspect of the Adaptive Large Neighborhood Search methodology is the necessity for an initial solution to commence the search process. Although this initial solution does not need to be close to the optimum, it is imperative as it establishes a starting point from which the algorithm can iteratively refine towards improved solutions. ALNS introduces an adaptive mechanism that dynamically adjusts the search strategy based on the performance of different neighborhood structures. This adaptation is key to the algorithm's success, as it allows ALNS to intelligently navigate through the search space by prioritizing the most promising neighborhood operators based on their recent performance. The selection of these operators is typically governed by mechanisms such as score-based or probability-based approaches, enabling the algorithm to balance between exploration of the solution space and exploitation of the best-found solutions.

In recent years, a multitude of scholars have employed the ALNS algorithm to tackle various NP-hard problems, demonstrating its versatility and effectiveness. For instance, Hiermann et al., 2016 leveraged ALNS to address the Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Charging Stations, showcasing how it can be adapted to manage the intricacies of electric vehicle logistics. Similarly, Keskin and Çatay, 2016 developed a model for the Electric Vehicle Routing Problem, incorporating a comprehensive range of constraints such as ensuring the connectivity of customers, visits to recharging stations, flow conservation, time feasibility of arcs, adherence to time windows for

customers and the depot, elimination of sub-tours, fulfillment of customer demands, tracking the battery state of charge, and managing the battery state after recharging. This model was also successfully solved using ALNS. Moreover, Sacramento et al., 2019 applied ALNS to the Unmanned Aerial Vehicle Routing Problem, considering the cost implications of various parameters for trucks and drones. These examples underline the suitability of ALNS for resolving complex logistical challenges, affirming its status as a highly adaptable tool in the field of operations research.

However, a review of the existing scholarly work reveals notable gaps that this study aims to fill. Specifically, most prior research has focused on isolated aspects of the larger problem, see table 2.1. To date, there is no comprehensive framework that successfully combines all these different elements, thereby creating a more adaptable and broadly applicable model. The main contribution of this paper, therefore, is to introduce a new, unified approach that integrates these various aspects into a single, more robust model.

Table 2.1: Features of the papers according to basic model settings

Paper	Frame				Container			Vehicle				Method	
	Time Window	Single Terminal	Multiple Terminal	Multiple Container Type	Resource Synchronization	Multi-trip	Heterogeneous Vehicles	Stay With	Separable	Exact Method	Heuristic
Wang and Regan, 2002	✓	✓						✓			✓
Tjokroamidjojo et al., 2004	✓	✓						✓			
Pi et al., 2006	✓	✓						✓		✓	✓
Imai et al., 2007		✓				✓		✓		✓	✓
Caris and Janssens, 2007	✓	✓				✓		✓			✓
Caris and Janssens, 2010	✓	✓						✓			✓
Braekers et al., 2013	✓		✓			✓			✓		✓
Braekers et al., 2014	✓		✓			✓			✓		✓
Sterzik and Kopfer, 2013	✓		✓					✓			✓
Nossack and Pesch, 2013	✓		✓					✓			✓
Shiri and Huynh, 2016	✓		✓			✓		✓			✓
Vidović et al., 2011		✓		✓				✓			✓
Vidović et al., 2017	✓	✓		✓			✓	✓			✓
Chen et al., 2021	✓	✓		✓			✓	✓			✓
Zhang et al., 2011	✓	✓			✓			✓			✓
Zhang et al., 2020	✓	✓			✓			✓			✓
Mingozzi et al., 2012		✓				✓		✓		✓	
Bruglieri et al., 2021		✓				✓		✓		✓	
Lai et al., 2013		✓					✓	✓			✓
Fazi et al., 2023	✓	✓			✓	✓		✓		✓	
Our model	✓		✓	✓	✓	✓		✓			✓

3

Problem formulation

3.1. Model setting

Figure 3.1 presents a simplified example illustrating the flow of containers. For simplicity, the containers in the diagram are not differentiated by size. In our model, the size of a container only affects the loading capacity of a truck. A truck can carry either one 40-foot container, regardless of whether it's empty or full, or two 20-foot containers, again irrespective of their state. The truck initiates its journey from its associated terminal and eventually returns to the same terminal. Each shipper's request must be fulfilled. Shipper i requires an empty container, which can be acquired from a terminal, an empty depot, or another shipper j (a process known as street turning), provided that the terminal has empty containers in stock or shipper j happens to need to dispatch an empty container. Shipper i may also send a full container to the terminal 0. Shipper j, on the other hand, needs to obtain a full container from the terminal 1 and must dispatch an empty container. The destination of this empty container could be a terminal, an empty depot, or shipper i. The return of empty containers to the terminal helps replenish its inventory. Each full container must be obtained from a specific terminal, and every full container is assigned a unique and distinct index. In contrast, empty containers are homogeneous, and a shipper's request for empty containers can be satisfied from any source. For trucks, each trip cannot revisit the same index more than once, and there is a maximum limit on the number of trips a truck can make within a planning horizon.

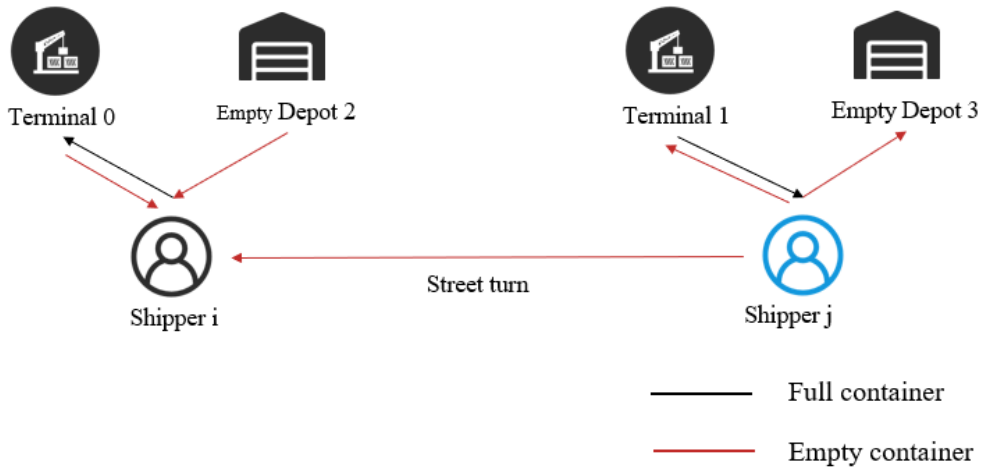

Figure 3.1: Possible flow of container

This model considers four types of containers: 40-foot empty containers, 40-foot full containers, 20-foot empty containers, and 20-foot full containers. For each shipper, theoretically, they have three choices

for each type of container: require, dispatch, or nothing. That means there are $3^4 = 81$ possible scenarios. However, due to the model's constraint that a shipper with the same ID can only be accessed by a truck once, and trucks have capacity limitations (i.e., they can carry either one 40-foot container or two 20-foot containers), it is not possible for the same shipper ID to simultaneously require one 40-foot container and any other type of container or to dispatch one 40-foot container and any other type of container, as this would exceed the maximum capacity of one truck. For full containers, a shipper has at most one request. In reality, every combination can be achieved by introducing a different shipper ID with the same coordinate. In addition, we assume that each shipper can demand a maximum of two types of containers.

We list all the 18 scenarios considered, shown as table 3.1. -1 means dispatching, 0 means no demand, 1 means requiring.

Table 3.1: Demand Scenarios of Shippers

Scenario	40-ft empty	20-ft empty	40-ft full	20-ft full
1	1	-1	0	0
2	-1	1	0	0
3	1	0	-1	0
4	-1	0	1	0
5	1	0	0	-1
6	-1	0	0	1
7	0	1	-1	0
8	0	-1	1	0
9	0	1	0	-1
10	0	-1	0	1
11	1	0	0	0
12	-1	0	0	0
13	0	1	0	0
14	0	-1	0	0
15	0	0	1	0
16	0	0	-1	0
17	0	0	0	1
18	0	0	0	-1

3.2. Mathematical formulation

We consider a network denoted as $G(N, A)$, where A represents the set of arcs, and N symbolizes the set of nodes. This set comprises the inland terminal T , the depots H for empty containers, along with a node belongs to set S designated for each shipper. Within N, the subset of shippers is marked as S. The set N is the universal set comprising terminal set T, depot set H, and shipper set S.

Next, we specify the set of trucks as K, comprising t subsets K_t and K_1, each associated respectively with terminal t. We establish a maximum limit of trips, denoted as R, along with its corresponding numerical series of trips from 0 to $|R|$. Finally, we designate P as the set of container types, encompassing four types of containers, specifically $[P_{E40}, P_{F40}, P_{E20}, P_{F20}]$. These represent the 40-foot empty container, the 40-foot full container, the 20-foot empty container, and the 20-foot full container, respectively.

In terms of parameters, for every shipper in the subset i within S, their time window is depicted by the interval $[A_i, B_i]$. At the inland terminal t, the initial availability of empty containers of type $[P_{E40}, P_{E20}]$ is indicated by D_t^p. For each shipper, the demand of container type $p \in P$ for shipper i is indicated by D_i^p, i.e., the element '1' indicates a requirement for one container, '-1' indicates the dispatch of one container, and '0' signifies no involvement with the container. The time distance between any two nodes i and j is indicated by C_{ij}.

For our decision variables, $x_{i,j}^{k,r}$ denotes the binary routing variable, which represents whether truck k uses the arc (i, j) during trip r. The flow variable $y_{i,j,p}^{k,r}$ (belonging to $0, 1, 2$) signifies the flow of container

type p from node i to node j on trip r by truck k. The time variable $t_i^{k,r}$ (also in R^+) indicates the arrival time at node i. The variables $t_{end}^{k,r}$ (in R^+) represent the end times for completing trip r of truck k at the terminal.

The sets, parameters and decision variables are shown in table 3.2

Table 3.2: Sets, Parameters and Decision Variables

Set	
N	Set of all nodes
A	Set of arcs
T	Set of inland terminal
H	Set of empty depot
S	Set of shippers
K	Set of trucks, where $K = \sum_{nt \in T} K_{nt}$
K_{nt}	Set of trucks belong to terminal nt
R	Set of trips
P	Set of container type
P_E	Set of empty container type, where P_E = $P_{E40} \cup P_{E20}$
P_F	Set of full container type, where P_F = $P_{F40} \cup P_{F20}$
P_{E40}	Set of empty 40-foot container
P_{E20}	Set of empty 20-foot container
P_{F40}	Set of full 40-foot container
P_{F20}	Set of full 20-foot container
Parameter	
D_i^p	Demand of node i of container type p, where $p \in [P_{E40}, \mathsf{P}_{20E}, P_{F40}, P_{F20}]$
F_p	Extra transportation time for container p
$[A_i, B_i]$	Time window of shipper i, where $i \in N$.
$C_{i,j}$	Transportation time between node i and j, where $i, j \in N$.
M	A large value
Decision variables	
$x_{i,j}^{k,r}$	Binary variable that equals 1 if truck k in trip r goes from node i to j, 0 otherwise
$y_{i,j,p}^{k,r}$	Flow variable $(0, 1, 2)$ represents the flow of container type p from node i to j in trip r by truck k
$t_i^{k,r}$	Arrival time (R^+) of truck k in trip r at node i
$t_{end}^{k,r}$	End time of trip r of truck k (R^+)

We formulate the problem as follows:

$$\text{Minimize} \quad \sum_{k \in K} \sum_{r \in R} \sum_{i,j \in N} C_{i,j} x_{i,j}^{k,r} + \sum_{k \in K} \sum_{r \in R} \sum_{p \in P} \sum_{i,j \in N} F_p y_{i,j,p}^{k,r} \tag{3.1}$$

Subject to:

$$\sum_{j\in N} x_{j,i}^{k,r} = \sum_{j\in N} x_{i,j}^{k,r} \qquad \forall k \in K, r \in R, i \in N \qquad (3.2)$$

$$\sum_{k\in K}\sum_{r\in R}\sum_{j\in N} x_{j,i}^{k,r} = 1 \qquad \forall i \in S \qquad (3.3)$$

$$\sum_{j\in N} x_{j,i}^{k,r} \leqslant 1 \qquad \forall k \in K, r \in R, i \in T \cup H \qquad (3.4)$$

$$t_i^{k,r} + C_{i,j} - M(1 - x_{i,j}^{k,r}) \leqslant t_j^{k,r} \qquad \forall k \in K_t, t \in T, r \in R, i, j \in N, j \neq t \qquad (3.5)$$

$$(A_i + C_{i,j}) * x_{i,j}^{k,r} \leqslant t_j^{k,r} \qquad \forall k \in K_t, t \in T, r \in R, i, j \in N, j \neq t \qquad (3.6)$$

$$t_i^{k,r} + C_{i,t} - M(1 - x_{i,t}^{k,r}) \leqslant t_{end}^{k,r} \qquad \forall k \in K_t, t \in T, r \in R, i \in N \qquad (3.7)$$

$$(A_i + C_{i,t}) * x_{i,t}^{k,r} \leqslant t_{end}^{k,r} \qquad \forall k \in K_t, t \in T, r \in R, i \in N \qquad (3.8)$$

$$A_i \sum_{j\in N} x_{i,j}^{k,r} \leqslant t_i^{k,r} \qquad \forall k \in K_t, t \in T, r \in R, i \in N\backslash\{t\} \qquad (3.9)$$

$$t_i^{k,r} \leqslant B_i \sum_{j\in N} x_{j,i}^{k,r} \qquad \forall k \in K, r \in R, i \in N \qquad (3.10)$$

$$t_t^{k,r} \leqslant t_i^{k,r} \qquad \forall k \in K_t, t \in T, r \in R, i \in N\backslash\{t\} \qquad (3.11)$$

$$M x_{i,j}^{k,r} \geqslant y_{i,j,p}^{k,r} \qquad \forall k \in K, r \in R, i, j \in N, p \in P \qquad (3.12)$$

$$\sum_{k\in K}\sum_{r\in R}\sum_{j\in N} y_{j,i,p}^{k,r} - \sum_{k\in K}\sum_{r\in R}\sum_{j\in N} y_{i,j,p}^{k,r} = D_i^p \qquad i \in S, p \in P \qquad (3.13)$$

$$\sum_{k\in K}\sum_{r\in R}\sum_{j\in N} y_{j,i,p}^{k,r} - \sum_{k\in K}\sum_{r\in R}\sum_{j\in N} y_{i,j,p}^{k,r} = D_i^p \qquad i \in T, p \in P_F \qquad (3.14)$$

$$\sum_{j\in N} y_{j,i,p}^{k,r} = \sum_{j\in N} y_{i,j,p}^{k,r} \qquad \forall k \in K, r \in R, i \in H, p \in P_F \qquad (3.15)$$

$$\sum_{p\in P_{E20}, P_{F20}} y_{i,j,p}^{k,r} + 2 * \sum_{p\in P_{E40}, P_{F40}} y_{i,j,p}^{k,r} \leqslant 2 \qquad \forall k \in K, r \in R, i, j \in N \qquad (3.16)$$

$$\sum_{k\in K}\sum_{r\in R}\sum_{j\in N} y_{i,j,p}^{k,r} - \sum_{k\in K}\sum_{r\in R}\sum_{j\in N} y_{j,i,p}^{k,r} \leqslant D_i^p \qquad i \in T, p \in P_E \qquad (3.17)$$

$$\sum_{j\in N\backslash\{t\}} x_{nt,j}^{k,r+1} \leqslant \sum_{j\in N\backslash\{t\}} x_{j,nt}^{k,r} \qquad \forall k \in K_{nt}, nt \in T, r \in 0 \ldots |R| - 1 \qquad (3.18)$$

$$\sum_{j\in N\backslash\{t\}} x_{nt,j}^{k+1,0} \leqslant \sum_{j\in N\backslash\{t\}} x_{j,nt}^{k,0} \qquad \forall k \in 0 \ldots |K_{nt}| - 1, nt \in T \qquad (3.19)$$

$$x_{i,j}^{k,r} \in \{0, 1\} \qquad \forall k \in K, r \in R, i, j \in N \qquad (3.20)$$

$$y_{i,j,p}^{k,r} \in \{0, 1, 2\} \qquad \forall k \in K, r \in R, p \in P, i, j \in N \qquad (3.21)$$

$$t_i^{k,r} \in R^+ \qquad \forall k \in K, r \in R, i \in N \qquad (3.22)$$

$$t_{end}^{k,r} \in R^+ \qquad \forall k \in K, r \in R \qquad (3.23)$$

The objective function, as defined in equation 3.1, aims to minimize the total routing cost. Constraint 3.2 serves as the truck flow conservation constraint. Constraint 3.3 ensures that all shippers are visited once. For every terminal and depot, they can be at most visited once in one trip, which are imposed by constraint 3.4. The arrival time constraints for two consecutive nodes are outlined in equation 3.5, specifically focusing on the time distance. Constraint 3.6 states that the arrival time at the next node must be greater than the earliest time window of the preceding node plus the time distance between them. Constraints 3.7 and 3.8 simply replace the arrival time of constraints 3.5 and 3.6 with the end time. As each trip involves visiting the starting point twice (at the beginning and at the end), we require additional end time and corresponding constraints to restrict the ending time. Constraints 3.9 and 3.10 restrict the arrival time at which a truck visits a particular node to fall within the time window

associated with that point. Constraint 3.11 stipulates the starting point for the truck, where the arrival time at the starting point is earlier than all arrival times for this trip. Constraint 3.12 establishes the relationship between $x_{i,j}^{k,r}$ and $y_{i,j,p}^{k,r}$, implying that the truck load variable is only possible after using this arc. Constraint 3.13 indicates that for each shipper and each container demand, the inflow minus the outflow of truck loads must satisfy the demand of that shipper. For full containers, terminals must also satisfy the inflow minus the outflow, equaling the amount they need to send and receive. This is because for each shipper needing to pick up or drop off a full container at the terminal, the terminal has the opposite requirement: if the shipper needs to receive, the terminal must send. It's worth noting that each full container is unique, and this information is known in advance. However, for empty containers, which are homogeneous, terminal and depot do not adhere to flow conservation and require additional input and output. This is represented by constraint 3.14 and 3.15. Constraint 3.16 expresses the truck's capacity limitation, indicating that the truck can carry at most one 40-ft container or two 20-ft containers. Constraint 3.17 implies that the empty container inventory at the terminal can be replenished from any source, and the inventory is limited. The constraints 3.18 ensure the correct sequencing of truck trips, while 3.19 ensure the proper sequence of truck usage, i.e., truck 2 can not be used, if truck 1 is not being used. Constraints 3.20 to 3.23 define the ranges of the decision variables.

4

Validation

In this section, we will validate the model. The specific process involves generating two routing instances with 10 (2_2_6 instance) and 15 nodes (3_2_10 instance in the appendix) respectively. 2_2_6 instance represents 2 terminals, 2 depots and 6 shippers, same for 3_2_10 instance. The 15 nodes cases are shown in the Appendix. These instance models will be input into CPLEX for optimization to determine the optimal solution. Subsequently, we will verify the rationality and effectiveness of these optimal solutions. Through this careful validation process, we aim to demonstrate the robustness and accuracy of our model, enhancing its potential for practical implementation in optimizing transportation networks.

4.1. Generic settings

We will use the T_H_S instance to represent the generated instances, where T represents the number of terminals, H represents the number of depots, and S represents the number of shippers. Their indexes are consecutive, for example, in a 2_2_6 instance, the indexes for terminals are 0 and 1, the indexes for depots are 2 and 3, and the indexes for shippers range from 4 to 9. It's important to note that depot 0 corresponds to index 2, and shipper 0 corresponds to index 4, and so on.

Testing Platform: The CPU is an AMD Ryzen 6800H, with 8 cores and 16 threads, a base frequency of 3.2GHz, and a maximum boost clock frequency of 4.7GHz. The system is equipped with 16GB of RAM, operating at a frequency of 4800MHz. The version of CPLEX is 22.1.1.0 running in Windows 11 23H2.

4.2. Instance Generation of 2_2_6

4.2.1. Parameters Settings

In the configured instance, depots are defined as points within the instance that do not have container demand. Therefore, the demand parameters for depots are all set to zero. However, trucks can freely pick up or drop off empty containers at depots because they are not subject to flow conservation constraints for empty containers.

Additionally, the demand parameters for terminals are unique. The demand parameter for empty containers at terminals corresponds to inventory, indicating how much can be utilized. However, for each full container, a shipper's request corresponds to a destination or origin terminal, generating an opposite request. For example, if a shipper wants to send a container, a terminal needs to receive a corresponding container.

There are a total of six trucks. The first three trucks are exclusively designated to start from Terminal 0 and must return to Terminal 0 upon completion of their routes. Conversely, the remaining three trucks are assigned to commence their journeys from Terminal 1, with the stipulation that they also conclude their routes by returning to Terminal 1. Furthermore, both Terminal 0 and Terminal 1 are initially equipped with three 20-foot empty containers and three 40-foot empty containers each. The

large value number M is set to 2000.

The initial demand matrix p for each shipper i is shown in Table 4.1. The column sequence represents the demand as $p = [p_{40E}, p_{20E}, p_{40F}, p_{20F}]$, where -1 indicates dispatching one container, 0 indicates no demand, and 1 indicates a requirement for one container.

Table 4.1: Initial demand matrix for 2_2_6 instance

	E40	E20	F40	F20
Shipper 0	1	0	-1	0
Shipper 1	0	1	0	-1
Shipper 2	-1	0	0	0
Shipper 3	0	0	1	0
Shipper 4	0	1	0	0
Shipper 5	0	0	0	1

Once we have determined the demands of the shippers, we need to expand this matrix to encompass more information: creating a unique column for each full container. Each full container needs to specify from which terminal it is picked up or delivered to, in order to create corresponding terminal demands. Table 4.2 below illustrates the final expanded demand matrix.

Table 4.2: Demand parameter of 2_2_6 instance

	E40	E20	S3F40-terminal0	S0F40-terminal1	S1F20-terminal0	S5F20-terminal1
Terminal 0	3	3	-1	0	1	0
Terminal 1	3	3	0	1	0	-1
Depot 0	0	0	0	0	0	0
Depot 1	0	0	0	0	0	0
Shipper 0	1	0	0	-1	0	0
Shipper 1	0	1	0	0	-1	0
Shipper 2	-1	0	0	0	0	0
Shipper 3	0	0	1	0	0	0
Shipper 4	0	1	0	0	0	0
Shipper 5	0	0	0	0	0	1

The four ”3”s in the upper left corner of Table 4.2 each represent that the inventory of 40-foot and 20-foot empty containers for Terminal 0 and Terminal 1 are 3 units each. Shipper 0 and 5 request that their full containers (S0F40-terminal1 and S5F20-terminal1) be routed through Terminal 1, while Shipper 1 and 3 request that their full containers (S1F20-terminal0 and S3F40-terminal0) be routed through Terminal 0.

We designate the start of the time windows for terminals and depots as 0, marking the beginning of the day, and set their closure at 1440, which corresponds to the end of the day, encompassing a full 24-hour period or 1440 minutes. On the other hand, the time windows for shippers are generated randomly. A_i and B_i are shown below:

A_i = [0, 0, 0, 0, 477, 57, 52, 598, 190, 589]

B_i = [1440, 1440, 1440, 1440, 745, 150, 305, 857, 457, 883]]

The distance matrix employed represents temporal distances, that is, the time taken to travel from point i to point j. This matrix is symmetric, reflecting the principle that the time required for travel is the same in both directions. To prevent trucks from traveling within points themselves, we have strategically set

their temporal distances to be 1000(the diagonal of the matrix). The time distance matrix $C_{i,j}$ is shown below:

[[1000, 1000, 101, 71, 79, 17, 68, 70, 94, 131],
[1000, 1000, 57, 139, 37, 98, 83, 87, 65, 87],
[101, 57, 1000, 41, 12, 34, 119, 82, 82, 43],
[71, 139, 41, 1000, 83, 65, 107, 121, 112, 45],
[79, 37, 12, 83, 1000, 69, 62, 66, 91, 124],
[17, 98, 34, 65, 69, 1000, 62, 39, 110, 75],
[68, 83, 119, 107, 62, 62, 1000, 67, 69, 25],
[70, 87, 82, 121, 66, 39, 67, 1000, 96, 103],
[94, 65, 82, 112, 91, 110, 69, 96, 1000, 72],
[131, 87, 43, 45, 124, 75, 25, 103, 72, 1000]]

4.2.2. Results of 2_2_6 instance

CPLEX takes 3.25 seconds to get a optimal solution with objective 539. 0 and 1 of index i, j represent Terminal 0 and 1. 2 and 3 represent Depot 0 and 1. 4...n represent Shipper 0...$n-4$. k and r represent the vehicle's ID and the trip's index. p represents the column sequence of the container demand matrix. The result of $x_{i,j}^{k,r}$ is shown as table 4.3:

Table 4.3: Solution $x_{i,j}^{k,r}$ of 2_2_6 instance

k	r	i	j	Value
0	0	0	7	1
0	0	7	0	1
0	1	0	5	1
0	1	5	0	1
1	0	0	6	1
1	0	6	0	1
3	0	1	8	1
3	0	2	4	1
3	0	4	1	1
3	0	8	9	1
3	0	9	2	1

The result of $y_{i,j,p}^{k,r}$ is shown as table 4.4:

Table 4.4: Solution $y_{i,j,p}^{k,r}$ of 2_2_6 instance

p	k	r	i	j	Value
0	1	0	6	0	1
0	3	0	2	4	1
1	0	1	0	5	1
1	3	0	1	8	1
2	0	0	0	7	1
3	3	0	4	1	1
4	0	1	5	0	1
5	3	0	1	8	1
5	3	0	8	9	1

The result of $t_i^{k,r}$ is shown as table 4.5:

Table 4.5: Solution $t_i^{k,r}$ of 2_2_6 instance

i	k	r	Value
2	3	0	632
4	3	0	644
5	0	1	57
6	1	0	68
7	0	0	598
8	3	0	190
9	3	0	589

The truck transportation trajectories are shown below. Three trucks were utilized, namely truck 0, 1 and 3. From figure 4.1, we can describe the trajectories of the trucks along with the types of containers they transport as follows:

Truck 0's Trajectories

- Trip 0: Truck 1 travels from Terminal 0 to Shipper 1, and then back to Terminal 0. This trip involves transporting one 20-foot empty container $(p = 1)$ to Shipper 1 and bringing back one 20-foot full container $(p = 4)$.

- Trip 1: Truck 2 travels from Terminal 0 to Shipper 3, and then back to Terminal 0, carrying one 40-foot full container $(p = 2)$ to Shipper 3.

- Trip 0: Truck 0 moves from Terminal 0 to Shipper 2, and then back to Terminal 0. During this trip, it carries one 40-foot empty container $(p = 0)$ to Shipper 2.

Truck 1's Trajectories

- Trip 0: Truck 0 moves from Terminal 0 to Shipper 2, and then back to Terminal 0. During this trip, it carries one 40-foot empty container $(p = 0)$ to Shipper 2.

Truck 3's Trajectories

- Trip 0: Truck 3 undertakes a series of movements involving multiple locations:

- From Terminal 1 to Shipper 4, carrying one 20-foot empty container $(p = 1)$ and one 20-foot full container $(p = 5)$.

- From Shipper 4 to Shipper 5, moving one additional 20-foot full container $(p = 5)$.

- From Shipper 5 back to Depot 0.

- From Depot 0 to Shipper 0, with one 40-foot empty container $(p = 0)$.

- Finally, from Shipper 0 to Terminal 1, transporting one 40-foot full container $(p = 3)$.

The above trajectories and truck container loading and transportation conditions are reasonable and meet the time window, so we can say that the test case for the first 10 nodes instance passed.

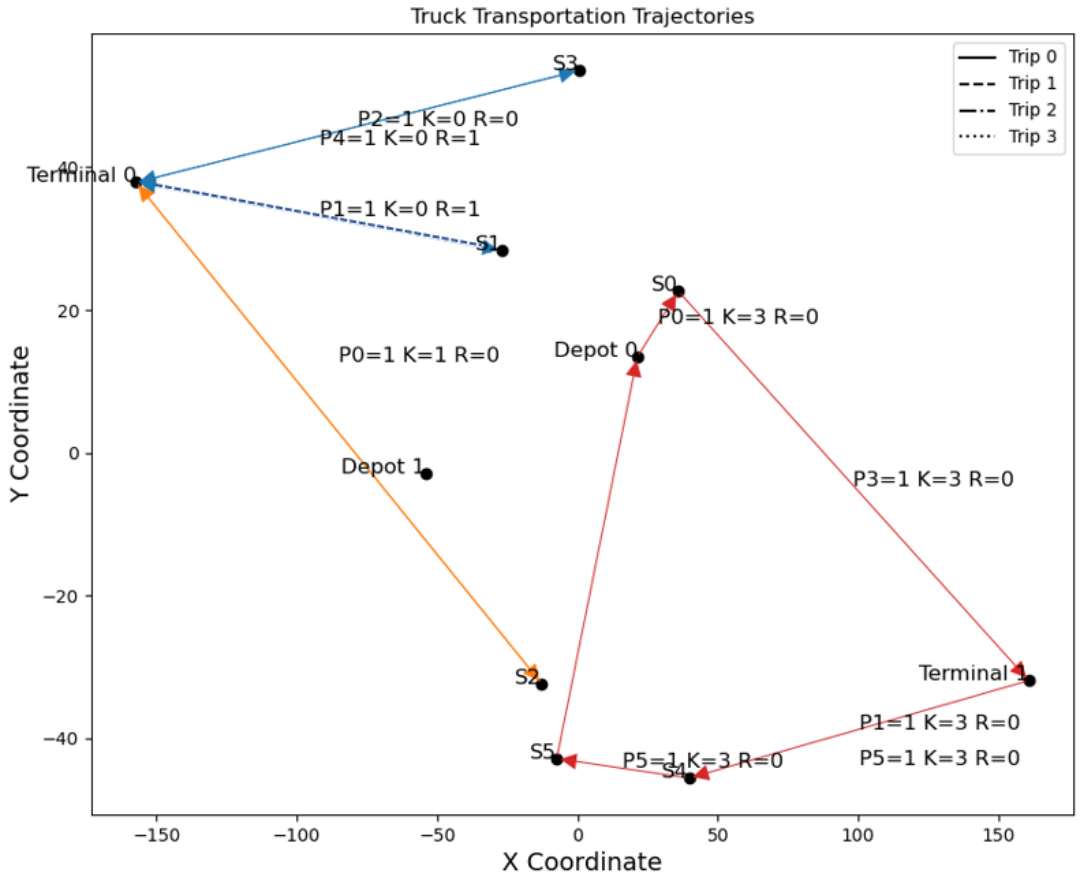

Figure 4.1: Truck Transportation Trajectories 2_2_6

5

A large adaptive neighborhood search algorithm

We introduces a novel mathematical model to address a variant of the container drayage problem, known for its NP-hard nature. Traditional computational solvers like CPLEX, while effective for smaller instances, often struggle with scalability due to prohibitive computational demands and extensive solution times. Consequently, this investigation turns towards metaheuristic algorithms, particularly focusing on the Adaptive Large Neighborhood Search (ALNS) algorithm. ALNS is recognized for its ability to manage complex, large-scale optimization challenges efficiently. It diverges from traditional local search strategies by allowing larger, more disruptive changes to the solution, thereby effectively exploring and exploiting the vast search space. With its adaptive mechanism that dynamically adjusts the search strategy based on the performance of different neighborhood structures, ALNS provides a robust tool for iteratively refining solutions and navigating through complex problem landscapes. As such, this chapter will detail the application of ALNS to our container drayage problem.

The Adaptive Large Neighborhood Search with Simulated Annealing pseudo-code is shown in Algorithm 1. This algorithm employs the roulette wheel selection to choose the destruction and repair operators. Line 1 indicates the initialization of the roulette wheel parameters where the scores vector $\vec{c}$ with parameters c_1, c_2, c_3, c_4 correspond to different scenarios after applying an operator: c_1 for when the candidate solution s' is a new global best s_{best}, c_2 for when the candidate solution s' is better than the current solution s but not a new global best s_{best}, c_3 is the case when the candidate solution s' is accepted, excluding the scenarios of c_1 and c_2, and c_4 for when the candidate solution s' is rejected. Line 2 initialize the weights for all destroy and repair operator. Then, on line 24, the weights of the operators are updated.

Line 3 specifies the parameters of the Simulated Annealing algorithm used as the acceptance criterion, which are the initial and final temperatures $T_{\text{start}}, T_{\text{end}}$, the rate of temperature decay, and the maximum number of iterations $N_{\max}$ without any solution updates, meaning that the search will end once $N_{\max}$ is reached. This parameter is crucial to the search results.

Line 4 specifies the number of points to be removed during the destruction phase, denoted as n_{remove}, which is also a very critical parameter. If it is too small, there is a possibility of falling into local optima, whereas if it is too large, it could significantly increase the solution time. Line 5 involves obtaining the initial solution from the algorithm for initial solution generation, referenced as Algorithm 2. Line 6 sets the first s_{best} as the initial solution. Line 7 is a counter for the number of iterations.

Line 8 indicates the start of the loop for the search process. Lines 9-12 describe the selection of a destroy operator based on the probability weights of each operator to remove points from the solution, followed by the use of a repair operator to mend the damaged solution s'. Lines 13-21 involve evaluating the repaired solution s' to decide whether to accept it as s_{best} or the current solution s (if it passes the simulated annealing criterion, s' may be accepted even if $s' > s$). Line 22 is for updating the weights

Algorithm 1 Adaptive Large Neighborhood Search with Simulated Annealing

1: Define scores vector $\vec{c} = [c_1, c_2, c_3, c_4]$ for outcomes, operator decay rate θ
2: Initialize all operators' weights $\omega_i = 1$ for all i
3: Initialize temperature $T_{\text{start}}, T_{\text{end}}$, temperature cooling rate α_t, max iterations $N_{\max}$
4: Initialize remove coefficient ρ, number of nodes to remove: $n_{\text{remove}} = \lceil \rho \cdot \text{shippers in } s \rceil$
5: Initialize solution s
6: Initialize best solution $s_{\text{best}} \leftarrow s$
7: $noImprovementIterations \leftarrow 0$
8: **while** $noImprovementIterations < N_{\max}$ **do**
9: Select a destroy operator d using roulette wheel selection based on weights ω
10: Generate a partial solution s' by applying d to remove n_{remove} points from s
11: Select a repair operator r using roulette wheel selection based on weights ω
12: Generate a candidate solution s' by repairing s' using r
13: **if** $s' < s_{\text{best}}$ **then**
14: $s_{\text{best}} \leftarrow s'$
15: $noImprovementIterations \leftarrow 0$ ▷ Reset the counter
16: **else**
17: $noImprovementIterations \leftarrow noImprovementIterations + 1$
18: **end if**
19: **if** $s' < s$ **or** $\exp\left(-\frac{\text{cost}(s') - \text{cost}(s)}{T}\right) > \text{random}(0, 1)$ **then**
20: $s \leftarrow s'$ ▷ Accept the new solution
21: **end if**
22: Update ω_d and ω_r based on the outcome, scores $\vec{c}$ and θ
23: $T \leftarrow T \cdot \alpha_t$ ▷ Decrease temperature
24: **end while**
25: **return** s_{best}

of the operators based on the acceptance of the solution. Line 23 involves cooling down. The process ends the entire ALNS algorithm.

5.1. Encoding

In this thesis, the representation of solutions is exemplified by the optimal solution discussed in Section 4.2. The routes of the trucks are as follows:

Routes:

$[[0, 5, 0],$

$[0, 7, 0],$

$[0, 6, 0],$

$[1, 8, 9, 2, 4, 1]]$

Each route starts and ends at a terminal (0 or 1), with the truck sequentially visiting each node. Taking $[0, 5, 0]$ as an example, the truck departs from terminal 0, visits shipper 1, and then returns to terminal 0.

The arrival times of the trucks are also formatted like the routes, where the position of an element corresponds to the time. The unit is minutes. Again, using the first route as an example, the arrival times of the truck are:

Routes time:

$[40, 57, 74]$

Additionally, the capacity of the trucks on each arc is still ordered sequentially. Taking the first route as an example, the variable representing the truck's load is:

Routes truck load:

$[0,0,0,0,0,0],[1,0,0,0,0,0]$

The first $[0,0,0,0,0,0]$ represents the truck traveling from terminal 0 to shipper 1 without loading any containers, and the second $[1,0,0,0,0,0]$ indicates the truck carrying one 40ft empty container from shipper 1 back to terminal 0.The columns of each truck load have the same meaning as the columns of the Demand parameter.

5.2. Generation of the initial solution

5.2.1. Generation process

The method for generating initial solutions in ALNS is based on the algorithm 2. In this thesis, we randomly sample from a broad solution space and then estimate the optimal solution based on these samples. The specific steps are as follows: a random generation function is called to insert nodes into routes randomly. For initial solutions, we only consider inserting essential nodes in the middle of trips, namely shippers and depots (if the inventory of empty containers at the terminal is insufficient).

Regarding shippers, we first group them based on their distance from the terminal. For all terminals, if shipper 0 is closer to terminal 0, then shipper 0 will be allocated to the group of terminal 0. This means that shipper 0 can only be serviced by trucks originating from terminal 0.

However, if some shippers have requests for full containers to specific terminal, these shippers are then re-allocated to the group of the corresponding terminal. Subsequent insertions of shippers can only be made into routes of the corresponding terminal group.

As for depots, since the demand for empty containers by shippers and the inventory at terminals are known, we can calculate the minimum number of depot visits needed to replenish empty containers. The formula is:

$$\begin{aligned}\text{min number of visiting depot} = &\max(0, \text{demand\_E40ft} - \text{stock\_E40ft}) \\ &+ \frac{\max(0, \text{demand\_E20ft} - \text{stock\_E20ft})}{2}\end{aligned} \tag{5.1}$$

where $demand\_E40ft$ represents the demand for 40ft empty containers from all shippers, $demand\_E20ft$ represents the demand for 20ft empty containers from all shippers, and $stock\_E40ft$ and $stock\_E20ft$ respectively represent the inventory of 40ft and 20ft empty containers at the terminal. In the initial solution, depots are randomly inserted.

Each solution is then checked for feasibility. There are four feasibility checks, which are: time window checks, truck capacity checks, checks against exceeding the number of trucks in the fleet and the number of trips, and checks on the inventory of empty containers at the terminal. Among them, time window checks are to check whether the time when the truck arrives at each node is within the corresponding time window. Truck capacity checks check whether the container loaded by the truck on each arc exceeds the maximum capacity of the truck. If a solution is feasible, its cost is calculated. If this cost is lower than the currently known minimum cost, the solution is updated as the optimal solution. Otherwise, the process continues in a loop until the target number of iterations $N_{initial}$ is reached.

5.2.2. Initial solution of 2_2_6 network

Here we still take 2_2_6 network in section 4.2 as example. First, we set the maximum number of iterations to 10,000. In this network, since the demand for 40ft and 20ft empty containers by shippers is less than the inventory at terminals, we know that the minimum number of depots to be inserted is zero. Therefore, no depots are inserted in the initial solution, resulting in an objective function value of 608 for the initial solution. The changes in the objective function value during the search process are shown in the following figure 5.1:

Compared to the optimal value of 539, it is 12.8% higher. The trajectories of the trucks are shown in the figure 5.2:

Algorithm 2 Initial Solution Generation

$best\_solution \leftarrow None$
$min\_cost \leftarrow \infty$
$generated\_count \leftarrow 0$
$iterations \leftarrow [\,]$ ▷ List to save iteration counts
while $generated\_count < N_{initial}$ **do**
    $solution \leftarrow generate\_initial\_solution\_without\_check()$
    if $feasible$ **then**
        $generated\_count \leftarrow generated\_count + 1$
        $cost \leftarrow calculate\_cost(solution)$
        $iterations.append(generated\_count)$
        if $cost < min\_cost$ **then**
            $min\_cost \leftarrow cost$
            $best\_solution \leftarrow solution$
        end if
    end if
end while
return $best\_solution$

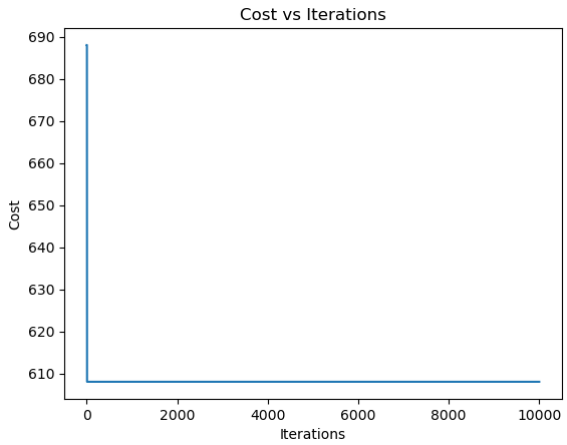

Figure 5.1: Cost vs Iterations 2_2_6 with no depot

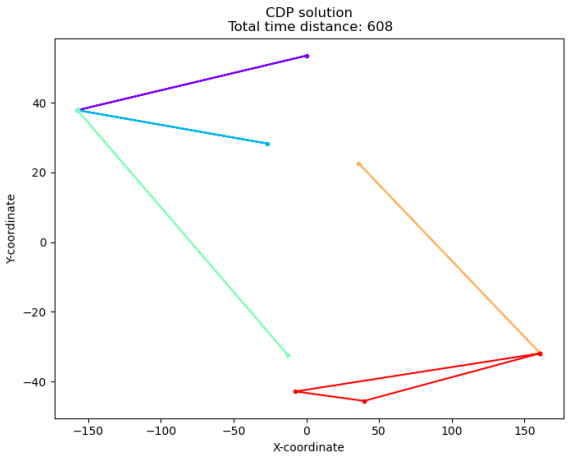

Figure 5.2: Truck transportation trajectories 2_2_6 with no depot

However, we have already obtained the optimal solution through CPLEX. In the optimal solution, the trucks made an additional visit to a depot. We can verify the effectiveness of the initial solution by setting the minimum number of depot visits in the initial solution to 1. With this adjustment, while keeping other settings unchanged, the search process for the newly generated initial solution is shown in Figure 5.3.

It can be observed that, after adding one depot, the initial solution directly matches the optimal solution, both being 539. The trajectory of the trucks in the initial solution (Figure 5.4) and the optimal solution (Figure 4.1) are the same. This demonstrates the reasonableness of the initial solution.

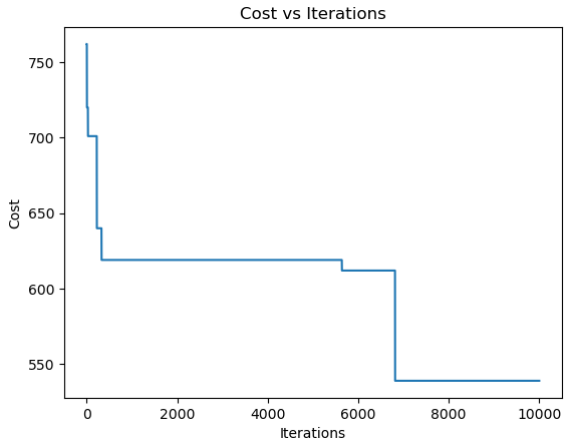

Figure 5.3: Cost vs Iterations 2_2_6 with one depot

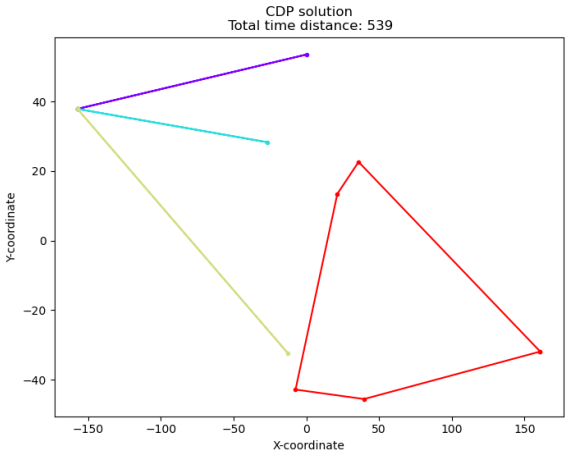

Figure 5.4: Truck transportation trajectories 2_2_6 with one depot

5.3. Destory Operator

This chapter will introduce two variants of the Destroy Operator called ”worst removal” originating from ropke2006adaptive. It takes a solution as input and outputs a solution from which n_{remove} points have been removed, as well as the original solution, which remains unaltered. The two variants of the ”worst removal” operator refer to the following methods: the first variant ”worst removal” involves removing shippers, terminals, and depots, and temporarily storing the removed points for future reinsertion into the solution using the Repair Operator. The second variant ”permanent removal” also involves removing shippers, terminals, and depots, but for terminals and depots, they are permanently removed and not reinserted into the solution during subsequent use of the Repair Operator.

5.3.1. Worst Removal

The ”Worst Removal” operator, as depicted in the algorithm 3, is a procedure designed to iteratively remove nodes from a given $solution$ with the goal of subsequent optimization.

The parameter p modulates the degree of randomness when selecting the nodes to be removed, thereby introducing a controlled stochastic element to the operation. By doing so, it mitigates the risk of consistently choosing the same nodes for removal, which can lead to a cyclic search pattern and potentially trap the optimization process in a local optimum.The randomness control facilitated by p ensures that the removal of nodes is not purely deterministic based on the removal cost, but rather, it is skewed towards worse-performing nodes while still allowing for variation. Each iteration of the operator removes a node from the list $L\_removal$, which consists of nodes sorted by their removal cost in descending order. The higher the removal cost of a node, the more likely it is to be removed, which signifies its detrimental impact on the current solution’s quality. Each node selected for removal is temporarily stored in $unassigned$, allowing for the possibility of reinsertion by a repair operator.

The final return comprises two parts: the $destroyed$ solution, which is the modified version of the original, and the unmodified $solution$. The rationale for returning the original solution is that, if the new solution post-repair is infeasible (implying that all available trucks are utilized with no surplus capacity to accommodate the insertion of nodes into any route), the process can be reset to before the application of the destroy and repair operators, effectively reverting to the original $solution$.

5.3.2. Permanent Removal

In the ”Permanent Removal” algorithm, as defined by the pseudocode under algorithm 4, nodes are removed from the solution based on a calculated removal cost. Differing from the ”Worst Removal” method detailed in algorithm 3, ”Permanent Removal” introduces a conditional step that dictates the finality of the node removal. If a node designated for removal is identified as either a terminal or a shipper, it is not stored in the $unassigned$ subset, implying that these nodes are not candidates for reinsertion into the $solution$ by any subsequent repair operators. This makes the removal permanent,

Algorithm 3 Worst Removal

```
1: function WorstRemoval(solution)
2:     p ← parameter for randomness control
3:     destroyed ← copy of solution
4:     L_removal ← list of nodes sorted by decreasing removal cost
5:     while n_remove > 0 do
6:         y ← random number between 0 and 1
7:         r_index ← integer part of (len(L_removal) × y^p)
8:         r ← L_removal[r_index]                          ▷ Select node to remove
9:         store r in destroyed.unassigned
10:        remove r from destroyed
11:        n_remove ← n_remove − 1
12:    end while
13:    return destroyed, solution
14: end function
```

contrasting with "Worst Removal," where all removed nodes are temporarily held in $unassigned$ with potential for reintegration into the $solution$.

Algorithm 4 Permanent Removal

```
1: function PermanentRemoval(solution)
2:     p ← parameter for randomness control
3:     destroyed ← copy of solution
4:     L_removal ← list of nodes sorted by decreasing removal cost
5:     while n_remove > 0 do
6:         y ← random number between 0 and 1
7:         r_index ← integer part of (len(L_removal) × y^p)
8:         r ← L_removal[r_index]                          ▷ Select node to remove
9:         if not (r is terminal or r is shipper) then
10:            store r in destroyed.unassigned
11:        end if
12:        remove r from destroyed
13:        n_remove ← n_remove − 1
14:    end while
15:    return destroyed, solution
16: end function
```

5.4. Repair Operator

This chapter will introduce the repair operators used. The ALNS algorithm actually employs three repair operators, all fundamentally based on Greedy Repair. They take two inputs: one is the $destroyed$ that has undergone removal, and the other is the unmodified $solution$. They produce one output, which is a feasible $solution$. It is important to note that the returned feasible $solution$ could either be the repaired $solution$ or the unmodified $solution$. The basic repair operator is shown in algorithm 5, and the other two repair operators simply add terminals and depots to the $destroyed.unassigned$.

5.4.1. Greedy Repair

The "Greedy Repair", presented in Algorithm 5, is designed to reintegrate unassigned nodes into a previously disrupted solution. The process commences with a random shuffle of the nodes in the $destroyed.unassigned$ list, introducing variability in the order of node reinsertion.

Each iteration of the repair cycle focuses on one node at a time, extracted from the $destroyed.unassigned$ list. The algorithm evaluates the most cost-effective and feasible route and position for inserting this node, taking into consideration constraints such as the compatibility of time windows and truck load capacities. If an appropriate insertion point is found, the node is inserted accordingly, thereby incre-

mentally reconstructing the route.

However, if no feasible insertion can be identified—typically due to all trucks being fully utilized or the absence of a route segment capable of accommodating the node—the algorithm terminates and returns the original, unmodified solution.

Following the attempt to reinsert all unassigned nodes, a final verification step assesses the terminal stock levels. If this verification confirms that the adjusted routes are feasible, the modified solution ($destroyed$) is accepted and returned. Conversely, if the terminal stock check fails, indicating that the repaired configuration does not meet required specifications, the original $solution$ is reinstated.

Algorithm 5 Greedy Repair

```
1: function GreedyRepair(destroyed, solution)
2:     Randomly shuffle the destroyed.unassigned nodes
3:     while not empty destroyed.unassigned do
4:         node ← pop from destroyed.unassigned
5:         Get the best route and position for insertion based on insert cost if time windows and truck
   load allow
6:         if can find the the best route and position then
7:             Insert node into destroyed.routes at the best route and position
8:         else
9:             return solution ▷ Return original solution (all trucks are utilized and no segment of any
   route can accommodate the insertion)
10:        end if
11:    end while
12:    Check terminal stock
13:    if terminal stock check pass then
14:        return destroyed
15:    else
16:        return solution                    ▷ Return original solution if repair is infeasible
17:    end if
18: end function
```

5.5. Selection Scheme

The Roulette Wheel selection method is an integral part of the Adaptive Large Neighborhood Search algorithm, embodying its adaptive nature. Initially, each operator i is assigned a uniform weight $\omega_i = 1$. The weights are dynamically adjusted based on the performance of the operators, with the normalization of these weights guiding the probabilistic selection of operators.

Upon the application of an operator, one of four possible outcomes occurs, each assigned a corresponding score c_j for $j = 1, \ldots, 4$ as shown in table 5.1. Where s_{best} is the gobal best solution, s is the current solution, s' is the candidate solution.

Table 5.1: Scores of Roulette Wheel selection

Score	Scenario
c_1	$s' > s_{\text{best}}$
c_2	$s_{\text{best}} > s' > s$
c_3	$s_{\text{best}} > s > s'$, s' is accepted
c_4	$s_{\text{best}} > s > s'$, s' is rejected

The update of operator weights is executed using the following equations 5.2 and 5.3:

$$\omega_d = \theta\omega_d + (1-\theta)c_j, \tag{5.2}$$

$$\omega_r = \theta\omega_r + (1-\theta)c_j, \tag{5.3}$$

where θ represents the operator decay rate, ranging from 0 to 1. This decay rate dictates the memory of the historical performances, controlling the emphasis placed on recent operator performances.

The probability of selecting an operator for a subsequent iteration is calculated as equation 5.4:

$$P(i) = \frac{\omega_i}{\sum_k \omega_k}, \tag{5.4}$$

where $P(i)$ is the selection probability of the i-th operator, directly proportional to its current weight. This probabilistic approach ensures that more successful operators are selected more frequently, thus fostering an adaptive search process.

5.6. Acceptance Criterion

In this work, the Simulated Annealing (SA) method has been selected as the acceptance criterion for its robustness in handling complex optimization problems by effectively navigating the solution space. The criterion for accepting new candidate solutions s' can be mathematically expressed as equation 5.5:

$$\text{accept if } s' < s \text{ or } s' < s_{\text{best}} \text{ or } \exp\left(-\frac{\text{cost}(s') - \text{cost}(s)}{T}\right) > \text{random}(0,1) \tag{5.5}$$

where T is the current temperature, analogous to thermodynamic temperature in physical processes. random$(0,1)$ generates a uniform random number between 0 and 1.

This acceptance criterion ensures that any new solution s' that improves upon either the current solution s or the best solution s_{best} is automatically accepted, promoting continuous improvements whenever possible. Additionally, solutions that are worse than both s and s_{best} can also be accepted but with a probability that is exponentially dependent on the cost difference between s' and s and inversely on the current temperature T. This probability serves to facilitate extensive exploration early in the algorithm's run when T is high, allowing the algorithm to escape local minima.

The temperature is adjusted according to the temperature cooling rate α_t, where the new temperature after each iteration is calculated by equation 5.6:

$$T \leftarrow T \cdot \alpha_t \tag{5.6}$$

This exponential cooling schedule gradually reduces the temperature, shifting the algorithm from a state of exploration to more intensive exploitation of the refined areas of the solution space as T approaches lower values.

5.7. Stopping Criterion

In this ALNS, we employ the *NoImprovement* stopping criterion to determine the termination point of the optimization process. This criterion effectively halts the algorithm if there is no enhancement in the best solution's quality over a defined number of iterations. The primary parameter governing this criterion is $N_{\max}$, which represents the maximum number of consecutive iterations without any improvement in the best solution's objective value.

6

Computational experiments

6.1. Instance setting

All test instances presented in this thesis were conducted on a personal computer, equipped with an AMD Ryzen 6800H processor, which boasts 8 cores and 16 threads, a base clock frequency of 3.2GHz, and a maximum boost clock up to 4.7GHz, alongside 16GB of RAM at a memory speed of 4800MHz. The benchmark platform used was CPLEX Studio IDE version 22.1.1.0, with the maximum CPU runtime for testing instances capped at one hour. The ALNS algorithm was implemented in Visual Studio 2024, utilizing a framework adopted from Wouda and Lan, 2023, with the programming conducted in Python, version 3.9.18.

The test instances within this thesis are categorized into two types: one with randomly generated positions for terminals, depots, and shippers, and the other based on real geographical coordinates of terminals and depots, with shippers randomly positioned around them.

For the first type, where terminals, depots, and shippers are randomly placed within the Euclidean plane, the time distance between them is distributed between [100, 151] minutes, indicating a travel time ranging from 100 to 150 minutes. The parameter F_p is set to 1. Time windows for terminals and depots are set to [0, 1440] minutes, signifying 24-hour accessibility. The start of the time window for shippers is randomly generated within [0, 801] minutes, with the corresponding end time window being randomly extended by [200, 301] minutes from the start time. Shipper container requests are randomly selected from Table 3.1. Each truck's maximum number of daily trips is capped at four.

For the second type of instance, we utilize three container terminals located in Belgium and the Netherlands, specifically BCTN terminals, with the depot positions identical to the BCTN terminals, as a terminal can serve the functions of a depot. The three terminals are located at Den Bosch (51.70, 5.27), Geel (51.11, 5.02) and Roermond(51.20,5.99). The shippers are randomly positioned within a rectangular area defined by the corners, their latitude and longitude coordinates are: top-left (51.72, 4.95), bottom-left (51.07, 4.95), top-right (51.72, 6.03), and bottom-right (51.07, 6.03), covering shippers in Germany, Belgium, and the Netherlands. The speed of the truck is 40km/h. Other parameter settings are identical to those of the first type of randomly generated instances.

6.2. Parameter tuning

In this section, we will divide our discussion into two parts. The first part introduces all the parameters used in the ALNS algorithm that can affect the quality of the search. In the second part, we will discuss how we adjusted these parameters and identify which ones had a significant impact on the results.

6.2.1. Parameters of ALNS

We will introduce the parameters used in the ALNS algorithm according to the order in which they are utilized. First is N_{initial}, which controls the number of initial solutions generated in the initial solution generation algorithm 2. It generates N_{initial} initial solutions and selects only the one with the lowest

cost.

After obtaining the initial solution, we proceed to the cycle of destroy and repair of the solution using the destroy operator and the repair operator. At this point, the parameter $N_{\max}$ is used to determine when to end the cycle, i.e., the cycle exits when the newly generated solution does not improve after $N_{\max}$ iterations.

The parameter w_i is uniformly set to 1 for all operators, indicating the weight of each operator being selected in each cycle. This parameter serves as a base value and does not impact the results.

In the destroy operator, parameters ρ and p are used. ρ represents the destruction rate, i.e., a percentage ρ of the total number of selected nodes is removed from the current solution. p controls the randomness of the selection in the ”worst removal” operator; the higher the value, the more random it is.

The repair operator performs a greedy repair based on the insertion cost of each node, without using any parameters.

After a solution is repaired, it must be evaluated to determine if it will be accepted. At this point, the simulated annealing algorithm uses the parameters T_{start}, T_{end}, and α_t, which represent the starting temperature, ending temperature, and temperature cooling rate, respectively.

Finally, depending on the acceptance of the solution, a roulette wheel is used to update the weights w_i of the operators, involving parameters c_1, c_2, c_3, c_4, where c_4 is set to 0 by default because it represents the worst-case scenario which should not be perpetuated. The values of c_1, c_2, c_3 will be adjusted in the next section. θ represents the memory of the historical performances; the larger the value, the greater the impact of historical performance.

6.2.2. ALNS parameters tuning

We conducted parameter tuning for the parameters mentioned above. The default values of the parameters were experimentally derived during the development of the algorithm and were optimized for small instances. Based on this, we proposed a reasonable range of values for each parameter, divided into four categories. When we test each parameter, we keep other parameters unchanged at their default values and only change the parameters we are currently interested in. We recalculated each value five times, took the average, and selected the value with the smallest mean as the final parameter for the ALNS. There are 11 parameters in total, and each set of parameters has 4 possible values. Each value is tested 5 times, and a total of $11 * 4 * 5 = 220$ sets are tested. Here, Gap represents the difference between the best and worst performing values for that parameter.

The final results of the parameter tuning are shown in the table 6.1. From this, we can see that the gap for parameters $N_{initial}$, c_1 and θ is less than 1%, indicating their minor impact on the results. The parameters $T_{\text{start}}, \alpha_t, c_3$, and p have a more significant impact on the outcomes. A higher T_{start} worsens the average results, suggesting that too high an initial temperature at the beginning of the algorithm leads to the acceptance of too many bad solutions, resulting in suboptimal final outcomes. α_t shows the largest gap at a value of 0.9, indicating that the temperature decreases too quickly, preventing the algorithm from escaping local optima. The best value for c_3 is at the minimum range of 1, and increasing c_3 means that the difference between accepting a better or a worse solution decreases. The optimal value for parameter p occurs at 1, and the worst at 10, suggesting that a higher p value increases randomness, making it difficult for the algorithm to identify the optimal node to destroy. Other parameters have little impact on the results of the algorithm.

While we could iterate the parameter-finding process, Ropke and Pisinger, 2006 have noted that parameter tuning for ALNS typically yields sufficiently high performance after just one round. Consequently, we concluded the process following a single round.

Table 6.1: Parameter tuning

Parameter	Range	Default	Best performance	Gap
ρ	0.2, 0.3, 0.4, 0.5	0.3	0.5	1.19%
T_{start}	100, 200, 300, 400	100	100	3.33%

T_{end}	0.001, 0.01, 0.1, 1	0.001	1	1.59%
α_t	0.9, 0.95, 0.99, 0.995	0.995	0.995	5.12%
$N_{\max}$	100, 200, 300, 400	300	400	2.09%
$N_{initial}$	1, 10, 50, 100	1	1	0.72%
c_1	10,15,20,25	25	10	0.54%
c_2	5,6,7,8	5	5	2.69%
c_3	1,2,3,4	1	1	4.41%
θ	0.6, 0.7, 0.8, 0.9	0.8	0.8	0.19%
p	1, 3, 5, 10	5	1	4.32%

6.3. Results of ALNS

This section presents the experiment results for small-, medium-, and large-scale instances, with small and medium instances being randomly or real-world generated, accompanied by comparative computational outcomes from both CPLEX and ALNS algorithms. Large-scale instances, based on the actual geographical locations of terminals and depots, are solely evaluated using the ALNS algorithm due to CPLEX's inability to initialize and provide a lower bound for such extensive cases. The configuration of instances is detailed in Section 6.1.

6.3.1. Experiments on small-scale instances

In this section, we show a total of 27 small-scale instances in table 6.2. In the setting notation $T\_H\_S\_N$, T represents the number of terminals, H denotes the number of depots, S indicates the number of shippers, and N specifies the sequence number of the instance under the same configuration of terminals, depots, and shippers. Here, 0 represents the first instance under a specific configuration, while 1 denotes the second instance. If there is no fourth one representing N, it means that this configuration is unique. The symbol Obj. represents the objective function value, Best bound denotes the best lower bound calculated by CPLEX, and Gap refers to the discrepancy between Obj. and Best bound. For the ALNS algorithm, Obj. Imp indicates the improvement in the objective function value compared to that obtained by CPLEX.

For small-scale instances, CPLEX demonstrates considerable capability to achieve optimal solutions within a 3600-second limit for certain configurations, such as instances 1-6, 14, 23, and 24. Notably, the largest of these configurations is 3_2_11, which involves 11 shippers being served by 3 terminals and 2 depots. As the size of the instance increases beyond this point, CPLEX often struggles to find the optimal solution. Specifically, for instances with up to 12 points, CPLEX computes solutions quickly, ranging from 3 to 11 seconds. Despite this, the ALNS algorithm not only achieves but also consistently finds the optimal solutions, often with a significant time advantage. This illustrates the effectiveness and correctness of the ALNS algorithm. For example, in Instance 24 (3_2_11), while CPLEX requires 825 seconds to compute the solution, ALNS completes the task in just 26.8 seconds, nearly 30 times faster. Instances 9 ($2\_2\_10\_0$) through 13 ($2\_2\_12\_0$) exemplify a continuing trend of increased complexity and difficulty for CPLEX, with gaps ranging from 2.1% to 29.6% and CPU times. The ALNS algorithm not only generally improves upon the objective values but also delivers these improvements significantly faster. Notably, in Instance 10 ($2\_2\_10\_0$), ALNS enhanced the solution by 21.6% while requiring only 13 seconds of computation time, in stark contrast to the exhaustive time consumed by CPLEX.

For Instance 16 ($2\_2\_13\_1$), CPLEX encountered significant difficulty, evidenced by a gap of 15.1%. In contrast, ALNS demonstrated greater efficiency, improving the objective function by 4.2% in a substantially shorter time period. Instance 22 ($2\_2\_16\_1$) illustrates the significant performance gap between CPLEX and ALNS, with ALNS not only achieving a better objective value (10.9%) but also demonstrating substantial improvements in efficiency and computational time(21.5s). Moving to Instances 25 (3_2_12) through 27 (3_2_15), CPLEX's performance sharply declines as it fails to efficiently solve these instances, exhibiting gaps as high as 20% while using up the allotted 3600 seconds of CPU time. Conversely, ALNS showcases its efficiency by improving CPLEX's objective values by 7.2% in Instance 25 and reducing the computational time significantly to just 13.3 seconds. For instances 26 (3_2_13) and 27 (3_2_15), CPLEX was unable to provide an integer solution within the allocated time frame, managing only to offer a lower bound. This limitation reflects the increasing difficulty of solving larger and more complex optimization problems using traditional methods. As the instance size expands, the com-

Table 6.2: Small-scale instances

Instance	Setting	CPLEX				ALNS		
		Obj.	Best bound	Gap%	CPU time(s)	Obj.	Obj. Imp(%)	CPU time(s)
1	2_2_6_0	548	548.0	0.0	3	548	0.0	2.2
2	2_2_6_1	614	614.0	0.0	4	614	0.0	4.8
3	2_2_7_0	1240	1240.0	0.0	7	1240	0.0	3.4
4	2_2_7_1	1609	1609.0	0.0	11	1609	0.0	8.2
5	2_2_8_0	1612	1612.0	0.0	11	1612	0.0	4.7
6	2_2_8_1	1657	1657.0	0.0	1050	1657	0.0	6.2
7	2_2_9_0	1925	1560.6	18.9	3600	1925	0.0	4.2
8	2_2_9_1	1678	1466.1	12.6	3600	1678	0.0	6.3
9	2_2_10_0	1929	1801.3	6.6	3600	1874	2.9	20.2
10	2_2_10_1	2124	1496.4	29.6	3600	1747	21.6	13.0
11	2_2_11_0	2185	1817.9	16.8	3600	1961	11.4	8.0
12	2_2_11_1	2099	1845.4	12.1	3600	2079	1.0	18.3
13	2_2_12_0	2209	2161.9	2.1	3600	2209	0.0	17.2
14	2_2_12_1	2310	2310	0.0	297	2310	0.0	21.0
15	2_2_13_0	2364	2133.7	9.7	3600	2235	5.8	17.8
16	2_2_13_1	2529	2146.5	15.1	3600	2426	4.2	20.0
17	2_2_14_0	2825	2455.2	13.1	3600	2622	7.7	27.2
18	2_2_14_1	2676	2379.8	11.1	3600	2518	6.3	26.4
19	2_2_15_0	3048	2685.9	11.9	3600	3034	0.5	23.6
20	2_2_15_1	3047	2720.7	10.7	3600	3025	0.7	21.7
21	2_2_16_0	3285	2976.2	9.4	3600	3273	0.4	21.9
22	2_2_16_1	3134	2544.4	18.8	3600	2826	10.9	21.5
23	3_2_10	1866	1866	0.0	100	1866	0.0	9.5
24	3_2_11	2345	2345	0.0	825	2345	0.0	26.8
25	3_2_12	2526	2020.8	20.0	3600	2357	7.2	13.3
26	3_2_13	-	2161	-	3600	2934	-	14.9
27	3_2_15	-	2494.2	-	3600	2830	-	18.8

putational challenges intensify, highlighting the necessity for more efficient or alternative optimization strategies capable of handling such complexity within reasonable time limits.

6.3.2. Experiments on medium-scale instances

We presented 23 medium-scale instances as shown in Table 6.3. For these medium-scale instances, CPLEX was unable to compute optimal solutions. From Instance 28 ($2\_2\_17\_0$) to Instance 38 (2_2_22), the gaps between the solutions found by CPLEX within one hour and the best bounds ranged from 2.3% to 24.6%. The ALNS algorithm improved upon CPLEX's results by 0.3% to 15.9%, completing each instance in under 34 seconds.

From Instance 39 (2_2_23) to Instance 43 (2_2_27), CPLEX failed to provide a feasible integer solution, only managing to produce a best bound. Hence, it is not possible to compare ALNS's results directly against those of CPLEX for these instances. However, when viewing from the perspective of best bounds, the solutions from ALNS closely approximated the lower bounds for each instance.

From Instance 44 (2_2_28) to Instance 50 (2_2_34), CPLEX was unable to complete the initialization, indicating insufficient memory during the probing phase and thus failing to solve. Nonetheless, the ALNS algorithm continued to provide feasible solutions despite these challenges.

6.3.3. Experiments on medium- and large-scale real-world instances

We also conducted computational experiments on medium-scale instances (Instances 51-55) and large-scale instances (56-60). We repeated the calculations 5 times for each instance and recorded the

Table 6.3: Medium-scale instances

Instance	Setting	CPLEX				ALNS		
		Obj.	Best bound	Gap%	CPU time(s)	Obj.	Obj. Imp(%)	CPU time(s)
28	2_2_17_0	3406	3271.8	3.9	3600	3349	1.7	23.8
29	2_2_17_1	3531	2974.7	15.8	3600	3522	0.3	31.8
30	2_2_18_0	3986	3006.6	24.6	3600	3438	15.9	23.7
31	2_2_18_1	3567	3258.6	8.7	3600	3413	4.5	28.2
32	2_2_19_0	4123	3356.5	18.6	3600	3735	10.4	23.4
33	2_2_19_1	3455	3091.4	10.5	3600	3401	1.6	39.0
34	2_2_20_0	3812	3724.7	2.3	3600	3796	0.4	29.5
35	2_2_20_1	3999	3219.9	19.5	3600	3861	3.6	31.5
36	2_2_21_0	4248	3816.0	10.2	3600	4193	1.3	39.5
37	2_2_21_1	3792	3194.8	15.8	3600	3573	6.1	30.8
38	2_2_22	3928	3654.5	7.0	3600	3895	0.8	33.9
39	2_2_23	-	3444.6	-	3600	4339	-	47.6
40	2_2_24	-	3656.8	-	3600	4553	-	53.2
41	2_2_25	-	4230.9	-	3600	4708	-	182.1
42	2_2_26	-	4055.0	-	3600	4341	-	90.7
43	2_2_27	-	4177.9	-	3600	4730	-	120.3
44	2_2_28	-	-	-	-	5635	-	113.2
45	2_2_29	-	-	-	-	5168	-	166.2
46	2_2_30	-	-	-	-	5784	-	101.9
47	2_2_31	-	-	-	-	5346	-	178.7
48	2_2_32	-	-	-	-	5454	-	140.6
49	2_2_33	-	-	-	-	5760	-	369.2
50	2_2_34	-	-	-	-	5953	-	132.1

results and time of the ALNS algorithm. The notable point is that in real-world instances, the positions of terminals and depots are the same. The three terminals and three depots are located at Den Bosch, Geel and Roermond. Instances 51 through 55 show that the minimum and maximum objective values are fairly close, indicating a strong consistency in results across multiple runs. The average objective values closely match the minimum objectives, with gaps ranging from 0.8% to 1.5%. Moving to the larger instances, 56 through 60, the range between the minimum and maximum objective values slightly widens, yet the performance remains impressive. The gaps are consistently low, from 0.9% to 1.4%, showcasing ALNS's capability to scale effectively while maintaining close proximity to the best-known solutions. Notably, the CPU times for these instances are significantly higher compared to the medium-scale instances, reflecting the increased computational effort required for larger problems. However, the times are justifiable given the complexity and the size of the problems being addressed.

Table 6.4: Medium- and large-scale real-world instances

Instance	Setting	ALNS				
		Min Obj.	Max Obj.	Average Obj.	Gap	Average CPU time(s)
51	3_3_44_1	3751	3781	3766.2	0.8%	409.9
52	3_3_44_2	3211	3257	3230	1.4%	426.9
53	3_3_44_3	2919	2945	2934.2	0.9%	430.2
54	3_3_44_4	3712	3768	3727.2	1.5%	455.4
55	3_3_44_5	3884	3919	3908.2	0.9%	350.6
56	3_3_94_1	7211	7275	7226.2	0.9%	2715.0
57	3_3_94_2	7154	7244	7179.8	1.2%	2284.9
58	3_3_94_3	7005	7099	7035.8	1.3%	2056.4
59	3_3_94_4	6804	6878	6859.2	1.1%	2546.6
60	3_3_94_5	7688	7796	7758.2	1.4%	2750.8

6.4. Sensitivity analyses

6.4.1. Impact of the location of depots

In this experiment, we demonstrated the impact of different depot locations on the objective function and truck depot visit strategies. To maximize the impact of varying depot locations on the truck visiting strategies and the objective function, we set the empty container stock at all terminals to zero. This approach ensures that trucks are compelled to visit depots more frequently, thereby highlighting the influence of depot positioning on logistical operations.

Based on Figure 6.1, we established three distinct scenarios. The first scenario utilizes the original configuration from Instances 51-60, where three terminals are located in Den Bosch, Geel, and Roermond. Each terminal has a nearby depot represented by red triangles and circles in the figure, with shippers randomly generated within the areas marked by red squares.

In the second scenario, while the terminals remain fixed, the three depots are relocated to the positions marked by black circles. This means that the depots are now further from both the shippers, who continue to appear within the red square areas, and the terminals.

The third scenario also keeps the terminals unchanged; however, the depots are moved closer to the center of the area where shippers are generated, signifying a reduced distance to the shippers but an increased distance from the terminals.

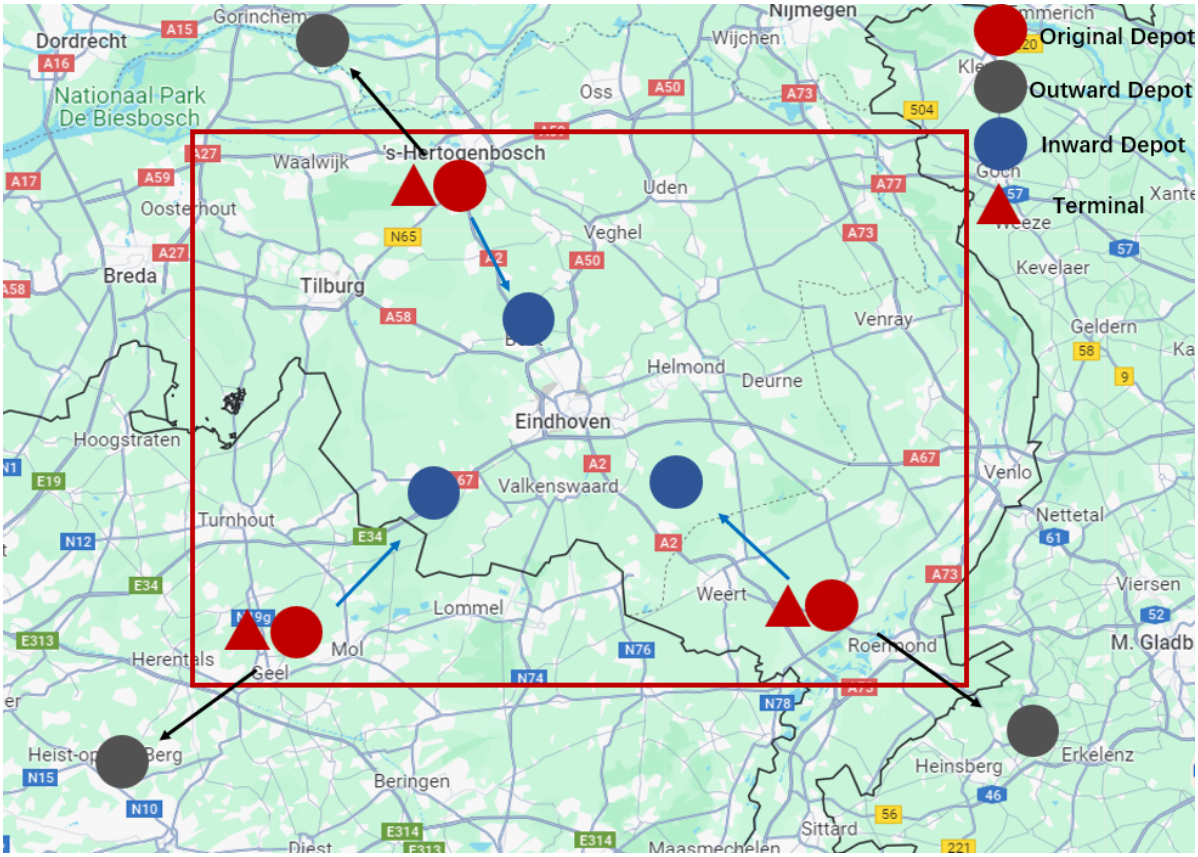

Figure 6.1: Different location of depots

Table 6.5: Impact of the location of depots

Instance	Original depot		Depot moved outward			Depot moved inwards		
	Obj.	Visit empty depot	Obj.	Visit empty depot	Obj. deteri.	Obj.	Visit empty depot	Obj. imp.
51	3741	3	3929	1	+4.8%	3715	3	-0.7%
52	3258	4	3542	3	+8.0%	3207	4	-1.6%
53	2994	3	3238	1	+7.5%	2835	3	-5.3%
54	3821	6	3961	2	+3.5%	3565	4	-6.7%
55	3930	11	4702	9	+16.4%	3853	11	-2.0%

From table 6.5, we derive significant insights regarding the impact of depot location on logistics operations. Comparing the original depot setup with the depot moved outward, it is evident that with terminals devoid of empty container stocks, the demand from shippers needing empty containers is either met through street turns or by visiting depots. However, with the depots now further from both the terminals (the starting points for trucks) and the shippers, the frequency of depot visits decreases. It is worth noting that instance 52 has reduced its access to the depot from 6 times to 2 times. This suggests a preference for utilizing empty containers from other shippers through street turns, with depot visits serving as a secondary option. Consequently, the objective function values decrease by margins ranging from 4.8% to 16.4%. In all scenarios of this experiment, the empty container stock at terminals is set to zero. If terminals had sufficient container stocks, trucks would not need to visit depots to fetch empty containers, which would otherwise lead to a deterioration in the objective function values.

Comparing the original depot setup with the depot moved inward, where depots are farther from the terminals but closer to shippers, we observe from table 6.5 that the number of depot visits remains rel-

atively unchanged, yet there is a slight improvement (from 0.7%-6.7%) in the objective function. This improvement suggests that trucks are choosing more optimal routes, and when operating in central areas close to shippers, they can conveniently access depots for empty containers needed for subsequent deliveries. However, in instance 51, the objective only increased by 0.7%. Therefore, carrier should carefully examine the locations of their high-frequency customers and conduct targeted simulations before deciding whether to relocate depots to save on transportation costs for trucks.

In summary, the positioning of depots affects overall routing efficiency. Ideally, depots should be established near areas with high concentrations of shippers to enhance truck flexibility. The least efficient scenario occurs when depots are located far from both terminals and shippers.

6.4.2. Impact of the initial empty container stock

In this experiment, we repeatedly utilized medium-scale Instances 51-55, modifying the stock scenarios of the instances into two categories: one where terminals had sufficient empty container stock, and another where terminals had no container stock at all. The location of terminals and depots are same with the scenario depots moved outwards in the section 6.4.1. This implies that trucks serving shippers needing empty containers had to either obtain them from depots or engage in street turns to acquire them from other shippers.

As shown in Table 6.6, when the terminal has sufficient inventory, the number of visits to empty depots for instances 51-55 is zero, indicating that trucks do not need to detour to depots to pick up empty containers. However, when the terminal has no empty container stock, the number of visits to empty depots increases to between 1 and 9, and the objective deterioration rate rises from 5.7% to 18.2%. This significantly impacts the total driving time of the trucks.

Table 6.6: Impact of the initial empty container stock in terminal

Instance	Enough stock		Zero stock		
	Obj.	Visit empty depot	Obj.	Visit empty depot	Obj. deteri.
51	3705	0	3929	1	+5.7%
52	3337	0	3542	3	+5.8%
53	3001	0	3238	1	+7.3%
54	3657	0	3961	2	+7.7%
55	3847	0	4702	9	+18.2%

6.4.3. Impact of the empty container street-turn

This experiment utilizes the scenario of *Depot moved inwards* from Section 6.4.1 as the benchmark, where terminals have zero empty container stock. We continue with Instances 51-55, but have eliminated all shippers that originally intended to send empty containers. Instead, these requests are either omitted or converted to receiving an empty container, while ensuring that the number of full containers remains largely unchanged. This approach effectively prohibits the street turn of empty containers among shippers. We aim to observe the impacts resulting from the ban on empty container street-turns.

Table 6.7: No shipper sending empty container

Instance	Basic inward depot instances		No shipper sending empty		
	Obj.	Visit empty depot	Obj.	Visit empty depot	Obj. deteri.
51	3715	3	4018	22	+7.5%
52	3207	4	3402	22	+5.7%
53	2835	3	3189	24	+11.1%
54	3565	4	3843	20	+7.2%
55	3853	11	4157	26	+7.3%

As shown in table 6.7, for all instances, there was a significant increase in the number of times trucks visited depots, with instance 53, for example, experiencing an increase from 3 to 24 visits. Furthermore,

the objective function, which represents the total running time of all trucks, showed a deterioration ranging from 5.7% to 11.1%.

This indicates that prohibiting street turns of empty containers between shippers significantly increases the frequency of depot visits. If the terminals lack sufficient empty container stocks, this could potentially lead to excessive depot visits, possibly causing congestion.

6.5. Discussion

The outcomes from our sensitivity analyses corroborate the intuitive notion that the strategic placement of depots influences the operational dynamics and efficiency of transportation logistics. The experimental results, which highlighted the impact of depot location on the frequency of depot visits and the objective function, underline the importance of considering depot proximity in supply chain planning.

Our experiments demonstrated that when empty container stocks at terminals are zero, the total routing time of trucks deteriorates by an average of 8.9%, and there is a marked increase in the number of depot visits. This aligns with practical observations in the field, where logistical inefficiencies and increased operational costs are often encountered due to extended travel times and increased fuel consumption by trucks needing to reach remote depots. Furthermore, as noted by Fazi et al., 2023, when the distance between depots and terminals is considerable, a reduction in the empty container stock at terminals increases the overall routing time. This observation corroborates our findings presented in section 6.4.1, where even a reduced number of depot visits still resulted in increased routing times by an average 8%.

Additionally, the experiment where depots were moved inward showed minimal change in the number of depot visits but a slight improvement in the objective function by an average 3.3% . This suggests that closer proximity of depots to shippers can lead to more optimal travel routes and potentially faster turnaround times, even if it means a longer distance from the terminals. However, there are individual instances where the improvement is not significant (improved around 0.7%). Therefore, we recommend that carriers conduct separate analyses of the depot locations for specific high-frequency shippers before deciding whether to relocate depots.

The substantial increase in depot visits clearly illustrates the logistical challenges and inefficiencies resulting from the ban on street turns. Specifically, when terminals lack sufficient empty containers and other shippers cannot provide empty containers, trucks are forced to undertake longer journeys to depots to satisfy their empty container requirements. The number of depot visits by trucks increased by 2 to 8 times and the average objective deterioration is around 7.8%. This not only extends operational times and emission of greenhouse gases but can also lead to significant congestion at depots, especially during peak operational periods. Such congestion can disrupt the overall flow of goods and escalate transportation costs.

The study by Fazi et al., 2023 exhibits trends similar to those observed in our study, where they reduced the number of customers releasing empty containers to examine the impact on routing. This approach inherently also results in a reduction of street turn occurrences. These findings highlight the drawbacks of centralizing container management exclusively at depots, which, while simplifying certain aspects of logistics, leads to increased travel times and reduced overall efficiency. Additionally, this approach demands further investments in depot infrastructure and management to handle the increased load and prevent service delays and added expenses. However, these negative consequences should serve as a catalyst for encouraging better communication and cooperation among different transportation service providers. By showcasing the inefficiencies associated with banning street turns, it becomes apparent that there is a critical need for a more integrated approach to managing empty containers. Tompkins, 2022, Chief Operating Officer at Port Technology Services, highlights the significant potential of street turns to enhance efficiency across the shipping industry by addressing issues such as costly empty truck trips and port congestion. He argues that street turns are probably the single most impactful thing we can do to improve these industry-wide challenges. Allowing street turns could foster a cooperative environment where service providers share information about container availability, thus building a more effective and efficient logistical network.(Chen et al., 2022) However, he identifies two major obstacles hindering their widespread adoption. First, different shippers and carriers perceive empty containers as a scarce resource and are reluctant to share these with competitors, impacting the logistics and

collaboration necessary for effective street turn strategies. Second, the adoption faces resistance from truck transporters who foresee a drop in revenue. This resistance is driven by clients preferring to pay only for one-way trips between ports and destinations, rather than covering the costs of round-trip transportation.

According to Tao and Wu, 2021, with regard to all-road transport, the CO2 intensity ranged from 0.520 kgCO2/TEU-km to 0.682 kgCO2/TEU-km. The Port of Rotterdam had a container throughput of 15.3 million TEUs in 2021 (port of rotterdam, 2021). Our experiments indicate that if street turns were fully permitted, the average operating time for trucks would decrease by 7.8%. Assuming that truck operating time is proportional to CO2 emissions, it can be simply calculated that perfect information sharing among carriers for street turns of empty containers would reduce CO2 emissions by approximately 620.6 tons CO2/km to 813.9 tons CO2/km. Rotterdam has set a goal to achieve carbon neutrality by 2050. Therefore, the government is motivated to implement policies or economic measures to encourage cooperation among carriers. This would compensate truck transporters for potential losses and help achieve the carbon neutrality target.

For future research, the model could be expanded to include the decoupling of truck heads and trailers, which reflects more complex operational capabilities. Additionally, the transportation of hazardous materials, which requires drivers to possess specific licenses in certain regions, could be integrated into the model. This inclusion would account for the heterogeneity of the truck fleet, addressing the varying qualifications and capabilities required to handle different cargo types.

7

Conclusion

This thesis investigates a container drayage problem involving terminals, depots, and shippers as the primary entities. Each terminal is equipped with a homogeneous fleet of trucks, which must start and end their operations at their respective terminals. Besides visiting depots and shippers, trucks are also allowed to visit terminals other than their starting points. Each truck is capable of carrying one 40ft container or two 20ft containers and can undertake multiple trips within a single planning horizon. For all terminals, they are equipped to handle full containers and have a limited stock of empty containers that can be replenished as needed. Depots, on the other hand, provide additional empty containers but do not handle full containers. As for the shippers, they each have specific time windows within which the trucks must operate to provide service, and all shippers must be serviced. They can generate requests as shown in table 3.1, which include the sending and receiving of empty or full containers to designated ports. Furthermore, street turns of empty containers between shippers are possible, allowing for more efficient container management and routing.

In summary, this study has developed an model to the Container Drayage Problem, which has evolved significantly from its origins in foundational theories like the Transportation Salesman and Vehicle Routing Problems. Over time, research expanded to explore various dimensions of this problem, including Inland Load Drayage, scenarios involving Single/Multiple Terminals or Depots, Resource Constraints, and complex multi-trip challenges. Whereas earlier studies tended to address these dimensions in isolation, this paper's main contribution is its integrated model that synthesizes all these diverse elements into a comprehensive and adaptable framework. This model not only addresses notable gaps identified in previous scholarly work, as detailed in table 2.1, but also offers a more robust solution that is broadly applicable across different scenarios in container logistics. We address this container drayage problem by implementing a Adaptive Large Neighborhood Search (ALNS) algorithm, as detailed in Algorithm 1. Unlike traditional methods such as CPLEX, which struggle with scalability for larger instances, ALNS excels in managing complex, large-scale optimization tasks. Our model incorporates advanced strategies such as diverse destroy and repair operators that significantly alter the solution landscape, facilitating both extensive exploration and intensive exploitation. Additionally, the algorithm employs simulated annealing as its acceptance criterion, allowing for a dynamic adjustment of the search strategy based on the performance feedback from different neighborhood structures. This adaptive mechanism ensures that ALNS can efficiently navigate through the complex problem space, iteratively refining solutions and enhancing the overall problem-solving process. The detailed mechanics of our ALNS, including the specific operators and flow of the algorithm, are comprehensively outlined in the pseudocode provided in Algorithm 1.

The comprehensive computational experiments conducted in this thesis spanned across small, medium, and large-scale instances, culminating in a series of 10 real-world instances specifically designed to reflect realistic operational scenarios. Our experimental framework was robust, involving a diverse array of settings that encompassed randomly generated and geographically based instances, thus ensuring a broad validation of the Adaptive Large Neighborhood Search (ALNS) algorithm's efficacy. Our sensitivity analyses, detailed in section 6.4.1, investigated the effects of strategic decisions such as the

placement of depots and the availability of empty containers at terminals. The experiments confirmed that optimal depot placement could significantly enhance logistical efficiency, reducing unnecessary travel and operational costs. Moreover, scenarios with depleted empty container stocks at terminals necessitated frequent depot visits. We also experimented with the negative impact on truck routing time if street turns were prohibited. From this, we discussed the benefits of implementing street turns in the real world (reducing CO2 emissions) and the challenges of implementation, and proposed possible solutions. The discussions further solidified these findings, drawing parallels with existing literature and emphasizing the practical implications of our study.

References

Braekers, K., Caris, A., & Janssens, G. (2014). Bi-objective optimization of drayage operations in the service area of intermodal terminals. *Transportation Research Part E: Logistics and Transportation Review*, *65*. https://doi.org/10.1016/j.tre.2013.12.012

Braekers, K., Caris, A., & Janssens, G. K. (2013). Integrated planning of loaded and empty container movements. *OR Spectrum*, *35*(2), 457–478. https://doi.org/10.1007/s00291-012-0284-5

Bruglieri, M., Mancini, S., Peruzzini, R., & Pisacane, O. (2021). The multi-period multi-trip container drayage problem with release and due dates. *Computers & Operations Research*, *125*, 105102. https://doi.org/10.1016/j.cor.2020.105102

Caris, A., & Janssens, G. (2007). Pre- and end-haulage of intermodal container terminals modelled as a full truckload pickup and delivery problem with time windows, 554–559. https://doi.org/10.7148/2007-0554

Caris, A., & Janssens, G. (2010). A deterministic annealing algorithm for the pre- and end-haulage of intermodal container terminals. *Int. J. of Computer Aided Engineering and Technology*, *2*. https://doi.org/10.1504/IJCAET.2010.035390

Chen, R., Chen, S., Cui, H., & Meng, Q. (2021). The container drayage problem for heterogeneous trucks with multiple loads: A revisit. *Transportation Research Part E: Logistics and Transportation Review*, *147*, 102241. https://doi.org/10.1016/j.tre.2021.102241

Chen, R., Meng, Q., & Jia, P. (2022). Container port drayage operations and management: Past and future. *Transportation Research Part E: Logistics and Transportation Review*, *159*, 102633. https://doi.org/https://doi.org/10.1016/j.tre.2022.102633

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints [Publisher: INFORMS]. *Transportation Science*, *46*(3), 297–316. https://doi.org/10.1287/trsc.1110.0400

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

Hiermann, G., Puchinger, J., Ropke, S., & Hartl, R. F. (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, *252*(3), 995–1018.

Imai, A., Nishimura, E., & Current, J. (2007). A lagrangian relaxation-based heuristic for the vehicle routing with full container load. *European Journal of Operational Research*, *176*(1), 87–105. https://doi.org/10.1016/j.ejor.2005.06.044

ISO. (2020). *ISO 668:2020* [ISO]. Retrieved September 21, 2023, from https://www.iso.org/standard/76912.html

Keskin, M., & Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation research part C: emerging technologies*, *65*, 111–127.

Lai, M., Crainic, T. G., Di Francesco, M., & Zuddas, P. (2013). An heuristic search for the routing of heterogeneous trucks with single and double container loads. *Transportation Research Part E: Logistics and Transportation Review*, *56*, 108–118. https://doi.org/10.1016/j.tre.2013.06.001

Lee, H., Pham, H. T., Kim, C., & Lee, K. (2019). A study on emissions from drayage trucks in the port city-focusing on the port of incheon. *Sustainability*, *11*(19). https://doi.org/10.3390/su11195358

Mingozzi, A., Roberti, R., & Toth, P. (2012). An exact algorithm for the multitrip vehicle routing problem [Publisher: INFORMS]. *INFORMS Journal on Computing*. https://doi.org/10.1287/ijoc.1110.0495

Nossack, J., & Pesch, E. (2013). A truck scheduling problem arising in intermodal container transportation. *European Journal of Operational Research*, *230*(3), 666–680. https://doi.org/10.1016/j.ejor.2013.04.042

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., & Tarantilis, C. D. (2017). Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, *263*(3), 737–754. https://doi.org/10.1016/j.ejor.2017.05.035

Pi, L., Pan, Y., & Shi, L. (2006). Nested partitions method for the local pickup and delivery problem [ISSN: 2161-8089]. *2006 IEEE International Conference on Automation Science and Engineering*, 375–380. https://doi.org/10.1109/COASE.2006.326911

port of rotterdam. (2021). Port of rotterdam operated at pre-corona level in 2021.

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.

Sacramento, D., Pisinger, D., & Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, *102*, 289–315.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *International conference on principles and practice of constraint programming*, 417–431.

Shiri, S., & Huynh, N. (2016). Optimization of drayage operations with time-window constraints. *International Journal of Production Economics*, *176*, 7–20. https://doi.org/10.1016/j.ijpe.2016.03.005

Sinclair, M., & Dyk, E. V. (1987). Combined routeing and scheduling for the transportation of containerized cargo. *Journal of the Operational Research Society*, *38*, 487–498.

Sterzik, S., & Kopfer, H. (2013). A tabu search heuristic for the inland container transportation problem. *Computers & Operations Research*, *40*(4), 953–962. https://doi.org/10.1016/j.cor.2012.11.015

Taillard, É. D., Laporte, G., & Gendreau, M. (1996). Vehicle routeing with multiple use of vehicles [Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/jors.1996.133]. *Journal of the Operational Research Society*, *47*(8), 1065–1070. https://doi.org/10.1057/jors.1996.133

Tao, X., & Wu, Q. (2021). Energy consumption and co2 emissions in hinterland container transport. *Journal of Cleaner Production*, *279*, 123394. https://doi.org/https://doi.org/10.1016/j.jclepro.2020.123394

Tjokroamidjojo, D., Kutanoglu, E., & Taylor, G. D. (2004). Advance load information value in transportation logistics under stochastic conditions [Num Pages: 6 Place: Norcross, United States Publisher: Institute of Industrial and Systems Engineers (IISE)]. *IIE Annual Conference. Proceedings*, 1–6. Retrieved September 19, 2023, from https://www.proquest.com/docview/192459395/abstract/5936FCBBA0174086PQ/1

Tompkins, B. (2022). Low-hanging fruit?

Vidović, M., Popović, D., Ratković, B., & Radivojević, G. (2017). Generalized mixed integer and VNS heuristic approach to solving the multisize containers drayage problem. *International Transactions in Operational Research*, *24*(3), 583–614. https://doi.org/10.1111/itor.12264

Vidović, M., Radivojević, G., & Raković, B. (2011). Vehicle routing in containers pickup up and delivery processes. *Procedia - Social and Behavioral Sciences*, *20*, 335–343. https://doi.org/10.1016/j.sbspro.2011.08.039

Wang, X., & Regan, A. C. (2002). Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological*, *36*(2), 97–112. https://doi.org/10.1016/S0965-8564(00)00037-9

Wouda, N. A., & Lan, L. (2023). ALNS: A Python implementation of the adaptive large neighbourhood search metaheuristic. *Journal of Open Source Software*, *8*(81), 5028. https://doi.org/10.21105/joss.05028

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

Zhang, R., Yun, W. Y., & Moon, I. K. (2011). Modeling and optimization of a container drayage problem with resource constraints. *International Journal of Production Economics*, *133*(1), 351–359. https://doi.org/10.1016/j.ijpe.2010.02.005

A

Appendix

A.1. Parameters Settings of 3_2_10 instance

For the first random generation of a network with 15 nodes, known as the 3_2_10 Network, we have recreated the network structure to include 3 terminals, 2 depots, and 10 shippers. Each terminal is allocated 3 trucks, with terminal 0 assigned truck index 0-2, terminal 1 assigned truck index 3-5, and terminal 2 assigned truck index 6-8. Each terminal has an inventory of 5 units for each type of empty container.

The demand of shippers D_i^p is shown as table A.1:

Table A.1: Demand parameter of 3_2_10 network

	E40	E20	S0F40-terminal2	S1F40-terminal0	S2F40-terminal2	S8F20-terminal2	S9F20-terminal1
Terminal 0	5	5	0	-1	0	0	0
Terminal 1	5	5	0	0	0	0	1
Terminal 2	5	5	-1	0	-1	1	0
Depot 0	0	0	0	0	0	0	0
Depot 1	0	0	0	0	0	0	0
Shipper 0	-1	0	1	0	0	0	0
Shipper 1	-1	0	0	1	0	0	0
Shipper 2	-1	0	0	0	1	0	0
Shipper 3	0	1	0	0	0	0	0
Shipper 4	1	0	0	0	0	0	0
Shipper 5	1	0	0	0	0	0	0
Shipper 6	0	1	0	0	0	0	0
Shipper 7	0	1	0	0	0	0	0
Shipper 8	0	1	0	0	0	-1	0
Shipper 9	0	1	0	0	0	0	-1

The time windows of each node A_i and B_i are shown below:

A_i = [0, 0, 0, 0, 0, 653, 497, 49, 282, 255, 673, 152, 417, 672, 254]

B_i = [1440, 1440, 1440, 1440, 1440, 918, 790, 307, 537, 531, 938, 443, 670, 935, 552]

The time distance matrix $C_{i,j}$ is shown below:

[[1000, 1000, 1000, 123, 144, 115, 146, 141, 112, 142, 120, 124, 122, 137, 114],
[1000, 1000, 1000, 121, 139, 111, 119, 114, 104, 111, 127, 145, 140, 113, 115],
[1000, 1000, 1000, 112, 104, 110, 123, 121, 114, 125, 109, 129, 134, 123, 142],
[123, 121, 112, 1000, 106, 110, 138, 128, 135, 118, 129, 128, 119, 106, 122],
[144, 139, 104, 106, 1000, 144, 108, 130, 121, 130, 132, 127, 124, 127, 120],
[115, 111, 110, 110, 144, 1000, 116, 131, 109, 134, 124, 125, 140, 119, 121],
[146, 119, 123, 138, 108, 116, 1000, 131, 115, 112, 100, 141, 119, 131, 124],
[141, 114, 121, 128, 130, 131, 131, 1000, 114, 104, 132, 122, 115, 131, 138],
[112, 104, 114, 135, 121, 109, 115, 114, 1000, 121, 127, 123, 126, 138, 103],
[142, 111, 125, 118, 130, 134, 112, 104, 121, 1000, 122, 144, 138, 118, 105],
[120, 127, 109, 129, 132, 124, 100, 132, 127, 122, 1000, 115, 127, 127, 129],
[124, 145, 129, 128, 127, 125, 141, 122, 123, 144, 115, 1000, 131, 136, 114],
[122, 140, 134, 119, 124, 140, 119, 115, 126, 138, 127, 131, 1000, 133, 129],
[137, 113, 123, 106, 127, 119, 131, 131, 138, 118, 127, 136, 133, 1000, 132],
[114, 115, 142, 122, 120, 121, 124, 138, 103, 105, 129, 114, 129, 132, 1000]]

A.2. Results of 3_2_10 instance

CPLEX finds the optimal objective 1851 of this 3_2_10 network within 89.7 seconds. Three trucks are utilized in this case, which are truck 3, 6 and 7.

The result of $x_{i,j}^{k,r}$ is shown as table A.2:

Table A.2: Solution $x_{i,j}^{k,r}$ of 3_2_10 nodes Network

k	r	i	j	Value
0	0	0	6	1
0	0	2	5	1
0	0	5	0	1
0	0	6	10	1
0	0	10	2	1
0	1	0	11	1
0	1	11	12	1
0	1	12	0	1
3	0	1	8	1
3	0	8	14	1
3	0	14	1	1
6	0	1	13	1
6	0	2	7	1
6	0	7	9	1
6	0	9	1	1
6	0	13	2	1

The result of $y_{i,j,p}^{k,r}$ is shown as table A.3:

Table A.3: Solution $y_{i,j,p}^{k,r}$ of 3_2_10 nodes Network

p	k	r	i	j	Value
0	6	0	6	10	1
0	7	0	7	9	1

0	7	1	5	2	1
1	3	0	1	8	2
1	3	0	8	14	1
1	6	0	2	11	2
1	6	0	11	12	1
1	7	0	3	13	1
2	7	1	2	5	1
3	6	0	0	6	1
4	7	0	2	7	1
5	7	0	13	2	1
6	3	0	14	1	1

The result of $t_i^{k,r}$ is shown as table A.4:

Table A.4: Solution $t_i^{k,r}$ of First 3_2_10 Nodes Network

i	k	r	Value
1	6	0	366
2	0	0	782
5	0	0	918
6	0	0	497
7	6	0	121
8	3	0	282
9	6	0	255
10	0	0	673
11	0	1	152
12	0	1	417
13	6	0	672
14	3	0	385

The transportation paths of the trucks are depicted in Figure A.1, where a total of three trucks were deployed.

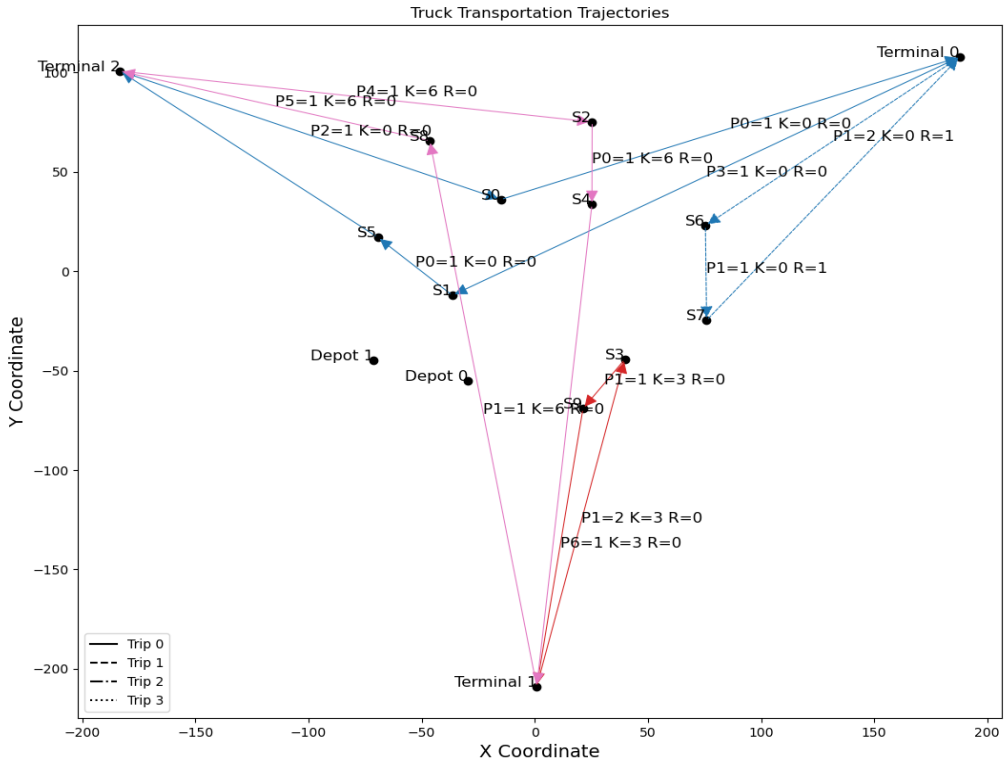

Figure A.1: Truck Transportation Trajectories

The above trajectories and truck container loading and transportation conditions are reasonable, optimal and meet the time window, so test case for the 3_2_10 network passed.