Integrated Vehicle Routing and Dock-Door Scheduling for Outbound Air Cargo Transport Using an Adaptive Large Neighbourhood Search Framework

An Air France KLM Martinair Cargo Case Study

J.K.E. van Kaam

AIRFRANCE KLM Martinair CARGO

TU Delft

Integrated Vehicle Routing and Dock-Door Scheduling for Outbound Air Cargo Transport Using an Adaptive Large Neighbourhood Search Framework

An Air France KLM Martinair Cargo Case Study

by

J.K.E. van Kaam

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday, April 11, 2025.

Student number:	4863968	
Project duration:	May 2024 - March 2025	
Thesis committee:	Dr. O. Sharpanskykh	TU Delft, Chair
	Dr. A. Bombelli	TU Delft, Daily Supervisor
	Dr. D. Zappala	TU Delft, Examiner
	J. den Uijl MSc	KLM, Daily Supervisor

Cover Image: Created by Microsoft's AI image generator.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Acknowledgements

The completion of this thesis project marks the end of my time as a student in Delft. I started my studies in 2018 with a bachelor's programme in Aerospace Engineering at TU Delft, full of uncertainty about what was laying ahead. After obtaining my bachelor's degree, I decided to take a gap year to better decide how I wanted to proceed my studies. This is a decision I have never regretted, as it ultimately led me to choose a master's programme that I truly enjoyed.

I would like to express my sincere gratitude to Jarik den Uijl for the daily supervision at KLM Cargo. Looking back, I can truly say I enjoyed working with you, and I will miss our weekly meetings, both for discussing my thesis challenges and for sharing our weekend stories. From the very first day I entered the office at Schiphol, I felt welcome and part of the Performance Management team at KLM Cargo, for which I am grateful to the entire team.

I would also like to thank Alessandro Bombelli for supervising me at TU Delft. Thank you, for the valuable insights and for the laughs we shared during our meetings. Finally, thank you to my friends from the Aerospace faculty, my teammates from the D.S.B.V. "Punch", and especially my parents and siblings, who supported me through every phase of my studies. Without all your support, this journey would have been far more difficult and certainly less fun.

J.K.E. van Kaam
Delft, March 2025

Contents

Introduction

In Europe, alongside its extensive flight network, KLM operates a trucking network that plays a crucial role in transporting a substantial portion of its cargo. Over the years, this network has grown organically, meaning that adjustments have been made without relying on data-driven insights or forecasts of customer demand. As a result, a number of complications and inefficiencies have arisen in the trucking operations that need to be addressed.

Hub constraints are not always satisfied due to limited handling capacity, leading to waiting times at the hub airport. Additionally, trucks with low load factors are operated in the network, leading to less optimal utilisation of truck capacities. Moreover, connections between inbound trucks and outbound flights are tight, resulting in missed connections and delays in shipment deliveries. On-time delivery is crucial for customers, as it directly impacts customer satisfaction and future bookings.

This thesis report is the result of a collaboration between Delft University of Technology and KLM Cargo and aims to describe the process of designing an integrated vehicle routing and dock-door scheduling model to improve the current operations of KLM Cargo's trucking network. The research presents both an Mixed Integer Linear Programming formulation and an Adaptive Large Neighbourhood Search framework that incorporate the real-world operational conditions of the KLM Cargo trucking network. To the best of the Masters student's knowledge, this specific integration of vehicle routing and dock-door scheduling has not been previously addressed in the literature.

The structure of this thesis report is as follows: Part I contains the relevant Literature Study that supports the research, along with the Research Proposal and Project Plan. Finally, in Part II the Scientific Paper is presented, which includes the models' methodologies, results and conclusions of the thesis project.

I

Literature Study and Research Proposal

Literature Review and Research Proposal

Integrated Vehicle Routing and Dock-Door Scheduling for Outbound Air Cargo Transport

J.K.E. van Kaam

Literature Review and Research Proposal

Integrated Vehicle Routing and Dock-Door Scheduling for Outbound Air Cargo Transport

by

J.K.E. van Kaam

Student Number:	4863968
Supervisor TU Delft:	Dr. A. Bombelli
Supervisor KLM:	J. den Uijl MSc
Institution:	Delft University of Technology
Place:	Faculty of Aerospace Engineering, Delft

Cover Image: KLM Royal Dutch Airlines

TU Delft

Contents

List of Abbreviations

Abbreviation	Definition
3PL	Third-Party Logistics
ABC	Artificial Bee Colony
ACO	Ant Colony Optimisation
AFKL	Air France-KLM
ALNS	Adaptive Large Neighbourhood Search
AMS	Amsterdam Airport Schiphol
BnB	Branch and Bound
BVNS	Basic Variable Neighbourhood Search
COVRPTW	Closed and Open Vehicle Routing Problem with Time Windows
CVRP	Capacitated Vehicle Routing Problem
DARP	Dial-A-Ride Problem
DE	Differential Evolution
FSMVRP	Fleet Size and Mix Vehicle Routing Problem
GA	Genetic Algorithm
GHA	Ground Handling Agent
GRASP	Greedy Randomised Adaptive Search Procedure
GVNS	General Variable Neighbourhood Search
GVRP	Green Vehicle Routing Problem
HVRP	Heterogeneous Vehicle Routing Problem
HVRPCD	Heterogeneous Vehicle Routing Problem with Cross Docking
ILS	Iterated Local Search
MCVRP	Vehicle Routing Problem with Multi-Compartment Vehicles
MDVRP	Multi-Depot Vehicle Routing Problem
MDVRPTW	Multi-Depot Vehicle Routing Problem with Time Windows
MILP	Mixed Integer Linear Programming
OVRP	Open Vehicle Routing Problem
OVRPCD	Open Vehicle Routing Problem with Cross Docking
OVRP-d	Open Vehicle Routing Problem with Driver Nodes
PMSP	Parallel Machine Scheduling Problem
PSO	Particle Swarm Optimisation
PVRP	Periodic Vehicle Routing Problem
RMP	Restricted Master Problem
RVNS	Reduced Variable Neighbourhood Search
RVRP	Rich Vehicle Routing Problem
ROVRP	Reverse Open Vehicle Routing Problem
ROVRPTW	Reverse Open Vehicle Routing Problem with Time Windows
SA	Simulated Annealing
SDVRP	Vehicle Routing problem with Split Delivery
SVRP	Separated Vehicle Routing Problem
TS	Tabu Search
TSP	Traveling Salesman Problem
ULD	Unit Load Device
VND	Variable Neighbourhood Descent
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem
VRPCD	Vehicle Routing Problem with Cross Docking
VRPPD	Vehicle Routing Problem with Pickup and Delivery
VRPTW	Vehicle Routing Problem with Time Windows

1

Introduction

The trucking network of KLM Cargo has grown organically and decisions regarding its development have neither been data-driven nor based on forecasted customer demand. Several complications have emerged within the routing and scheduling processes, which should ideally be improved. Hub constraints are not always satisfied due to limited handling capacity at the dock-doors, and trucks with low load factors are frequently operated, using only a few of the available four cargo positions in a truck. Additionally, tight connections between inbound trucks and outbound flights lead to missed connections and delays in on-time deliveries, which are critical for customer satisfaction and future bookings. Given the dynamic nature of the air cargo industry, KLM Cargo is continuously seeking ways to enhance its operational performance. In collaboration with TU Delft, KLM Cargo has initiated a thesis project aimed at optimizing and scheduling its trucking network, which is responsible for transporting cargo that connects with outbound flights for final delivery.

The objective of this report is to describe the first phase of the thesis project: literature review and research definition. The literature review aims to provide insights into the research conducted on vehicle routing problems, parallel machine scheduling problems, combined vehicle routing and scheduling problems, and both exact and approximate optimisation methods. Additionally, the research definition formulates the research proposal, describing the problem explanation, research objective, and research questions.

The report is structured in two parts. First, Chapter 2 presents the content of the literature study. The second part of the report, Chapter 3, presents the research proposal and project plan. This chapter introduces the problem and describes the relevance of the project. Furthermore, the research question is presented as well as the methodology and project planning of the thesis project.

2

Literature Review

This chapter presents the literature review which aims to provide insights into the research conducted on vehicle routing problems, parallel machine scheduling problems, combined vehicle routing and scheduling problems, and both exact and approximate optimisation methods. Sections 2.1 and 2.2 will cover vehicle routing problems and parallel machine scheduling problems. After that, Section 2.3 will describe the research conducted on combined routing and scheduling problems. Finally, Section 2.4 will delve into the field of exact and approximate optimisation algorithms.

2.1. Vehicle Routing Problem

In 1959, the Vehicle Routing Problem (VRP) was introduced for the first time by Dantzig and Ramser (1959), described as the truck dispatching problem. Dantzig and Ramser opted to design an optimum routing for a homogeneous fleet of gasoline delivery trucks that start and end their trip at a depot. The trucks were used to transport gasoline between a bulk terminal and multiple service stations. The goal was to design a route for each truck to meet the customers' demands while minimising the total distance covered. To solve the problem, they designed a linear programming formulation to find a near-optimal solution (Dantzig & Ramser, 1959). A more recent description of the basic vehicle routing problem is described by Toth and Vigo:

"Determine a set of vehicle routes to perform all (or some) transportation requests with a given vehicle fleet at minimum cost; in particular, decide which vehicle handles which requests in which sequence so that all vehicle routes can be feasibly executed." (Toth & Vigo, 2014)

The vehicle routing problem is a generalisation of the Traveling Salesman Problem (TSP). The assignment related to the TSP is, given a set of nodes, to design a route for a vehicle or salesman that has to visit each node in the network once and only once. The topic of vehicle routing problems is one of the most studied topics within the domain of combinatorial optimisation. Despite its long history of published papers, the field of VRPs remains of great interest to both scientists and practitioners. Advances in technology and knowledge have enabled researchers to bridge the gap between theory and real-world applications, exploring more realistic scenarios, efficient solution methods, and complex VRPs (Mor & Speranza, 2022). Since the publishing of the paper of Dantzig and Ramser, multiple developments and derivatives on the vehicle routing problem have arisen. This includes research on exact algorithms, heuristics and meta-heuristics. Several papers have been published that provide surveys and taxonomies of the vehicle routing problem and its derivatives (Caceres-Cruz et al., 2014; Han & Wang, 2018; Khayya et al., 2024; Koç et al., 2016; Kumar & Panneerselvam, 2012; Lahyani et al., 2015; Laporte, 2009; Mor & Speranza, 2022).

The VRP encompasses numerous variations, with additional constraints required to address complexities beyond its basic form, depending on the specific application. Next, various derivatives of the VRP will be introduced along with a brief explanation of each variant.

Capacitated VRP

The Capacitated Vehicle Routing Problem (CVRP) stands out as the classical variation of the VRP, widely regarded as one of the most extensively researched in the field (Bombelli et al., 2024). In the CVRP, a fleet of vehicles is tasked with deliveries, where all vehicles have a predefined capacity. This capacity can be defined by a number of items, the total weight of shipments, or the dimensions of the packages loaded in the vehicle. Additionally, the CVRP assumes a homogeneous fleet, where all vehicles share identical characteristics, including capacity. (Mor & Speranza, 2022)

The mathematical model of the CVRP, presented by Bombelli et al. (2024), is as follows:

CVRP Mathematical Model (Bombelli et al., 2024)

Sets and indices

$\mathcal{N}$	Set of nodes $i, j \in \mathcal{N}, \lvert\mathcal{N}\rvert = n+2$, 0 is the start and $n+1$ is the end node (depot)
$\mathcal{C}$	Set of customer nodes $i, j \in \mathcal{C}$, excluding the depot nodes 0 and $n+1, \mathcal{C} \subset \mathcal{N}$
$\mathcal{V}$	Set of vehicles $k \in \mathcal{V}$

Parameters

C_{ij}	Cost (or distance or time) from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$
D_i	Demand of node i
Q_k	Capacity of vehicle k

Decision Variables

$x_{ijk} \in \{0,1\}$	Unitary if arc (i, j) is traversed by vehicle k, 0 otherwise
u_{ik}	Order of node $i \in \mathcal{N} \setminus \{n+1\}$ in the tour of vehicle k

$$\min \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} \sum_{k\in\mathcal{V}} C_{ij} x_{ijk} \tag{2.1}$$

subject to:

$$\sum_{k\in\mathcal{V}} \sum_{j\in\mathcal{N}} x_{ijk} = 1 \qquad \forall i \in \mathcal{C} \tag{2.2}$$

$$\sum_{i\in\mathcal{C}} D_i \sum_{j\in\mathcal{N}} x_{ijk} \leq Q_k \qquad \forall k \in \mathcal{V} \tag{2.3}$$

$$\sum_{i\in\mathcal{N}} x_{0ik} = 1 \qquad \forall k \in \mathcal{V} \tag{2.4}$$

$$\sum_{i\in\mathcal{N}} x_{ihk} - \sum_{j\in\mathcal{N}} x_{hjk} = 0 \qquad \forall h \in \mathcal{C}, k \in \mathcal{V} \tag{2.5}$$

$$\sum_{i\in\mathcal{N}} x_{i,n+1,k} = 1 \qquad \forall k \in \mathcal{V} \tag{2.6}$$

$$u_{0k} = 1 \qquad \forall k \in \mathcal{V} \tag{2.7}$$

$$2 \leq u_{ik} \leq n+1 \qquad \forall i \in \mathcal{C}, k \in \mathcal{V} \tag{2.8}$$

$$u_{ik} - u_{jk} + (n+1)x_{ijk} \leq n \qquad \forall i, j \in \mathcal{N} \setminus \{n+1\}, i \neq j, k \in \mathcal{V} \tag{2.9}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i, j \in \mathcal{N}, k \in \mathcal{V} \tag{2.10}$$

$$u_{ik} \geq 0 \qquad \forall i \in \mathcal{N} \setminus \{n+1\}, k \in \mathcal{V} \tag{2.11}$$

The objective function of the model (2.1) minimises the sum of the costs of travelling over the arcs between the nodes in the network. The constraints ensure that:

- each node is visited exactly once (2.2);
- the shipment per vehicle does not exceed the vehicle's capacity (2.3);
- all vehicles start their trips at the depot (2.4);
- each customer node is visited and left by the same vehicle (2.5);
- all vehicles end their trips at the depot (2.6);
- subtours (disjoint routes) are avoided (2.7-2.9);
- the correct domain of the decision variables (2.10 and 2.11)

Note that in this model the fleet can also considered to be heterogeneous as the vehicle capacity (Q_k) can be different per vehicle in the fleet. For the fleet to be homogeneous, the values of Q_k should be the same. Furthermore, when

using the model it is assumed that the demand of each customer fits solely in one vehicle because the model does not allow multiple vehicles to visit the same customer. However, a variation of the CVRP, the split delivery VRP, does allow multiple vehicles to visit the same customer. This type of VRP will be explained later on in this section.

Heterogeneous VRP

In addition to the CVRP, the Heterogeneous VRP (HVRP) considers a fixed fleet of heterogeneous vehicles, which increases the complexity of the model. This means that different types of vehicles are available in the fleet to fulfill the shipping demand. Vehicles with varying capacities and costs are considered in the model for this type of VRP. (Taillard, 1999)
Furtermore, the Fleet Size and Mix VRP (FSMVRP) considers a heterogeneous fleet of vehicles. However, instead of a fixed number of vehicles per vehicle type, an unlimited number of vehicles is available. The number of vehicles used in the network will be determined by the model. (Golden et al., 1984)

VRP with Time Windows

The VRP with Time Windows (VRPTW) was introduced for the first time by Solomon and Desorios, after which many papers have been published presenting different solutions to solve this type of problem (Rabbouch et al., 2018). The VRP variation with time windows adds an extra layer of complexity to the CVRP. The VRPTW takes into account hard or soft time windows depending on the nature of the problem. The VRP variant with hard time windows requires the vehicles to arrive within a specified time interval at the customers' nodes. The model with a soft time interval allows vehicles to violate the time interval constraint, however, a cost penalty is added to the objective function if the time interval constraint is violated (Toth & Vigo, 2014). Furthermore, some models include multiple time windows per customer node (VRP with multiple time windows), meaning the model provides multiple options for a vehicle to arrive at a customer (Caceres-Cruz et al., 2014).

VRP with Pickup and Delivery

In various real-world scenarios, vehicles serve not only to deliver goods to designated customer nodes but also to facilitate the pickup of new goods. Those goods are subsequently transported to another node along the same route or back to the depot. This requires the capability for both loading and unloading goods at customer nodes (Caceres-Cruz et al., 2014). VRPs with Pickup and Delivery (VRPPD) are also known in the literature under the name of Dial-A-Ride problems (DARP). This pertains to situations involving persons or goods that are picked up and dropped off at various locations in the network (Bombelli et al., 2024).

Split Delivery VRP

The basic VRP assumes that the demand of each customer can completely be served by one vehicle and that each customer is only visited once. However, this is not a realistic scenario when looking at real-world applications. In some scenarios, it will not be possible to serve the entire demand of a customer by one vehicle. This is mostly caused by capacity-related limitations. For example, when the weight or dimensions of a shipment exceed a vehicle its capabilities, an extra vehicle will be needed to transport the shipment. This extension of the VRP is covered by the Split Delivery VRP (SDVRP)). (Dror et al., 1994; Mor & Speranza, 2022)

Multi-Depot VRP

The Multi-Depot VRP (MDVRP) deals with networks featuring multiple depots or hubs. In this scenario, vehicles are assigned to start their routes from a particular depot within the network and conclude their journey at the same depot (Rabbouch et al., 2018). Another variant of this problem exists in which vehicles have the option to terminate their routes at a different depot. This type of problem is related to the Open Vehicle Routing Problem (OVRP), which will also be discussed in this section.

Periodic VRP

The Periodic VRP (PVRP) extends traditional route planning to cover multiple days, considering varying frequencies of customer visits. This involves optimising delivery routes over a period, deciding which customers to serve each day and which orders to postpone. The PVRP addresses scenarios such as recurring service or restocking cycles, providing a comprehensive approach to long-term route planning. (Khayya et al., 2024)

Green VRP

Instead of minimising the operational costs or total distance covered, the Green VRP (GVRP) aims to optimise the routing problem in terms of minimising negative environmental effects. For example, the carbon footprint, waste or noise pollution could be chosen to be minimised when planning the routing of the vehicles (Caceres-Cruz et al., 2014).

VRP with Incompatible Products

Another relevant variant of the VRP, is the VRP with incompatible products. This variant entails that some of the to be shipped goods need to be separated and can not be transported using the same vehicle or in the same compartment. For instance, some goods require different temperature levels during transport. This means the goods need to be split and transported using different trucks with different temperature-controlled environments (Lahyani et al., 2015; Z. Wang et al., 2015). Z. Wang et al. (2015) describe a model that considers two types of vehicles: refrigerated and non-refrigerated. The model also categorises products into three types based on their storage requirements: refrigerated, non-refrigerated, and those that can be stored using either option. They developed a mathematical model to optimise truck routing, minimising overall costs while ensuring that products are transported in the appropriate trucks.

Multiple papers explore the VRP with Multi-Compartment Vehicles (MCVRP) (Chen et al., 2019; Coelho & Laporte, 2015; Hübner & Ostermeier, 2019; L. Wang et al., 2020). This approach allows products to be separated in the same vehicle without sharing the same environment. For example, when transporting animals, not all animals can be shipped using the same vehicle or compartment (Oppen & Løkketangen, 2008; Oppen et al., 2010).

Another approach to tackling product incompatibilities discussed in the literature is the Separated VRP (SVRP). Several studies have detailed different models within this category of VRP (Guo et al., 2021; Huang et al., 2019; Veenstra et al., 2018). The SVRP enables the delivery of products needed by distinct customers using dedicated vehicles.

Moreover, multiple papers introduce and address VRPs with (pairwise) incompatibility between goods (Bianchessi et al., 2021; Ceselli et al., 2009; Gendreau et al., 2016; Manerba & Mansini, 2015; Palma-Blanco et al., 2019). Products that are incompatible with one another are not allowed to be placed in the same vehicle. To incorporate this in a mathematical model, for example, an additional set with product incompatibilities or compatibilities can be added. Manerba and Mansini (2015) and Gendreau et al. (2016) both introduce such sets in their mathematical models, along with two constraints that ensure incompatible products cannot be loaded in the same vehicle.

Open VRP

Finally, an important variation, relevant to the trucking network of KLM Cargo, will be discussed. KLM Cargo does not own a fleet of vehicles and outsources its transportation activities. When a company outsources its trucking activities, the company does not have to manage a fleet of vehicles which involves several time consuming tasks such as handling maintenance, insurances and taxes. The company that is outsourcing its transportation activities to a Third Party Logistic (3PL) company acts as a customer and only needs to focus on the trip from the depot to the last customer or from the first customer to the depot (Vincent et al., 2016). This type of routing is covered by the OVRP. The OVRP is different in terms of the final destination of each vehicle in the network. This type of VRP permits vehicles to terminate their trip after serving the last customer node. In contrast to the basic VRP, where vehicles are required to return to the depot where they initiated their route, vehicles in the OVRP do not to return to their home base (Caceres-Cruz et al., 2014). This makes the OVRP relevant for scenarios where companies that do not own a fleet of trucks. Various extensions to the OVRP exist in the literature, most of which are combinations with the VRP types discussed in this section. A few of those OVRP extensions will be briefly discussed in the remainder of this section.

Vincent et al. (2016) developed a mathematical model for a single-product, homogeneous OVRP with cross-docking. The model introduces a dummy depot with zero transportation costs to all nodes to correctly simulate an open route network. They proposed a heuristic approach to solve the model using a simulated annealing algorithm.

Aksen et al. (2007) introduced an OVRP with driver nodes (OVRP-d) and time deadlines. Instead of finishing the route at a customer node, each route has to end at a predefined driver node, which can be the home of the driver or a vehicle parking facility. A mathematical model is presented in the paper as well as a new Tabu Search (TS) algorithm to solve the problem.

Another real-world scenario is the VRPTW with both open and closed routes (COVRPTW). A model for this is presented by Brito et al. (2015), were they propose a mathematical model using fuzzy constraints. Travel times and customer demands are considered to be uncertain, and therefore capacity and time window constraints are modelled using fuzzy constraints (Brito et al., 2015). The paper presents hybrid meta-heuristic algorithm called ACO-GRASP-VNS, which combines Ant Colony Optimisation (ACO), the Greedy Randomised Adaptive Search Procedure (GRASP), and Variable Neighbourhood Search (VNS) to solve the problem with fuzzy constraints.

As explained before, the SDVRP allows multiple vehicles to visit the same customer node. Ruiz y Ruiz et al. (2022) proposed an extension to the OVRP that incorporates split deliveries, known as the OVRP with split deliveries. Their paper presented two mathematical formulations that describe the problem. Furthermore, a cutting-plane algorithm is proposed to improve the performance of one of the formulations.

Furthermore, Eroglu et al. (2014) analysed a real-world scenario for a production company, which they categorised as an OVRP. They introduced a mathematical model for the problem which they classified as a multi-capacity heterogeneous OVRP with split delivery and multi-products. Each vehicle in the model is restricted to visiting two customer nodes at most, meaning that the demand of each customer can be split over a maximum of two vehicles. The authors of the paper present a genetic algorithm with local search to handle large-size problems with more than 50 customer nodes.

In some real-world applications, instead of transporting shipments from a depot to a set of customers, scenarios involve a 3PL collecting shipments from customer nodes to eventually deliver them to a central depot. Schopka and Kopfer (2015) described this variation of the OVRP as the Reverse Open Vehicle Routing Problem (ROVRP). Additionally, their problem incorporates time windows, resulting in a problem called the Reverse Open Vehicle Routing Problem with Time Windows (ROVRPTW). In this scenario, trucks start from specific positions and are routed to the depot while visiting other customer nodes along the way, adhering to operational constraints. A mathematical model is presented in the paper as well as an adaptive large neighbourhood search algorithm to solve the problem and generate a near-optimal solution.

Rich VRP

Since the VRP was first introduced by Dantzig and Ramser, numerous variations have been researched, addressing increasingly complex scenarios. Over the years, the literature has evolved to describe a class of Rich VRPs (RVRPs), which aim to represent the complexities of real-world routing scenarios by incorporating various constraints and specifications. For example, combining several of the above mentioned classes of VRPs. Unlike traditional VRPs, which often focus on idealised models, RVRPs address the challenges encountered in practical situations by integrating factors such as multiple constraints, diverse objectives, and real-world conditions into the routing problem formulation. Lahyani et al. (2015) described the taxonomy of RVRPs and presented a figure outlining the taxonomy of the RVRP. Figure 2.1 shows the figure presented in their paper. The figure illustrates the most commonly found applications in the industry, although not all derivatives of the VRP are shown. Furthermore, as described by the authors, the taxonomy shown is dynamic and should be updated according to developments and challenges in industry (Lahyani et al., 2015).

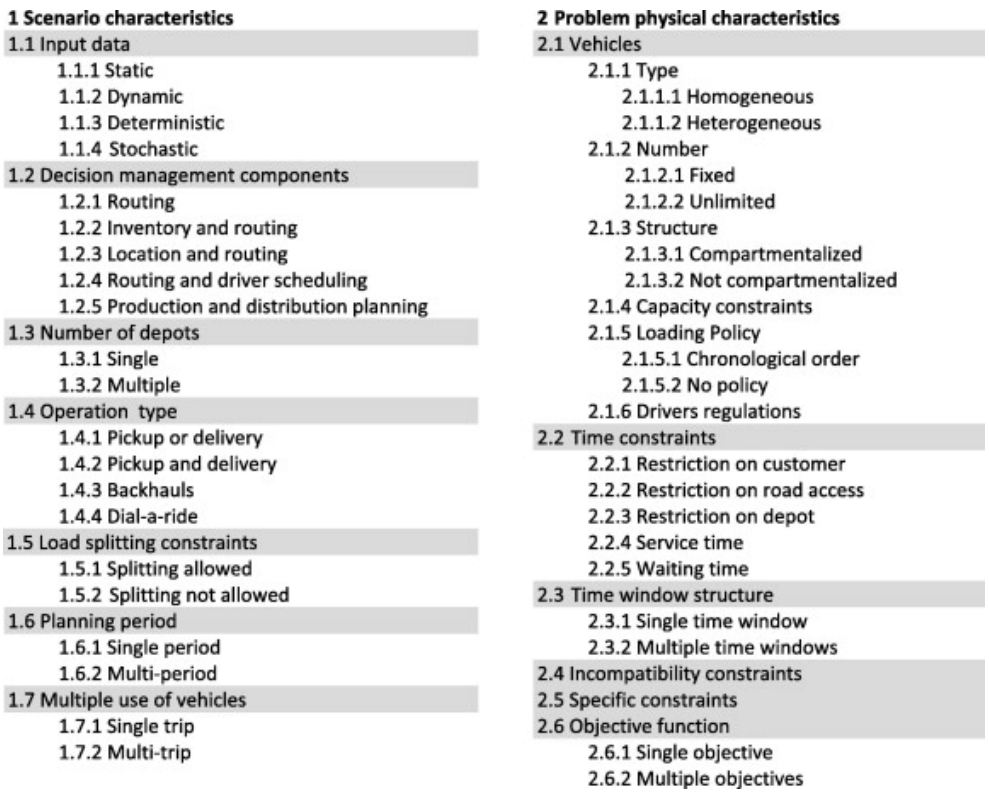


Figure 2.1: RVRP Taxonomy (Lahyani et al., 2015)

2.2. Parallel Machine Scheduling Problem

The Parallel Machine Scheduling Problem (PMSP) involves scheduling a set of predefined tasks across multiple machines. Objectives typically include minimising the total time required to complete all tasks or minimising the total

delay time of all tasks. A simplified variation of the PMSP is a scheduling problem where a task should solely be processed and finished using one machine from the set of available machines. Bombelli et al. (2024) present a mathematical model for this type of PMSP which minimises the total time required to complete all tasks.

PMSP Mathematical Model (Bombelli et al., 2024)

Sets and indices

$\mathcal{N}$	set of jobs $i \in \mathcal{N}$
$\mathcal{S}$	set of job pairs $i, j \in \mathcal{S} \subseteq \mathcal{N} \times \mathcal{N}$ such that $i \prec j$
$\mathcal{M}$	set of machines $m \in \mathcal{M}$

Parameters

P_i	processing time of job $i \in \mathcal{N}$
D_i	deadline of job $i \in \mathcal{N}$

Variables

$t_i \in \mathbb{R}_0$	start time of job i
$c_i \in \mathbb{R}_0$	completion time of job i
$x_{im} \in \{0,1\}$	unitary if job m is assigned to machine m
$y_{ij} \in \{0,1\}$	unitary if job i precedes job j
$c \in \mathbb{R}_0$	latest completion time across all jobs $i \in \mathcal{N}$

$$\min c \tag{2.12}$$

subject to

$$\sum_{m \in \mathcal{M}} x_{im} = 1 \qquad \forall i \in \mathcal{N} \tag{2.13}$$

$$c_i \geq t_i + P_i \qquad \forall i \in \mathcal{N} \tag{2.14}$$

$$c \geq c_i \qquad \forall i \in \mathcal{N} \tag{2.15}$$

$$t_i + P_i \leq t_j + M\left(3 - y_{ij} - x_{im} - x_{jm}\right) \qquad \forall i, j \in \mathcal{N} \setminus \{\delta\} \wedge i < j, m \in \mathcal{M} \tag{2.16}$$

$$t_j + P_j \leq t_i + M y_{ij} \qquad \forall i, j \in \mathcal{N} \setminus \{\delta\} \wedge i < j \tag{2.17}$$

$$t_i + P_i \leq t_j \qquad \forall i, j \in \delta \tag{2.18}$$

$$t_i \leq D_i - P_i \qquad \forall i \in \mathcal{N} \tag{2.19}$$

$$c_i \leq D_i \qquad \forall i \in \mathcal{N} \tag{2.20}$$

$$x_{im} \in \{0,1\} \qquad \forall i \in \mathcal{N}, m \in \mathcal{M} \tag{2.21}$$

$$y_{ij} \in \{0,1\} \qquad \forall i, j \in \mathcal{N} \setminus \{\delta\} \wedge i < j \tag{2.22}$$

$$c \in \mathbb{R}_0 \tag{2.23}$$

The objective function of the model (2.12) minimises total time required to complete all tasks. The constraints of the model ensure that:

- each task is assigned to one and only one machine (2.13);
- determines the time that each task is finished (2.14);
- defines the time the last task is completed (2.15);
- if task i and j are scheduled on the same machine, task j starts after task i is finished and vice versa (2.16 - 2.17);
- tasks with a pre-defined hierarchy start in the right order and do not overlap (2.18);
- tasks are finished before their pre-defined deadlines (2.19);
- the completion time of each task is smaller than the imposed deadline for that task. (2.20)
- the correct domain of the decision variables (2.21-2.23)

This type of problem can also be translated to a truck-to-dock assignment problem. The tasks in the problem are similar to loading and unloading processes of trucks, and the machines are similar to the docks, which are used to load and unload the trucks at a depot. Extensions to this type of problem are widely discussed in the literature (Li et al., 2024). Most variations are about cross-docks. Cross-dock-door assignment problems cover the assignment of both incoming and outgoing goods to trucks and the assignment of trucks to dock-doors at a depot, focusing on optimising the flow of goods through the cross-docking facility.

Furthermore, in parallel machine scheduling, some tasks cannot be processed on every machine. In the literature, this is referred to as machine eligibility. Tasks can only be processed on machines that are in the eligible subset of machines for that task. Eligibility constraints are used to avoid tasks from being processed on machines that are not suitable for that task. (Jiang et al., 2021)

The machine eligibility constraints can also be used for truck-to-dock assignment problems. If machine eligibility is translated to the subject area of truck docking, this means that not all trucks can load and unload at every dock. This depends mostly on the type of cargo that is being transported. For example, air cargo can be shipped using pallets or Unit Load Devices (ULDs). Pallets are loaded and unloaded using forklifts, and ULDs are loaded and unloaded using a special lift at the dock. However, only one of the two options is available at each dock. Depending on how the cargo is transported, different docks are suitable at the hub to load and unload cargo.

Karadgi and Hiremath (2022) presented a Mixed Integer Linear Program (MILP) formulation for job scheduling on parallel machines with both precedence and machine eligibility constraints. Precedence constraints ensure that certain tasks are processed only after other specified tasks have been completed. The model uses several precedence constraint sets to specify which tasks must be finished before others can start, and whether those tasks should be processed on the same machine, on different machines, or on any of the available machines. For the eligibility constraints, a set is created for each task that contains the machines eligible to process that task. Each task should be assigned to one of the eligible machines in their corresponding set of eligible machines. Furthermore, to reduce the computational time of the model, the authors introduced an genetic algorithm.

The same method of implementing the machine eligibility constraints in a MILP model is used in several other papers (Kurt & Çetinkaya, 2024; Maecker et al., 2023). Kurt and Çetinkaya (2024) and Maecker et al. (2023) developed a local search and genetic algorithm, respectively, to reduce computation times. Furthermore, Ik et al. (2023) also implemented machine eligibility constraints in their model but introduced a binary parameter that should equal 1 if a job can be processed on a machine and 0 otherwise.

2.3. Combined Vehicle Routing and Scheduling Problems

The previous sections have provided insights into the research conducted on vehicle routing problems and parallel machine scheduling. This section will address the integration of both vehicle routing problems and scheduling problems, with a particular focus on dock scheduling.

If a limited number of docks is available at a depot, trucks need to be individually scheduled and assigned to a dock-door to prevent waiting times at the depot. Lower waiting times decrease the overall transportation times from the pickup location to the destination of the shipments, which is favourable for the customers requesting the transportation activities. This combination of routing and scheduling problems is described in several papers.

Most papers discuss this VRP variation with Cross-Docking (VRPCD). Cross-docking refers to the process where products from inbound vehicles are unloaded at a cross-docking terminal and directly loaded onto outbound vehicles for delivery to customers. Instead of directly loading products onto outbound vehicles, products can also be temporarily stored at the cross-dock to bridge the time between offloading from the inbound vehicle and loading onto the outbound vehicle. The VRPCD involves planning optimal routes for vehicles or assigning pre-defined routes to vehicles, while also scheduling and assigning vehicles to the available dock-doors. This ensures efficient handling of both inbound and outbound logistics, timely transferring of goods through the cross-dock, minimising transportation costs, and meeting customer demands. In most papers, the model of the VRPCD is separated into three parts: routing of inbound vehicles, scheduling at the docks, and routing of outbound vehicles.

Dondo and Cerdá (2014) formulated a mathematical model that integrates the VRP with dock-door scheduling, while considering a limited number of dock-doors. Dondo and Cerdá (2015) expanded this model by replacing the homogeneous fleet of vehicles with a heterogeneous fleet of vehicles. A sweep heuristics algorithm was used to allocate the customer nodes to the vehicles available when solving the problem. Additionally, Rahbari et al. (2019) and Grangier et al. (2021) proposed other models to solve the VRPCD with a fixed amount of dock-doors available. The papers of Rahbari et al. (2019) and Grangier et al. (2021) described exact methods and a combination of exact methods and

meta-heuristics based on large neighbourhood search, respectively, to solve the problem.

In addition to VRPCDs, several other papers have investigated the combination of vehicle routing and dock-door scheduling. Gromicho et al. (2012) combined vehicle routing and dock-door scheduling, addressing constraints that consider a limited amount dock-doors and strict delivery windows. They employed a decomposition scheme with a heuristic column generation framework, generating columns for a master problem via dynamic programming and solving an integer linear programming model. In their approach, the working period is discretised into fixed time intervals, allowing the evaluation of loading sequences and routing decisions to be separated.

Furthermore, Liang et al. (2023) investigated the VRPTW and Loading Scheduling (VRPTW-LS). They presented a MILP model and an Adaptive Large Neighbourhood Search (ALNS) algorithm, featuring a tailored solution representation and an efficient feasibility check mechanism, to solve the VRPTW-LS.

Moreover, Liao (2020) introduced a new model for the integrated vehicle routing and scheduling problem and proposed a hybrid optimisation method using Iterated Local Search (ILS) and greedy search. The method iteratively solves the vehicle routing and scheduling problem, where in each iteration, the vehicle routing sub-problem is solved first by ILS, followed by a greedy search for the vehicle scheduling sub-problem.

Note that the three above mentioned papers consider a vehicle routing problem where the trucks are loaded at the depot, deliver the products to the customers, and then return to the depot. Liao (2021) instead, proposed an integrated vehicle routing and scheduling problem where vehicles depart from the depot, pick up products at suppliers and deliver them at a cross-dock of which outbound vehicle's routes, dock-doors and shipments have been pre-determined. This is similar to the trucking process at KLM cargo, which can be compared to an inbound open vehicle routing and scheduling problem. To solve the problem, Liao (2021) proposed a cooperative co-evolutionary decomposition-based algorithm. The author presented four types of heuristic methods. The SAABC-HACO method was proven to perform the best in terms of both solution quality and computational time. The method combines Artificial Bee Colony (ABC) with Simulated Annealing (SA) for routing and ACO with local search for scheduling.

2.4. Exact Optimisation Algorithms

Numerous optimisation methods have been studied for solving combinatorial problems. In the literature, optimisation methods are categorised into two groups: exact and approximate optimisation algorithms. Exact optimisation methods aim to find the optimal solution and bring the optimality gap to zero percent, while approximate optimisation methods aim to find a near-optimal solution to the problem. Compared to exact algorithms, approximate algorithms are able to find a solution to the problem within a relatively low time frame. As mentioned before, approximate algorithms do not aim to find an optimal solution but try to find a solution that is near-optimal. Moreover, approximate algorithms are able to solve large-scale problems. This section will discuss the type of exact optimisation algorithms. Next, Section 2.5, will elaborate on approximate algorithms, which includes heuristic and meta-heuristic optimisation algorithms.

Depending on the size of the problem, complexity and computational resources, exact methods are being used to solve optimisation problems. A variety of approaches exist such as Branch and Bound, Branch and Cut, Branch and Price, Column Generation, Dynamical Programming, and Bender's decomposition. Each of these methods will be explained briefly below.

Branch and Bound uses a tree structure to find the optimal solution. The search space is divided in branches, and pruning rules are used to eliminate branches of the search space that will not help to find a better solution to the problem. Gurobi, a well-known commercial solver, employs the branch and bound technique, among others, when solving optimisation problems. Branch and Cut uses a combination of Branch and Bound and cutting planes, where cutting planes are used to strengthen the linear relaxation of the Branch and Bound tree at each node (Lysgaard et al., 2004). A Column Generation algorithm is usually used for problems with a relatively high number of variables. The algorithm starts with solving the Restricted Master Problem (RMP) using a subset of the problem's variables. A pricing problem is then solved to determine which variables should be added to the RMP to improve the solution. This process is repeated until the result of the pricing problem does not show any variables that should be added to improve the solution. Branch and Price combines the bound step of the Branch and Bound approach and the pricing step of the Column Generation approach to efficiently explore the search space and add useful columns to improve the solution. Furthermore, the dynamic programming approach divides optimisation problems into several sub-problems. For each sub-problem, an optimal solution is found, and the solution for the full problem is then built up recursively using the solutions to the sub-problems. Dynamic programming is mostly used for optimisation problems that consist of overlapping sub-problems. By storing the solutions to the sub-problems, each solution needs to be determined only once, which reduces computational time. Lastly, Bender's decomposition technique breaks

down large-scale mixed-integer linear programming problems into a master problem, which handles the integer variables, and a sub-problem, which addresses continuous variables. Through an iterative process, the master problem is solved to provide candidate integer solutions, which are then used in the sub-problem to find continuous solutions. If the sub-problem is infeasible or the solution doesn't satisfy all constraints, Bender's cuts, additional constraints, are generated and added to the master problem to refine it. This iterative refinement continues until an optimal solution is found that satisfies all constraints, making Benders decomposition an efficient technique for solving complex optimisation problems.

2.5. Approximate Optimisation Algorithms

The VRP has been proven to be NP-hard, meaning that the problem cannot be solved in polynomial time (Lenstra & Kan, 1981). Given that the basic VRP is NP-hard, all its variants inherently share this computational complexity. Consequently, exact algorithms are primarily used for smaller-scale instances due to their computational demands, resulting in limited presence in the literature compared to approximate algorithms. For VRPs with large scale instances, approximate algorithms are commonly applied in order to obtain near-optimal solutions within a reasonable amount of time. The available literature on VRP solution methods shows that approximate optimisation techniques are widely used to solve the VRP, its variants and other real-world applications of the VRP (Khayya et al., 2024; Konstantakopoulos et al., 2022; X. Liu et al., 2023).

Approximate algorithms are often divided into two categories: heuristic and meta-heuristic algorithms. Both heuristic and meta-heuristic algorithms aim to find solutions that are close to the optimal solution of the problem. Generally, approximate algorithms are more time-efficient and can determine a solution to the problem relatively quickly compared to exact algorithms. Heuristic algorithms are problem-specific approaches, specific for particularities of a given problem, and may not always be applicable to a broader range of problems. On the other hand, meta-heuristic algorithms are high-level approaches that can be applied to a wide variety of problems. It should be noted that approximate algorithms are not always able to generate feasible solutions, especially as the complexity of real-world problems continues to increase. (Goel & Bansal, 2020)

2.5.1. Heuristics

Heuristics for the VRP are broadly divided into three categories: construction algorithms, two-phase algorithms and improvement algorithms. Each type of heuristic contains specific algorithms. The number of algorithms is extensive as numerous methods have been developed to address the VRP and its variants. Therefore, it is impossible to review every type of heuristic in this literature study. The savings algorithms and sweep algorithms have been selected and will be explained in this section. Those heuristics formed the basis of multiple algorithms that have been developed after its introduction.

Five years after Dantzig and Ramser published their paper about the truck dispatching problem, Clarke and Wright presented a heuristic for the VRP called the savings algorithm. This is one of the first heuristics for the VRP described in the literature and is a construction heuristic. The algorithm first assigns a separate truck to every single node in the network, resulting in an infeasible solution. After that, the algorithm identifies where it can merge two single routes in order to remove one truck from the solution and reduce costs. Then it examines whether an extra single route can be added to the merged route. This process is repeated until the maximum capacity of the vehicle has been reached. Once the truck's capacity is reached, the algorithm continues by attempting to merge two existing routes to further reduce the number of trucks and overall costs. (Clarke & Wright, 1964)

The sweeping algorithm is another simple heuristic which has been developed by Gillett and Miller (1974). The algorithm is known for its two phase method. During the first phase, the algorithm selects a vehicle and starts to assign nodes with the smallest angle relative to its starting point. The process of assigning nodes is repeated until the capacity of the truck is reached. After that, a new truck is selected, and the process of assigning nodes to the truck is repeated. The entire process is finished after all nodes have been assigned to a truck. Lastly, the second phase is initiated and the TSP is solved for each truck. (Gillett & Miller, 1974)

Examples of improvement heuristics are the k-opt algorithm and the λ-interchange algorithm. Improvement heuristics use the an incumbent solution to the problem and try to apply local changes in the neighbourhood to improve the solution. For further reading, Liu et al. (F. Liu et al., 2023) offer a comprehensive survey on heuristics for VRPs. The paper discusses multiple heuristic algorithms as well as the savings method and sweeping method.

2.5.2. Meta-heuristics

Meta-heuristic optimisation is a rapidly evolving field within approximate optimisation algorithms. Meta-heuristics are particularly attractive because the methods can be applied to a wide range optimisation problems, regardless of problem specific characteristics. The literature distinguishes local search methods and population-based search

methods.

Local search methods focus on iteratively improving an incumbent solution by exploring its neighbourhood, and they are well-suited for finding good solutions. However, there is a risk of getting stuck at a local optimum. On the other hand, several meta-heuristics have been developed that are able to escape local optima. For example, algorithms that use procedures to avoid getting trapped in local optima and escape them include SA, Variable Neighbourhood Search (VNS), TS, GRASP, and ILS (Konstantakopoulos et al., 2022).

Population-based search algorithms use multiple solutions to the problem to find a near-optimal solution. By combining and pairing existing solutions, a diverse exploration in the search space is possible which reduces the risk of getting stuck in local optima. Mainly, two classes of population-based search meta-heuristics are described in the literature: swarm intelligence, which includes methods such as Particle Swarm Optimisation (PSO) and ACO, and evolutionary computation, which includes Genetic Algorithms (GA) and Differential Evolution (DE). These classes of population-based search algorithms utilise inspiration from natural processes, such as social behaviour of animals and natural selection, to iteratively improve the solution population towards a near-optimal solution. Some of the local search and population-based search methods will be discussed below.

Simulated Annealing
Simulated Annealing (SA) is an optimisation technique that uses probabilities to escape from local optima when trying to find a global optimum. This technique is inspired by the process of material annealing. In SA, after a new solution to the problem is generated, it is always accepted if it is better than the previous solution. However, if the new solution is worse than the previous one, it is either rejected or accepted based on a probability determined by the acceptance probability function, Equation 2.24. For a minimisation problem, the probability of acceptance is calculated using the following formula:

$$P(\text{accept new solution}) = \begin{cases} e^{-\frac{(f(\text{new})-f(\text{old}))}{T_k}}, & \text{if } f(\text{new}) > f(\text{old}) \\ 1, & \text{if } f(\text{new}) \leq f(\text{old}) \end{cases} \tag{2.24}$$

After determining the probability of acceptance, a random continuous number between 0 and 1 is picked and compared with this probability. If the probability of acceptance is smaller than the random number, the solution is rejected. If it is equal to or larger than the random number, the solution is accepted and used for the next iteration. An important feature of this method is the decreasing temperature over time, which reduces the probability of accepting worse solutions. As the process continues, the model becomes less likely to accept worse solutions as its current solution. This is analogous to the process of material annealing, where the temperature T_k in Equation 2.24 decreases over time. After each iteration, a new temperature is calculated using Equation 2.25. The temperature changes based on the value of α, which can be tuned ($0 < \alpha < 1$) to modify the behaviour of the algorithm. The algorithm aims to eventually find a solution that is near the global optimum. The model ceases execution after a predetermined duration, a specified number of unimproved solutions, or once a certain temperature is reached. (Delahaye et al., 2019)

$$T_k = \alpha T_{k-1} \tag{2.25}$$

The SA algorithm is used and described in numerous papers which try to solve different kinds of VRPs. Mostly, the algorithm is used in combination with other meta-heuristic algorithms, which are called hybrid algorithms. Recently, a paper has been published by Y. Liu et al. (2023) that introduces a hybrid GA-SA algorithm to solve a combination of the SDVRP and VRPPD. Furthermore, J. Zhang et al. (2022) describe in their paper an algorithm to solve a multi-trip time-dependent SDVRP. Their algorithm includes the concept of SA in combination with an intelligent auction mechanism. Vincent et al. (2022) presented a paper that described a SA algorithm for solving a VRPTW with locker delivery.

Genetic Algorithms
Similar to the concept of SA, Genetic Algorithms (GAs) draw inspiration from a natural phenomenon, the biological theory of evolution. Instead of iterating on a single solution, GAs work with a population of multiple solutions. The theory of evolution explains how natural selection operates through the inheritance of chromosomes, where offspring with beneficial traits are more likely to survive and reproduce. Additionally, random genetic mutations occasionally provide beneficial changes that further enhance a species' adaptation over time.

In GAs, two parent solutions are selected from a pool of solutions based on their fitness. The better a solution's fitness, the higher its chance of being chosen as a parent. These parent solutions are then used to form new trial solutions, called children, which inherit features from both parents. Similar to natural processes, child solutions occasionally undergo mutations that introduce new traits. Over successive generations, the algorithm selects the fittest solutions,

gradually evolving towards a near-optimal solution for the given optimisation problem. (Hillier & Lieberman, 2015)

Genetic algorithms continue to be a popular approach for solving VRPs, particularly the more complex rich VRPs. The literature introduces various modifications and hybrid approaches to improve the efficiency and effectiveness of GAs in solving these problems. Given the extensive literature on GAs, only a few recently published papers will be briefly discussed. Olaniyi et al. (2022) presented a modified GA to solve a VRPTW with split delivery. Their work focused on enhancing the reproduction, crossover, and mutation operators. Furthermore, Dubey and Tanksale (2023) introduced a GA integrated with a 3-opt local search algorithm to address a MDVRPTW that includes split pickup and delivery. Their simulations, using benchmark problems from the literature, demonstrated a significant improvement in performance, validating the effectiveness of their approach. Also, Su et al. (2024) proposed a lightweight GA combined with a Variable Neighbourhood Search (VNS) algorithm to solve the MDVRPTW. In this hybrid model, the GA operates as the upper-level optimisation method, while the VNS serves as the local search mechanism to explore different neighbourhood structures and refine the solutions.

Variable Neighbourhood Search
Variable Neighbourhood Search (VNS) was first introduced by Mladenovi and Hansen (1997). The idea behind the method is to systematically explore the solution space by making changes to the solution and moving through different neighbourhood structures. This process involves both descending to local minima and escaping from the valleys of these local optima, with the eventual goal of finding a solution that is near the global optimum, which is not necessarily guaranteed.

As explained, the core concept of VNS is to systematically explore the search space by changing neighbourhood structures to escape local optima and find better solutions. VNS starts with an initial solution and performs local search within a neighbourhood. If no improvement is found, the search is moved to a different neighbourhood. However, if an improved solution is found, the search continues within the same neighbourhood, refining the solution further.

Several types of VNS have been developed which consist of deterministic, stochastic or both stochastic and deterministic phases. Variable Neighbourhood Descent (VND) follows a deterministic approach for neighbourhood changes. It performs a local search in one neighbourhood at a time, only moving to the next neighbourhood once a local minimum is reached in the current neighbourhood. This systematic search ensures that each neighbourhood is thoroughly explored. Reduced VNS (RVNS) uses a stochastic approach to navigate through the search space. It applies a shaking function, which randomly perturbs the current solution without performing any local search after shaking. Futhermore, Basic VNS (BVNS) uses both deterministic and stochastic phases in its method of optimisation. It first uses a stochastic shaking function after which a deterministic local search is being used. Examples of local search are best improvement or first improvement heuristics. Lastly, General VNS (GVNS) is similar to BVNS but uses a VND algorithm for its local search step. (Hansen et al., 2019)

Similar to GAs for VRPs, the literature on VNS for VRPs is as extensive. A few recent papers that describe VNS algorithms for VRPs will be highlighted in this section. Zhen et al. (2024) proposed a tailored VNS algorithm to solve a routing and scheduling task for unmanned aerial vehicles. Their algorithm included a VND and shaking algorithm which makes it a VNS of the type GVNS. They demonstrated its scalability for large scale and instances, similar to the algorithm developed for a VRPTW by Dhahri et al. (2016). Furthermore, Baniamerian et al. (2019) presented a modified VNS with four shaking and two neighbourhood structures, combined with a GA to solve a HVRPCD problem. They compared the GA-MVNS algorithm with their developed SA and ABC algorithms. The results showed that the GA-MVNS algorithm efficiently finds optimal solutions for small-size problems within a reasonable time, and for large-size instances, it outperforms the other algorithms in both computational time and solution quality. Bezerra et al. (2023) proposed Smart General Variable Neighbourhood Search with an Adaptive Local Search (SGVNSALS) algorithm for solving an MDVRPTW. By alternating between local search strategies based on the strategies' success rates, the algorithm outperformed standard VNS methods. This type of algorithm is also known as the ALNS algorithm, first developed by Ropke and Pisinger (2006). The work highlighted the effectiveness of VNS approaches for rich VRPs involving multiple depots and time windows. Also, Y. Wang et al. (2024) addressed an MDVRPTW, focusing on optimising urban logistics networks for sustainability. They developed a model to minimise operating costs, fleet size, and carbon emissions, solving it with a hybrid meta-heuristic that combines clustering, ACO, and VNS. Their approach demonstrated showed a better performance compared to other algorithms, highlighting its potential for solving VRPs of its type (Y. Wang et al., 2024).

Ant Colony Optimisation
The ACO algorithm has been inspired by the natural behaviour of ants. When looking for food sources, ants make use of pheromones to communicate with other ants in their colony. If an ant finds a food source, it returns to its nest and leaves a pheromone trail. The strength of this pheromone trail depends on the quality and quantity of the food source. This type of communication is called stigmergy (Dorigo & Stützle, 2004). Other ants in the colony are more likely to

follow a strong trail of pheromones leading them to the food source that the other ant found. At the same time the trail of pheromones becomes even stronger because of the other ants leaving pheromones on the same trail as well. However, pheromones gradually evaporate over time, preventing ants from being attracted to less optimal paths and reinforcing those paths. Furthermore it allows for the continuous exploration of better routes. This creates a positive feedback loop in the cycle where ants will choose the shorter paths which are rich of pheromones. (Bell & McMullen, 2004)

The ACO algorithm mimics the natural processes of real ant colonies described above. This is done by creating an artificial ant colony that operates on pheromone trails laid along each route in the VRP. The algorithm has a stocastic nature, meaning the path choice of each artificial ant is probabilistic. However, this decision is heavily influenced by the intensity of the pheromone on each path, where stronger pheromone trails represent routes more likely to be chosen by the ants. ACO is considered an agent-based model, since each ant acts independently that follows simple local rules based on pheromone levels in the network. This decentralised decision-making, combined with pheromone updating and evaporation, allows the algorithm to collectively explore and converge on near-optimal routes. The ACO algorithm can be tuned by for example changing the pheromone evaporation rate and the parameters for calculating the probabilities of ants choosing a certain path. (Rizzoli et al., 2004)

Finally, this section about ACO concludes with a brief overview of several recent studies addressing VRPs using ACO. Liao (2021) developed an ACO algorithm for the dock-door scheduling part of an integrated vehicle routing and scheduling problem. ACO is often combined with other meta-heuristic algorithm such as TS and SA (Moghadam et al., 2014; Ren et al., 2023). The paper of Ren et al. (2023) proposed an ACO algorithm for a SDVRP where SA is employed to update the pheromone trails, and TS is used to improve the local search process. Their ACO algorithm showed improvements in model performance when compared to traditional ACO, without the integration of TS and SA. Furthermore, Siddalingappa et al. (2023) combined an ACO algorithm with a graph algorithm to solve a VRPTW, describing the model's performance as comparable to state-of-the-art techniques. Moreover, Fahmy and Gaafar (2023) presented a hybrid ACO algorithm integrated with Local Search techniques (ACO/LS) for a SDVRP to improve exploitation capabilities and convergence speed. Their experiments demonstrated that the ACO/LS algorithm effectively managed the complexities of the SDVRP.

Artificial Bee Colony Optimisation

Similar to ACO, ABC optimisation mimics the behaviour of a group of animals, this time a bee colony. It uses the behaviour of bees searching for nectar sources near their hives. A bee colony consists of three types of bees: employed bees, onlookers and scouts. The employed bees are responsible for gathering information about food sources they found and sharing that with onlooker bees. Onlooker bees use that information to select the food sources to exploit based on their quality. Employed bees transform in scout bees when they abandon a food source and start looking for a new food source near their hive. (Szeto et al., 2011)

The ABC algorithm starts by randomly assigning food sources (solutions) to employed bees. Each bee explores other food sources near its assigned food source, and the current source is replaced if the new solution has a better fitness. After the employed bees have shared this information, onlooker bees select and exploit those food sources based on roulette wheel selection. If a food source does not improve after a set number of iterations of neighbourhood operators, the employed bee abandons it and becomes a scout bee to randomly be assigned to a new food source. The process repeats until a stopping condition, such as a maximum number of iterations or a number of non-consecutive improvements, is met. (Szeto et al., 2011)

Compared to the other meta-heuristics discussed in this section, ABC is used less for solving VRPs. However, a few papers have been published that present ABC algorithms for solving VRPs. Szeto et al. (2011) proposed an enhanced ABC to solve a CVRP. It was shown that this algorithm outperformed the original ABC heuristic, and thus a better alternative to solve the CVRP. Furthermore, D. Zhang et al. (2017) introduced a hybrid meta-heuristic using TS and ABC (TS-ABC), which was proven to be an effective method for solving the VRPTW. Liao (2021) presented a hybrid meta-heuristic that combined SA with ABC (SAABC) to solve the routing problem of an integrated routing and scheduling problem.

Hybrid meta-heuristics

The algorithms mentioned above represent only a fraction of the meta-heuristic methods developed for solving VRPs. However, hybrid meta-heuristics, which combine two or more meta-heuristic algorithms, are increasingly used and often demonstrate better performance compared to individual meta-heuristics. In a literature survey about meta-heuristics for VRPs, Elshaer and Awad (2020) presented a classification tree of meta-heuristic algorithms used for VRPs, which can be found in Figure 2.2. Furthermore, an explanation of how and how often the most widely used meta-heuristic algorithms for VRPs work is provided by the paper of F. Liu et al. (2023). The surveys indicate that VNS and TS are the most frequently utilised local search methods for solving VRPs, while GAs are the predominant choice

among population-based search methods.

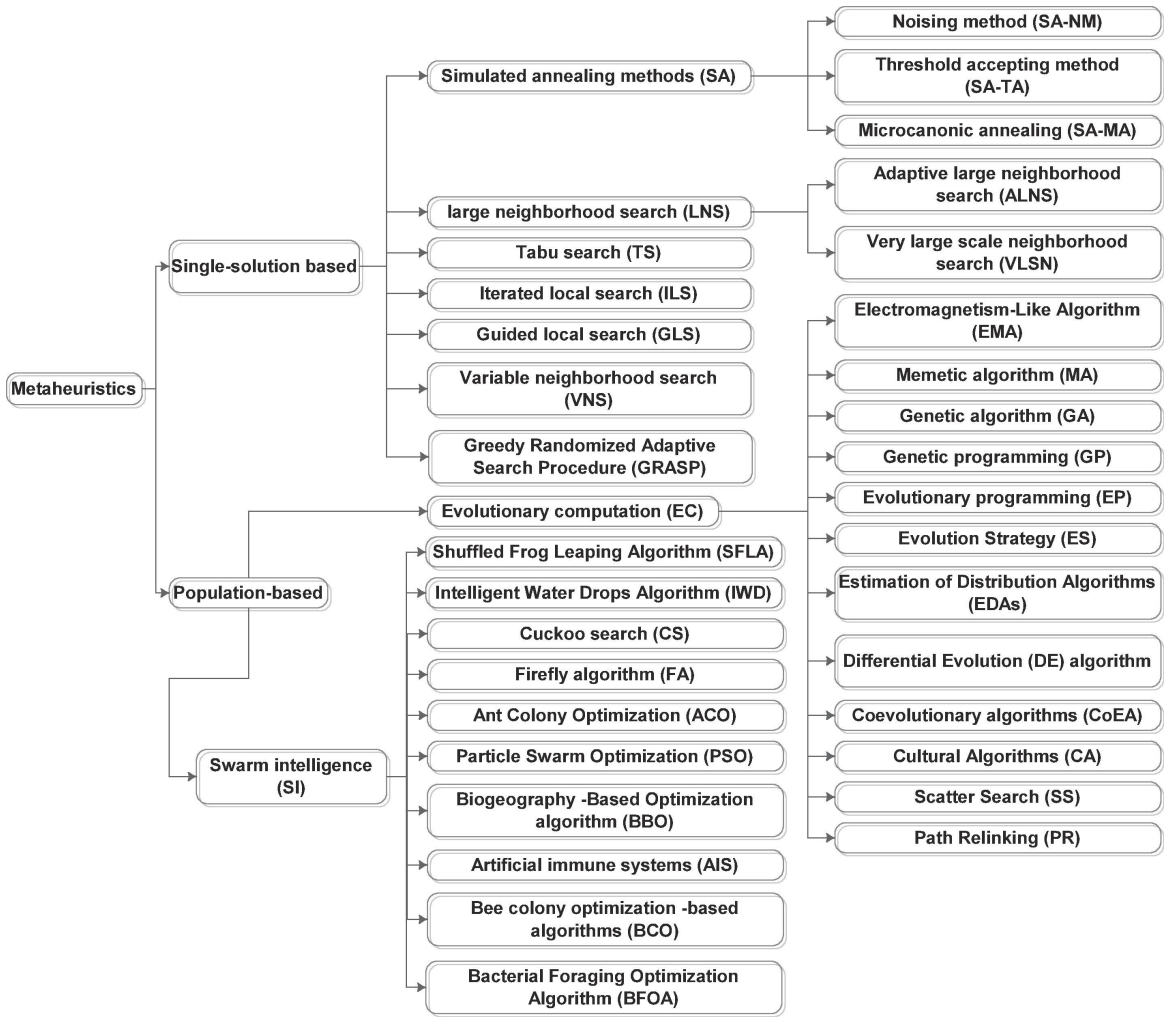


Figure 2.2: Overview of Meta-heuristic Algorithms (Elshaer & Awad, 2020)

3

Research Proposal and Project Plan

Using the findings of the literature study, a research proposal has been formulated. Together with a project planning, this will be presented in this chapter. This chapter is structured as follows: Section 3.1 provides an introduction to the trucking processes at KLM Cargo, explains the relevance of the project, and presents the research objective. Following from the research objective, the research questions will be introduced in Section 3.2. Furthermore, Section 3.3 presents the research methodology that will be used to answer the research questions. Finally, Section 3.4 outlines the planning and timeline of the thesis project.

3.1. Introduction and Relevance of the Project

A substantial amount of cargo that KLM Cargo transports in Europe is tranported using trucks. Customers deliver their shipments to outstations in the KLM Cargo trucking network. At the outstations, Ground Handling Agents (GHA) process the delivered shipments and ensure that the trucks at the outstations are properly loaded. Once loaded, the trucks drive towards the Amsterdam Airport Schiphol (AMS), where the shipments are unloaded and subsequently loaded into cargo aircraft ready to fly to their destinations. Figure 3.1 shows a simple schematic overview of this process.

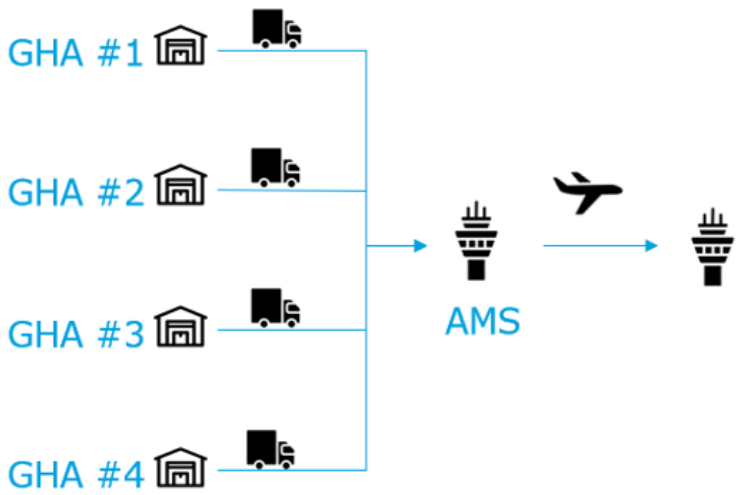


Figure 3.1: Schematic Representation of KLM Cargo's Transport Process

Furthermore, trucks can also make an intermediate stop at one of the outstations on the way to AMS in order to increase the load factor of the truck. Note that this is not visualised in Figure 3.1. KLM Cargo does not operate own trucks in the network but outsources its transportation activities to 3PL companies. KLM Cargo has several contracts with 3PL companies that operate trucks in the network. Figure 3.2 provides an overview of most of the outstations in the European network. The network of KLM Cargo consist of more than [redacted] outstations. In the network, approximately [redacted] rides are carried out per month. For about [redacted]% of the rides, a refrigerated truck is used. Table 3.1 presents the different markets in Europe and their corresponding number of rides and outstations.

Figure 3.2: Outstations in the KLM Cargo Network (confidential)

Table 3.1: KLM Cargo Market Segmentation (confidential)

Customers of KLM Cargo can book an itinerary that consists of one or multiple flight legs, truck legs, or a combination of both. The trucking slot visible for the customer is a virtual truck with unlimited capacity. Once all bookings are received, the planning department starts assigning the bookings to operational trucks. One operational truck usually consists of four ULD positions. An approximation of the capacity of one position is roughly 10 m^3, which means that an average truck can transport about 40 m^3 of cargo. However, there are exceptions on this as on some legs trucks with two or seven positions are available.

Until now, the trucking network of KLM Cargo has grown organically. Furthermore, when changing and developing the network, decisions on this were neither data-driven nor based on forecasted customer demand. Several complications are currently relevant and should ideally be improved. Hub constraints are not always satisfied due to limited handling capacity and dock-doors. Low load factor trucks are operated in the network, meaning that only a few of the positions in a truck are used instead of the, usually, four available positions. Moreover, connections between inbound trucks and outbound flights are tight, resulting in missed connections and a lack of on-time deliveries of shipments. On-time delivery of shipments is important for customers as it influences customer satisfaction as well as future bookings of customers.

This research will look into ways to improve the above mentioned factors within the trucking network of KLM Cargo. A vehicle routing problem and truck to dock-door scheduling problem will be integrated to design a model that is capable of both routing and scheduling trucks in the KLM Cargo network. In addition, the model will take into account special handling constraints for different cargo and truck types, split collection of cargo, a heterogeneous fleet, and dock limitations. Furthermore, an open routing network will be considered when designing the model. To the best of the MSc student's knowledge conducting this research, this specific integrated approach has not been explored in existing literature, making it a novel contribution to the field. Following from this, the research objective has been defined as follows:

Research Objective

Optimising a cargo airline's trucking network by increasing truck load factors, reducing operational costs and ensuring on-time delivery of cargo to the hub, to match with outbound flight legs, by designing a vehicle routing and scheduling model.

3.2. Research Question

After defining the research objective, it is necessary to formulate the research question. The research question helps to yield information that is necessary for accomplishing the research objective that has been described in the previous section. Furthermore, it aids in designing the technical part of the research project. Specifically, the research question has been formulated as follows:

Main Research Question

How can integrating a vehicle routing problem and dock-door scheduling problem be used to design and implement a model that optimises a trucking network for outbound air cargo and does this integration provide operational benefits with respect to the current operations?

To address the main research question, it has been broken down into several sub-questions. These sub-questions will together contribute to answering the main research question. The sub-research questions have been defined as follows:

Sub-Research Questions

- *What are the constraints in the truck routing problem that should be considered for the cargo airline?*
- *What are the constraints in the truck to dock-door scheduling problem that should be considered for the cargo airline?*
- *What optimisation methods can be used to integrate the vehicle routing and scheduling problem?*
- *What modelling decisions should be made to improve the computational time of the model while still producing a near optimal solution?*
- *To what extent does the model improve the current performance of operations of the cargo airline?*
- *To what extent is the solution of the model implementable in the current operations of the cargo airline?*

3.3. Methodology

This section will explain how the research questions will be answered during the research project. Utilising the findings from the literature study, a methodology has been formulated, which will be elaborated upon in this section.

Simultaneously, during the execution of the literature study, a preliminary assessment of the available data in the databases of Air France-KLM (AFKL) was conducted. Data needed for both the routing and scheduling problem was collected. Table 3.2, shows the available data that was found to be useful for the project. The table provides the description, format and storage location of the data sets. Furthermore, it is described how the data is retrieved, the purpose of processing the data, and who has access to the data. Before using the data provided by the company supervisor, it is necessary to address issues with incomplete and missing data. Therefore, data pre-processing and cleaning are required. Additionally, all tables containing data from files with a .pdf extension should be converted to files with a .csv or .xlsx extension to facilitate data analysis and manipulation.

Table 3.2: Available data for the thesis project

Type of data	File Format	How will the data be collected	Purpose of processing	Storage Location	Who will have access to the data
Stations and station coordinates	.csv	Retrieved from database by company supervisor	To visualise the locations of the outstation in Europe	OneDrive	Company supervisor and thesis intern
Trucking contract data	.csv	Retrieved from database by company supervisor	To collect data about the available trucking options including distances, trucking prices per position, transit times, contract duration and name of the trucking company	OneDrive	Company supervisor and thesis intern
Historical data of booking and ops trucks	.csv	Retrieved from database by company supervisor	To collect data about historical shipments including transit times, costs, shipment weight, shipment volume, number of positions and flight connection times	OneDrive	Company supervisor and thesis intern
Segregation rules of incompatible shipments	.pdf	Downloaded from the AFKL SharePoint ->CarGo Documentation Records Management ->AF & KL Cargo Procedures	To collect data about which type of shipments should be taken into account when applying incompatibility constraints	OneDrive & AFKL Sharepoint	Thesis intern (OneDrive) All AFKL employees (Sharepoint)
Special handling codes	.pdf	Downloaded from amerijet.com	To understand the meaning of all special handling codes needed for using the shipment segregation rules	OneDrive	Company supervisor and thesis intern

After all the data is processed, a MILP will be formulated for the routing and scheduling model. The paper by Liao (2021) will serve as the theoretical basis of this research. As described in Section 2.3, the paper proposes a model for an integrated vehicle routing and scheduling problem. The model considers a homogeneous fleet of trucks that start at a cross-dock, visit several suppliers to collect multiple types of goods, and then return to the cross-dock. In this model, the routes, dock-doors, and shipments for outbound trucks have been predetermined. This model will be tailored to reflect real-world conditions which are applicable to the inbound trucking network of the cargo airline. To

achieve this, adjustments will be made to the objective function, and modifications will be required for some of the constraints.

In order to ensure that shipments are separated when they are not allowed to be transported together, it is necessary to add incompatibility constraints to the model. The models of Manerba and Mansini (2015) and Gendreau et al. (2016) will be used as references for incorporating those constraints. These models contain pairwise incompatibility constraints among product types to prevent loading two or more incompatible products into the same truck. Similar to those models, a set of incompatibilities between product types will be introduced into the model, along with a constraint for a combination of each truck and pair of incompatible products that prohibit placing those products together in the same truck.

Furthermore, the model of Liao (2021) addresses a closed routing problem where all trucks start and finish their route at a depot. The model should be adjusted such that the routes in the model are open, which means that each truck can start its route at any node in the network and finishes its route at the depot. Therefore, a dummy depot will be introduced in the model, as demonstrated in the paper of Vincent et al. (2016). This dummy depot has arcs connecting to all nodes in the network with a cost and travel time of zero.

Moreover, the paper of Taillard (1999) will be used to adjust the fleet described in the model by Liao (2021) from a homogeneous fleet to a heterogeneous fleet. Additionally, the model will be adjusted to a routing problem with split delivery, as described by Dror et al. (1994). This will allow multiple trucks to visit the same node in the network instead of the restricted visits of one truck per node. Also, a set of constraints will be designed that sets a time deadline for each truck to arrive at the depot. This deadline is determined by the group of products in the truck with the shortest transfer time to their corresponding outbound flight.

The MILP model will be programmed using Python, and GitHub will be used to control the different versions throughout the development of the model in Python. This can be especially useful if any updates to the model need to be reverted. The Gurobi solver will be used to solve the MILP model programmed in Python. The laptop provided by the company, a Lenovo Thinkpad with a 13th Gen Intel(R) Core(TM) i7-1365U processor running at 1.80 GHz and 32 GB of RAM, will be used for all programming tasks and the execution of the model. The laptop operates on a 64-bit Windows operating system.

After formulating the MILP model, an approximate optimisation algorithm will be developed. As described by Liao (2021), the model in the paper is classified as a NP-hard problem. In the literature, exact algorithms are mostly used for small-scale problems. Therefore, to solve the problem for larger-scale instances, an approximate optimisation algorithm will be developed.

Finally, it will be examined if the models can improve the current state of the network: will truck load factors increase, will the required connection times be satisfied without making transfer times at the cross-dock too long, and will handling constraints at the hub be met? The performance of both the exact and approximate model will be tested and compared.

3.4. Planning

The timeline of the thesis project is divided into four parts: the literature review & research definition, research phase 1, research phase 2, and research dissemination. A Gantt chart has been created to illustrate the work that needs to be done for each of the four parts of the thesis project. The chart shows time estimates for each work package and the dependencies between them. Furthermore, each of the four parts on the timeline includes the milestones and review points for that phase of the thesis project. Breaks and holidays are included in the chart as well. Figure 3.3 shows the Gantt chart for the parts literature review & research definition and research phase 1, and Figure 3.4 shows the Gantt chart for the parts research phase 2 and research dissemination.

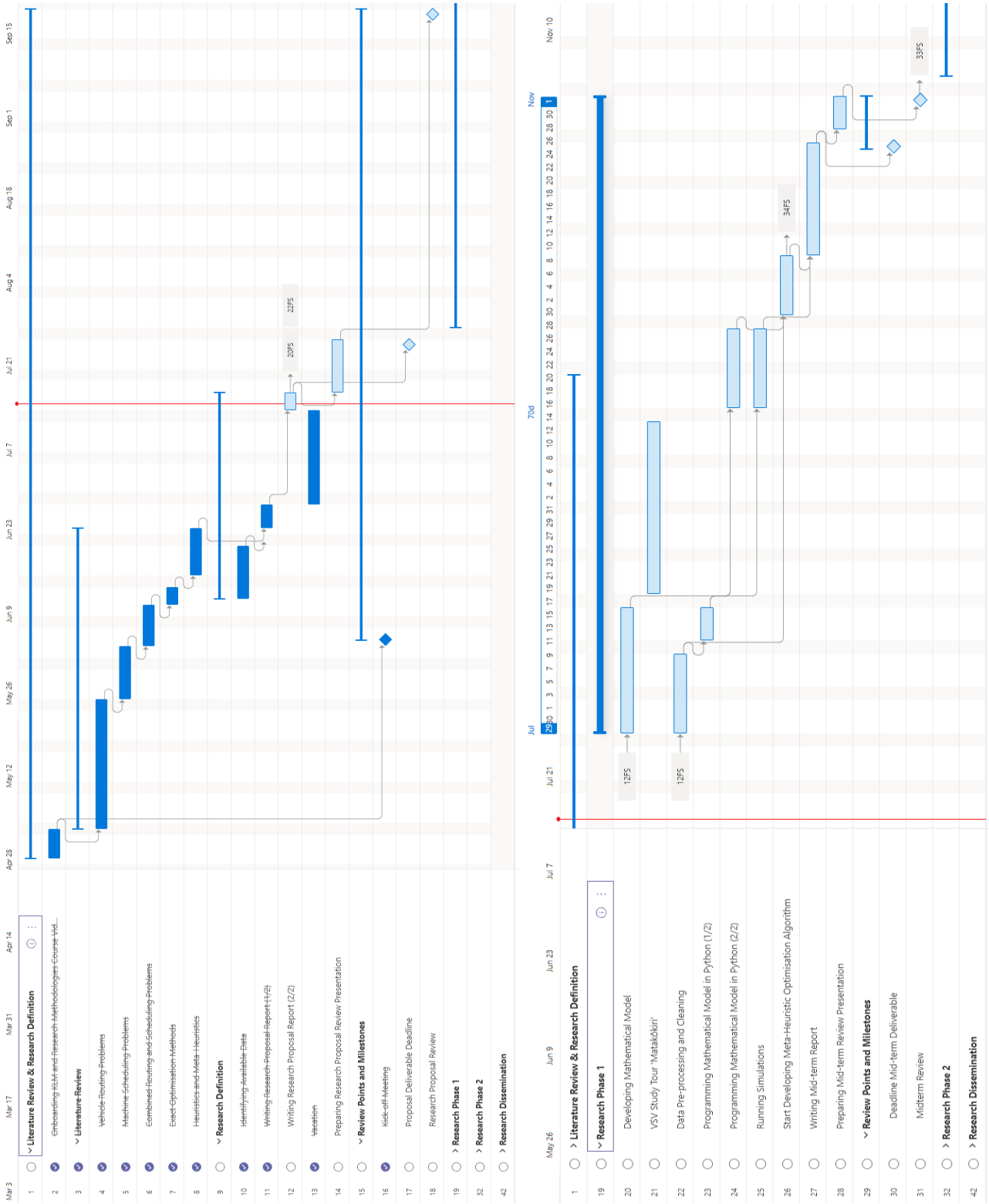


Figure 3.3: Gantt Charts of Literature Review & Research Definition and Research Phase 1

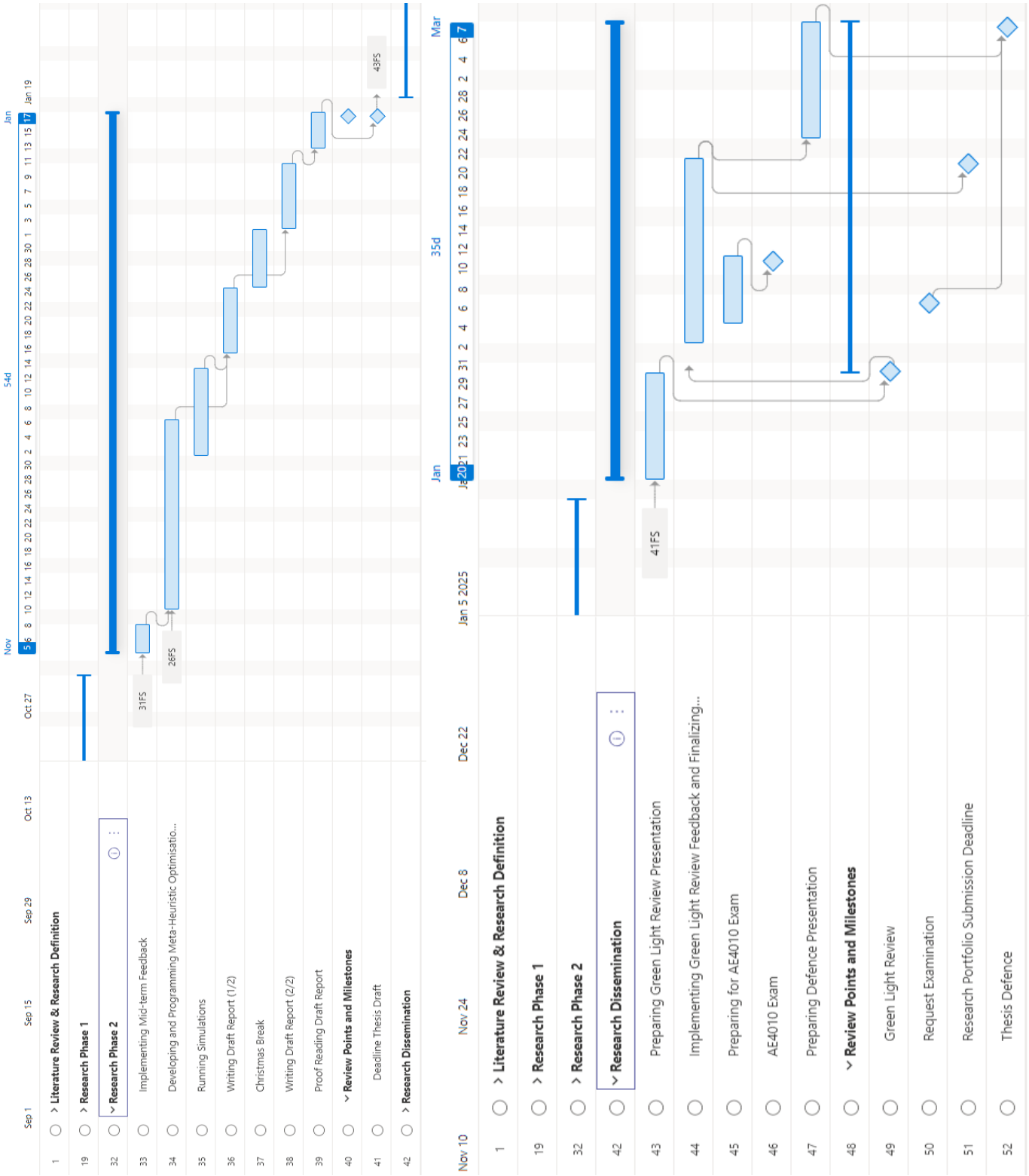


Figure 3.4: Gantt Charts of Research Phase 2 and Research Dissemination

References

Aksen, D., Özyurt, Z., & Aras, N. (2007). Open vehicle routing problem with driver nodes and time deadlines. *Journal of the Operational Research Society, 58*(9), 1223–1234.

Baniamerian, A., Bashiri, M., & Tavakkoli-Moghaddam, R. (2019). Modified variable neighborhood search and genetic algorithm for profitable heterogeneous vehicle routing problem with cross-docking. *Applied Soft Computing, 75*, 441–460.

Bell, J. E., & McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced engineering informatics, 18*(1), 41–48.

Bezerra, S. N., Souza, M. J. F., & de Souza, S. R. (2023). A variable neighborhood search-based algorithm with adaptive local search for the vehicle routing problem with time windows and multi-depots aiming for vehicle fleet reduction. *Computers & Operations Research, 149*, 106016.

Bianchessi, N., Irnich, S., & Tilk, C. (2021). A branch-price-and-cut algorithm for the capacitated multiple vehicle traveling purchaser problem with unitary demand. *Discrete Applied Mathematics, 288*, 152–170.

Bombelli, A., Atasoy, B., Fazi, S., & Boschma, D. (2024). *From theory to application: Learning to optimize with operations research in an interactive way*. TU Delft OPEN.

Brito, J., Martínez, F. J., Moreno, J. A., & Verdegay, J. L. (2015). An aco hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints. *Applied Soft Computing, 32*, 154–163.

Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., & Juan, A. A. (2014). Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR), 47*(2), 1–28.

Ceselli, A., Righini, G., & Salani, M. (2009). A column generation algorithm for a rich vehicle-routing problem. *Transportation Science, 43*(1), 56–69.

Chen, L., Liu, Y., & Langevin, A. (2019). A multi-compartment vehicle routing problem in cold-chain distribution. *Computers & Operations Research, 111*, 58–66.

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

Coelho, L. C., & Laporte, G. (2015). Classification, models and exact algorithms for multi-compartment delivery problems. *European Journal of Operational Research, 242*(3), 854–864.

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

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2019). Simulated annealing: From basics to applications. *Handbook of metaheuristics*, 1–35.

Dhahri, A., Mjirda, A., Zidi, K., & Ghedira, K. (2016). A vns-based heuristic for solving the vehicle routing problem with time windows and vehicle preventive maintenance constraints. *Procedia Computer Science, 80*, 1212–1222.

Dondo, R., & Cerdá, J. (2014). A monolithic approach to vehicle routing and operations scheduling of a cross-dock system with multiple dock doors. *Computers & Chemical Engineering, 63*, 184–205.

Dondo, R., & Cerdá, J. (2015). The heterogeneous vehicle routing and truck scheduling problem in a multi-door cross-dock system. *Computers & Chemical Engineering, 76*, 42–62.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. The MIT Press.

Dror, M., Laporte, G., & Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics, 50*(3), 239–254.

Dubey, N., & Tanksale, A. (2023). A multi-depot vehicle routing problem with time windows, split pickup and split delivery for surplus food recovery and redistribution. *Expert Systems with Applications, 232*, 120807.

Elshaer, R., & Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering, 140*, 106242.

Eroglu, D. Y., Gencosman, B. C., Cavdur, F., & Ozmutlu, H. C. (2014). Introducing the mchf/ovrp/sdmp: Multicapacitated/heterogeneous fleet/open vehicle routing problems with split deliveries and multiproducts. *The Scientific World Journal, 2014*.

Fahmy, S. A., & Gaafar, M. L. (2023). Modelling and solving the split-delivery vehicle routing problem, considering loading constraints and spoilage of commodities. *International Journal of Systems Science: Operations & Logistics, 10*(1), 2074566.

Gendreau, M., Manerba, D., & Mansini, R. (2016). The multi-vehicle traveling purchaser problem with pairwise incompatibility constraints and unitary demands: A branch-and-price approach. *European Journal of Operational Research, 248*(1), 59–71.

Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research, 22*(2), 340–349.

Goel, R. K., & Bansal, S. R. (2020). Hybrid algorithms for rich vehicle routing problems: A survey. In *Smart delivery systems* (pp. 157–184). Elsevier.

Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research*, *11*(1), 49–66.

Grangier, P., Gendreau, M., Lehuédé, F., & Rousseau, L.-M. (2021). The vehicle routing problem with cross-docking and resource constraints. *Journal of Heuristics*, *27*, 31–61.

Gromicho, J., Van Hoorn, J., Schutten, J. M., et al. (2012). Vehicle routing with restricted loading capacities.

Guo, F., Huang, Z., & Huang, W. (2021). Heuristic approaches for a vehicle routing problem with an incompatible loading constraint and splitting deliveries by order. *Computers & Operations Research*, *134*, 105379.

Han, M., & Wang, Y. (2018). A survey for vehicle routing problems and its derivatives. *IOP conference series: materials science and engineering*, *452*, 042024.

Hansen, P., Mladenovi, N., Brimberg, J., & Pérez, J. A. M. (2019). Variable neighborhood search. In M. Gendreau & J.-Y. Potvin (Eds.), *Handbook of metaheuristics* (pp. 57–97, Vol. 3). Springer.

Hillier, F. S., & Lieberman, G. J. (2015). *Introduction to operations research*. McGraw-Hill.

Huang, Z., Huang, W., & Guo, F. (2019). Integrated sustainable planning of self-pickup and door-to-door delivery service with multi-type stations. *Computers & Industrial Engineering*, *135*, 412–425.

Hübner, A., & Ostermeier, M. (2019). A multi-compartment vehicle routing problem with loading and unloading costs. *Transportation Science*, *53*(1), 282–300.

Ik, E. E., Topaloglu Yildiz, S., & atr Akpunar, Ö. (2023). Constraint programming models for the hybrid flow shop scheduling problem and its extensions. *Soft Computing*, *27*(24), 18623–18650.

Jiang, X., Lee, K., & Pinedo, M. L. (2021). Ideal schedules in parallel machine settings. *European Journal of Operational Research*, *290*(2), 422–434.

Karadgi, S., & Hiremath, P. (2022). Job scheduling on parallel machines with precedence constraints using mathematical formulation and genetic algorithm. *International Conference on Robotics, Control, Automation and Artificial Intelligence*, 835–847.

Khayya, E., Medarhri, I., & Zine, R. (2024). A survey of the vehicle routing problem and its variants: Formulations and solutions. *Mathematical Modeling and Computing*, *11*(1), 333–343.

Koç, Ç., Bekta, T., Jabali, O., & Laporte, G. (2016). Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, *249*(1), 1–21.

Konstantakopoulos, G. D., Gayialis, S. P., & Kechagias, E. P. (2022). Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Operational research*, *22*(3), 2033–2062.

Kumar, S. N., & Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants.

Kurt, A., & Çetinkaya, F. C. (2024). Unrelated parallel machine scheduling under machine availability and eligibility constraints to minimize the makespan of non-resumable jobs. *International Journal of Industrial Engineering and Management*, in–press.

Lahyani, R., Khemakhem, M., & Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, *241*(1), 1–14.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation science*, *43*(4), 408–416.

Lenstra, J. K., & Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, *11*(2), 221–227.

Li, M., Hao, J.-K., & Wu, Q. (2024). A flow based formulation and a reinforcement learning based strategic oscillation for cross-dock door assignment. *European Journal of Operational Research*, *312*(2), 473–492.

Liang, Y., Wang, X., Luo, Z., & Zhang, D. (2023). Integrated optimisation of loading schedules and delivery routes. *International Journal of Production Research*, *61*(16), 5354–5371.

Liao, T. W. (2020). Integrated outbound vehicle routing and scheduling problem at a multi-door cross-dock terminal. *IEEE Transactions on Intelligent Transportation Systems*, *22*(9), 5599–5612.

Liao, T. W. (2021). Integrated inbound vehicle routing and scheduling under a fixed outbound schedule at a multi-door cross-dock terminal. *IEEE Transactions on Intelligent Transportation Systems*, *23*(8), 13217–13229.

Liu, F., Lu, C., Gui, L., Zhang, Q., Tong, X., & Yuan, M. (2023). Heuristics for vehicle routing problem: A survey and recent advances. *arXiv preprint arXiv:2303.04147*.

Liu, X., Chen, Y.-L., Por, L. Y., & Ku, C. S. (2023). A systematic literature review of vehicle routing problems with time windows. *Sustainability*, *15*(15), 12004.

Liu, Y., Qin, Z., & Liu, J. (2023). An improved genetic algorithm for the granularity-based split vehicle routing problem with simultaneous delivery and pickup. *Mathematics*, *11*(15), 3328.

Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical programming*, *100*, 423–445.

Maecker, S., Shen, L., & Mönch, L. (2023). Unrelated parallel machine scheduling with eligibility constraints and delivery times to minimize total weighted tardiness. *Computers & Operations Research*, *149*, 105999.

Manerba, D., & Mansini, R. (2015). A branch-and-cut algorithm for the multi-vehicle traveling purchaser problem with pairwise incompatibility constraints. *Networks*, *65*(2), 139–154.

Mladenovi, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, *24*(11), 1097–1100.

Moghadam, S. S., Ghomi, S. F., & Karimi, B. (2014). Vehicle routing scheduling problem with cross docking and split deliveries. *Computers & chemical engineering*, *69*, 98–107.

Mor, A., & Speranza, M. G. (2022). Vehicle routing problems over time: A survey. *Annals of Operations Research*, *314*(1), 255–275.

Olaniyi, O. S., James, A. K., Ibrahim, A. A., & Makanjuola, A. F. (2022). On the application of a modified genetic algorithm for solving vehicle routing problems with time windows and split delivery. *IAENG International Journal of Applied Mathematics*, *52*(1), 1–9.

Oppen, J., & Løkketangen, A. (2008). A tabu search approach for the livestock collection problem. *Computers & Operations Research*, *35*(10), 3213–3229.

Oppen, J., Løkketangen, A., & Desrosiers, J. (2010). Solving a rich vehicle routing and inventory problem using column generation. *Computers & Operations Research*, *37*(7), 1308–1317.

Palma-Blanco, A., González, E. R., & Paternina-Arboleda, C. D. (2019). A two-pheromone trail ant colony system approach for the heterogeneous vehicle routing problem with time windows, multiple products and product incompatibility. *International Conference on Computational Logistics*, 248–264.

Rabbouch, B., Mraihi, R., & Saâdaoui, F. (2018). A recent brief survey for the multi depot heterogenous vehicle routing problem with time windows. *Hybrid Intelligent Systems: 17th International Conference on Hybrid Intelligent Systems (HIS 2017) held in Delhi, India, December 14-16, 2017*, 147–157.

Rahbari, A., Nasiri, M. M., Werner, F., Musavi, M., & Jolai, F. (2019). The vehicle routing and scheduling problem with cross-docking for perishable products under uncertainty: Two robust bi-objective models. *Applied Mathematical Modelling*, *70*, 605–625.

Ren, T., Luo, T., Jia, B., Yang, B., Wang, L., & Xing, L. (2023). Improved ant colony optimization for the vehicle routing problem with split pickup and split delivery. *Swarm and Evolutionary Computation*, *77*, 101228.

Rizzoli, A. E., Oliverio, F., Montemanni, R., & Gambardella, L. M. (2004). Ant colony optimisation for vehicle routing problems: From theory to applications. *Galleria Rassegna Bimestrale Di Cultura*, *9*(1), 1–50.

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.

Ruiz y Ruiz, E., García-Calvillo, I., & Nucamendi-Guillén, S. (2022). Open vehicle routing problem with split deliveries: Mathematical formulations and a cutting-plane method. *Operational Research*, *22*(2), 1017–1037.

Schopka, K., & Kopfer, H. (2015). An adaptive large neighborhood search for the reverse open vehicle routing problem with time windows. In *Logistics management: Contributions of the section logistics of the german academic association for business research, 2015, braunschweig, germany* (pp. 243–257). Springer.

Siddalingappa, P. N., Basavaraj, P., Basavaraj, P., & Gowramma, P. B. (2023). Route optimization via improved ant colony algorithm with graph network. *Int. J. Reconfigurable Embed. Syst*, *12*, 403–413.

Su, Y., Zhang, S., & Zhang, C. (2024). A lightweight genetic algorithm with variable neighborhood search for multi-depot vehicle routing problem with time windows. *Applied Soft Computing*, *161*, 111789.

Szeto, W. Y., Wu, Y., & Ho, S. C. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, *215*(1), 126–135.

Taillard, É. D. (1999). A heuristic column generation method for the heterogeneous fleet vrp. *RAIRO-Operations Research*, *33*(1), 1–14.

Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications*. SIAM.

Veenstra, M., Roodbergen, K. J., Coelho, L. C., & Zhu, S. X. (2018). A simultaneous facility location and vehicle routing problem arising in health care logistics in the netherlands. *European Journal of Operational Research*, *268*(2), 703–715.

Vincent, F. Y., Jewpanya, P., & Redi, A. P. (2016). Open vehicle routing problem with cross-docking. *Computers & Industrial Engineering*, *94*, 6–17.

Vincent, F. Y., Susanto, H., Jodiawan, P., Ho, T.-W., Lin, S.-W., & Huang, Y.-T. (2022). A simulated annealing algorithm for the vehicle routing problem with parcel lockers. *IEEE Access*, *10*, 20764–20782.

Wang, L., Kinable, J., & Van Woensel, T. (2020). The fuel replenishment problem: A split-delivery multi-compartment vehicle routing problem with multiple trips. *Computers & Operations Research*, *118*, 104904.

Wang, Y., Wei, Z., Luo, S., Zhou, J., & Zhen, L. (2024). Collaboration and resource sharing in the multidepot time-dependent vehicle routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, *192*, 103798.

Wang, Z., Li, Y., & Hu, X. (2015). A heuristic approach and a tabu search for the heterogeneous multi-type fleet vehicle routing problem with time windows and an incompatible loading constraint. *Computers & Industrial Engineering*, *89*, 162–176.

Zhang, D., Cai, S., Ye, F., Si, Y.-W., & Nguyen, T. T. (2017). A hybrid algorithm for a vehicle routing problem with realistic constraints. *Information Sciences*, *394*, 167–182.

Zhang, J., Zhu, Y., Li, X., Ming, M., Wang, W., & Wang, T. (2022). Multi-trip time-dependent vehicle routing problem with split delivery. *Mathematics*, *10*(19), 3527.

Zhen, L., Yang, Z., Laporte, G., Yi, W., & Fan, T. (2024). Unmanned aerial vehicle inspection routing and scheduling for engineering management. *Engineering*.

II

Scientific Paper

Integrated Vehicle Routing and Dock-Door Scheduling for Outbound Air Cargo Transport Using an Adaptive Large Neighbourhood Search Framework

J.K.E. van Kaam*

Delft University of Technology, Delft, The Netherlands

Abstract

This study addresses the initial phase of a multi-modal air cargo transport network, where trucks collect shipments from multiple origins and deliver them to the hub airport of an airline. Efficient coordination between ground transport and outbound flights is crucial for optimising truck load factors, reducing operational costs, and ensuring on-time cargo transfers at the hub. Poor synchronisation can cause delays and increased expenses, reducing the efficiency of the entire transport network. This paper presents a novel Mixed Integer Linear Programming (MILP) formulation and an Adaptive Large Neighbourhood Search (ALNS) framework for an integrated vehicle routing and dock-door scheduling problem that includes split delivery, incompatible products, time windows, and open routes, with the objective of minimising operational costs. The ALNS framework uses a dock-door-based route representation along with multiple insertion and removal operators to improve the solution to the problem at hand. A comparative analysis between the MILP and ALNS model shows that the ALNS model consistently outperforms the MILP model in computational efficiency and solution quality for larger and more complex instances. The ALNS model efficiently finds feasible solutions within significantly reduced computational times, making it practical for real-world applications. Moreover, using a case study of an airline, the ALNS-generated network demonstrates improvements in cost efficiency, fleet utilisation, and truck load factors compared to the airline's historical routing data. Despite differences between the actual network data and the model-generated data, stemming from assumptions that create an idealised scenario that does not fully capture the complexities of real-world operations, the ALNS model offers significant enhancements in efficiency for the airline's trucking network.

1 Introduction

Air cargo plays an important role in the world-wide logistics network of cargo. Airline networks help to transport goods across long distances in a fast and efficient way. This is particularly crucial for high-value, time-sensitive products such as electronics, pharmaceuticals, and perishable goods, where on-time delivery and reliability are required. Although the volumetric percentage of cargo transported by air is relatively low, the value of those shipments is significantly high considering the small volume. As the global air cargo demand continues to rise, the importance of air cargo in facilitating global trade becomes even more critical (IATA, 2025).

The transport of air cargo requires precise route and time planning. Effective routing and planning involves determining the optimal transportation method, route, and time schedule for each shipment from its origin to its final destination. In many cases, multiple transport modes and carriers are involved in moving cargo from its origin to its destination. Typically, a customer chooses one carrier that is tasked with complete the transport process. This carrier can either manage the entire process or outsource parts of the transportation activities.

In particular, air cargo is frequently transported through a multi-modal transport network. A network of trucks collects the cargo from pickup locations and delivers it to the airline's airport. This trucking network is integrated with a network of cargo and passenger aircraft that transport the cargo to its destination country. Upon arrival, the cargo can be picked up by the customer or transported by trucks to its final destination.

In addition to planning the routes of the trucks in the network, scheduling their arrivals at the airport is important because customers prioritise on-time delivery, which, together with the shipping costs, influences their choice of shipment carrier. Therefore, the trucking network and flight schedules must be synchronised to ensure on-time arrival at the airport for processing and transfer to the correct flight. Furthermore, the number of truck docking spaces at the airport can be limited, which means careful coordination of truck arrivals is needed to avoid exceeding the dock-door capacity. Integrating both the truck routing and dock-door scheduling steps improves the efficiency of the entire

*MSc Student, Sustainable Air Transport, Faculty of Aerospace Engineering, Delft University of Technology

network (Schiewe & Schöbel, 2022).

This research focuses on designing an integrated vehicle routing and dock-door scheduling model to optimise an airline's trucking network by increasing load factors, reducing operational costs, and ensuring on-time cargo delivery to the hub, to match with outbound flight legs. This paper presents a novel Mixed Integer Linear Programming (MILP) formulation and an Adaptive Large Neighbourhood Search (ALNS) algorithm (Ropke & Pisinger, 2006) for a vehicle routing and dock-door scheduling problem with split delivery, incompatible products, time windows and open routes.

This paper is organised as follows. Section 2 presents the literature review and research gap found. After that, Section 3 describes methodology followed during the research, which includes the problem definition, the mathematical formulation of the MILP, and an explanation of the ALNS algorithm. Furthermore, Section 4 describes the study case scenario which is used to evaluate the performance of both models. Then, the results of the case study scenario are discussed in Section 5. Section 6 concludes this paper with the findings of the research presented and recommendations for future research.

2 Literature Review

Optimising truck routes while simultaneously scheduling those trucks to different dock-doors at their destination hub requires addressing multiple complexities of the optimisation problem. The problem can divided into two subproblems, which are the Vehicle Routing Problem (VRP) and the Dock-door Assignment Problem (DAP). This literature review explores and discusses relevant studies presenting research about VRPs, DAPs and a combination of both problems.

2.1 Vehicle Routing Problem

The VRP was introduced by Dantzig and Ramser in 1959. Dantzig and Ramser opted to design an optimum routing for a homogeneous fleet of gasoline delivery trucks. The trucks were used to transport gasoline between a bulk terminal and multiple service stations. The goal was to design a route for each truck to meet the customers' demands while minimising the total distance covered. To solve the problem, Dantzig and Ramser designed procedure based on a linear programming formulation to find a near-optimal solution (Dantzig & Ramser, 1959). The most basic form of the VRP can be described as follows: "Determine a set of vehicle routes to perform all (or some) transportation requests with a given vehicle fleet at minimum cost; in particular, to decide which vehicle handles which requests in which sequence so that all vehicle routes can be feasibly executed" (Toth & Vigo, 2014).

The VRP has been proven to be NP-hard, which means that the problem cannot be solved in polynomial time (Lenstra & Kan, 1981). Given that the basic VRP is NP-hard, all its variants inherently share this computational complexity. Consequently, exact algorithms are primarily used for smaller-scale instances due to their computational demands, resulting in limited presence in the literature compared to approximate algorithms. For VRPs with large-scale instances, approximate algorithms are commonly applied to obtain near-optimal solutions within a reasonable amount of time compared to the computational time of exact algorithms. The available literature on VRP solution methods shows that approximate optimisation techniques are widely used to solve the VRP, its variants and other real-world applications of the VRP (Khayya et al., 2024; Konstantakopoulos et al., 2022; Liu et al., 2023).

Given the extensive history of VRPs in the literature and the numerous variations of the problem, this review will only focus on the aspects of the VRP closest to this research, which are the Open VRP (OVRP), the VRP with incompatible products and the Split Delivery VRP (SDVRP).

SDVRP

The basic VRP assumes that the demand of each customer can completely be served by one vehicle and that each customer is only visited once. However, this is not a realistic scenario when looking at real-world applications. In some scenarios, it will not be possible to serve the entire demand of a customer by one vehicle. This is mostly caused by limitations of the vehicle capacity. For example, when the weight or volume of a shipment exceeds a vehicle's capacity, an extra vehicle will be needed to transport part of the shipment. This extension of the VRP is covered by the SDVRP.

Dror and Trudeau (1989) introduced this SDVRP for the first time and Archetti et al. (2006) showed the potential for cost savings by splitting customer demands among vehicles. Since the introduction of the SDVRP, several heuristic algorithms have been proposed to solve the SDVRP. Archetti and Speranza provided a survey on SDVRPs in 2012, highlighting the potential savings of split deliveries (Archetti & Speranza, 2012). A more recent survey on SDVRPs is published by Zhang et al. (2022). Local Search (LS), Tabu Search (TS), Variable Neighbourhood Descent (VND), Iterated Local Search and Genetic Algorithms (GAs) are proven to be useful methods to solve the SDVRP (Chen et al., 2017; Zhang et al., 2022). Moreover, Khmelev and Kochetov (2015) showed that their TS outperformed GAs in solving performance. Furthermore, literature shows that the trend towards hybrid heuristic algorithms has grown, and it has been shown that hybrid heuristic algorithms can achieve better solutions than a single heuristic algorithm (Zhang et al., 2022).

Two more recent papers involving the splitting of shipments where published by Gu et al. (2019) and Chen et al. (2017). Gu et al. used an Adaptive Large Neighbourhood Search (ALNS) and used a splitting procedure to make their initial solution feasible such that the capacity constraints of the vehicles are satisfied. It showed that high-quality solutions for large and

medium instances could be obtained by their method (Gu et al., 2019). Chen et al. used an a priori splitting strategy in their method, which means that the shipments are split into smaller pieces in advance. They used the 25/10/5/1 and 20/10/5/1 rules to split shipments in as many large pieces as possible using the quantities defined by the rule used. This splitting procedure allows the use of any existing VRP solver, and Chen et al. showed that their algorithm is faster and often equally effective compared to state-of-the-art algorithms across 82 benchmark instances (Chen et al., 2017). Torkzaban et al. (2024) used this a priori splitting concept and improved it by adapting the splitting rule based on customer locations, customer demand and vehicle capacities. Using this adaptive splitting rule, they showed that their technique outperformed the model with the fixed splitting rule of Chen et al..

VRP with Incompatible Products

The VRP with incompatible products addresses real-world constraints where some of the shipments need to be separated and can not be transported using the same vehicle. For instance, some shipments require different temperature levels during transport. This means the shipments need to be separated and transported using different trucks with different temperature-controlled environments.

Wang et al. (2015) describe a model that considers two types of vehicles: refrigerated and non-refrigerated vehicles. Their model categorises shipments into three types based on their storage requirements: refrigerated, non-refrigerated, and shipments that can be stored using either option. They developed a mathematical model to optimise truck routing, minimising overall costs while ensuring that products are transported in trucks with appropriate environmental control capabilities. Wang et al. claim to be the first to address a VRP with such loading constraints. A ruin-recreate heuristic algorithm, and a threshold tabu search method is presented in their paper to solve the described problem.

Moreover, several papers introduce and address VRPs with pairwise incompatibilities between shipments (Bianchessi et al., 2021; Gendreau et al., 2016; Manerba & Mansini, 2015; Palma-Blanco et al., 2019). Products that are incompatible with one another are not allowed to be placed in the same vehicle. To incorporate this in a mathematical model, an additional set with product incompatibilities or compatibilities can be added.

Manerba and Mansini (2015) and Gendreau et al. (2016) both introduce similar sets in their mathematical models, along with two constraints that ensure incompatible products cannot be loaded in the same vehicle. Manerba and Mansini proposed a branch-and-cut algorithm and also introduced a four-step heuristic to find an initial feasible solution. Their method is found to be a valuable procedure to replace exact approaches for the large instances (Manerba & Mansini, 2015). However, Gendreau et al. propose a branch-and-price algorithm for which the pricing problem is solved by combining two exact methods: a tailored branch-and-cut algorithm, and a labelling algorithm for solving a Resource-Constrained Elementary Shortest Path Problem. They compared their model performance with the model of Manerba and Mansini and showed that it outperforms in both solution quality and computational time (Gendreau et al., 2016). In 2021, Bianchessi et al. proposed a branch-price-and-cut algorithm where the sub-problems of the column generation are solved by a dynamic programming labelling algorithm. They proved it to achieve better results in terms of computational time and solution quality compared to the model of Gendreau et al. (Bianchessi et al., 2021).

Furthermore, Palma-Blanco et al. (2019) use an ant colony optimisation with a two-pheromone trail strategy while handling shipment incompatibility constraints among others. They achieve this by removing shipments from the list of possible loads for a vehicle if that vehicle already contains shipments with which the new possible shipment is incompatible. The same technique is applied to vehicle capacity and routing time constraints to ensure that all constraints are satisfied. This model has been demonstrated to achieve significant solutions for instances of the well-known Solomon instances for VRPs (Palma-Blanco et al., 2019).

OVRP

The OVRP is a variant of the VRP where vehicles are not required to return to the depot after completing their routes, first described by Schrage (1981). The OVRPis particularly suitable for companies that outsource their transportation to Third-Party Logistics (3PL) providers. This eliminates the need to manage a fleet, allowing focus solely on route planning (Vincent et al., 2016). Various extensions to the OVRP have been explored to address diverse real-world scenarios.

Vincent et al. (2016) developed a mathematical model for a single-product, homogeneous OVRP with cross-docking. The model introduces a dummy depot with zero transportation costs to all nodes to correctly simulate open routes. They proposed a Simulated Annealing (SA) algorithm that uses several neighbourhood structures to improve the initial solution. It was shown that it could find optimal solutions for small and medium instances while requiring a shorter computational time than the CPLEX solver. On large instances the SA performed better in terms of computational time and solution quality compared to the CPLEX solver.

Pisinger and Ropke (2007) adopted a similar approach for modelling open routes, setting the travel times and distances to the depot to zero. In their paper, they propose an ALNS algorithm to solve five different variants of the VRP, one of which is the OVRP. The authors argue that the ALNS framework can be applied to a wide range of highly constrained optimisation problems. This is useful for problems with a rich nature of various constraints and specifications.

Furthermore, Eroglu et al. (2014) analysed a real-world scenario for a production company. They introduced a Mixed Integer Programming (MIP) model for

the problem which they classified as a multi-capacity heterogeneous OVRP with split delivery and multi-products. Each vehicle in the model is restricted to visiting two customer nodes at most. The authors present a GA with local search to handle large-size problems with more than 50 customer nodes. GA reaches feasible solutions of problems within the time limit of 7200 seconds, something that could not be achieved with the MIP model.

Contrary to what has been discussed before, some real-world applications involve a 3PL collecting shipments from customer nodes for delivery to a central depot, rather than transporting shipments from a depot to customers. Schopka and Kopfer (2015) described this variation of the OVRP as the Reverse Open Vehicle Routing Problem (ROVRP). Additionally, their problem incorporates time windows, resulting in a problem called the Reverse Open Vehicle Routing Problem with Time Windows (ROVRPTW). In this scenario, trucks start from specific positions and are routed to the depot while visiting other customer nodes along the way, and adhering to operational constraints. A MIP model is presented in the paper as well as an ALNS algorithm to solve the problem and generate a near-optimal solution.

2.2 Dock-door Assignment Problem

The DAP is of similar nature as the Parallel Machine Scheduling Problem (PMSP), which involves scheduling a set of predefined tasks across multiple machines. The PMSP's objective typically includes minimising the total time required to complete all tasks or minimising the total delay time of all tasks. The PMSP can be translated to a DAP. The tasks in the PMSP resemble the loading and unloading processes of trucks, while the machines correspond to dock-doors, which are used for loading and unloading trucks at a depot. Extensions to this type of problem are widely discussed in the literature (Li et al., 2024). Most variations are about cross-docks. Cross-dock Door Assignment Problem (CDAP) covers the assignment of both incoming and outgoing goods and trucks to dock-doors at a depot, focusing on optimising the flow of goods through the cross-docking facility.

Karadgi and Hiremath (2022) presented a Mixed Integer Linear Program (MILP) formulation for job scheduling on parallel machines with both precedence and machine eligibility constraints. Precedence constraints ensure that certain tasks are processed only after other specific tasks have been completed. For the eligibility constraints, which restrict some tasks from being assigned to certain machines, a set is created for each task, containing the machines eligible to handle it. Each task must be assigned to one of the machines in its corresponding eligibility set. Furthermore, to create a model that has lower computational times than the MILP, the authors introduced a GA.

The same method of implementing the machine eligibility constraints in a MILP is used in several other papers. To reduce computation times, Kurt and Çetinkaya (2024) and Maecker et al. (2023) developed a model using local search and GA, respectively.

2.3 Combined Vehicle Routing and Scheduling Problems

The previous sections have provided insights into the research conducted on VRPs and PSMPs. This section will address the integration of both VRPs and DAPs.

If a limited number of dock-doors is available at a depot, trucks need to be individually scheduled and assigned to a dock to prevent waiting times at the depot. Lower waiting times decrease the overall transportation times, from the pickup location to the moment of completing unloading at the destination, which is favourable for the customers requesting the transportation activities. This combination of routing and scheduling problems is described in several papers.

Most papers discuss this VRP variation with Cross-Docking (VRPCD). Cross-docking refers to the process where shipments from inbound vehicles are unloaded at a cross-docking terminal and loaded into outbound vehicles for delivery to customers. Instead of directly loading shipments into outbound vehicles, shipments can also be temporarily stored at the cross-dock to bridge the time between offloading from the inbound vehicle and loading into the outbound vehicle. The VRPCD involves planning optimal routes for vehicles or assigning pre-defined routes to vehicles, while also scheduling and assigning vehicles to the available dock-doors. This ensures efficient handling of both inbound and outbound logistics, transferring goods through the cross-dock on time, minimising transportation costs, and meeting customer demands. In most papers, the model of the VRPCD is separated into three parts: routing of inbound vehicles, scheduling vehicles at the dock-doors, and routing of outbound vehicles.

Dondo and Cerdá (2014) formulated a mathematical model that integrates the VRP with cross-dock scheduling, while considering a limited number of dock-doors. Dondo and Cerdá (2015) expanded this model by replacing the homogeneous fleet of vehicles with a heterogeneous fleet of vehicles. A sweep heuristics algorithm was used to allocate the customer nodes to the vehicles available when solving the problem. Additionally, Rahbari et al. (2019) and Grangier et al. (2021) proposed other models to solve the VRPCD with a fixed amount of dock-doors available. The papers of Rahbari et al. and Grangier et al. describe exact methods and a combination of exact methods and meta-heuristics using large neighbourhood search, respectively.

Several other papers describe the integration of vehicle routing and loading dock scheduling. Compared to VRPCDs, shipments do not have to be moved from incoming to outgoing vehicles and only need to be picked up from a depot. Gromicho et al. (2012) combined vehicle routing and loading dock scheduling, addressing constraints that consider limited dock-doors and strict delivery windows. They employed a decomposition scheme with a column generation framework,

generating columns for a master problem via dynamic programming and solving an integer linear program. In their approach, the working period is discretised into fixed time intervals, allowing the evaluation of loading sequences and routing decisions to be separated.

Furthermore, Liang et al. (2023) investigated the VRP with Time Windows and Loading Scheduling (VRPTW-LS). They presented a MILP model and an ALNS algorithm, featuring a tailored solution representation and an efficient feasibility check mechanism, to solve the VRPTW-LS. In the ALNS a dock-based route representation is used that allows for an efficient feasibility check. The representation captures the loading sequence for a fleet of vehicles at the dock and specifies the routing sequence for each vehicle.

Moreover, Liao (2020) introduced a model for an integrated vehicle routing and scheduling problem and proposed a hybrid optimisation method using Iterated Local Search (ILS) and Greedy Search (GS). The method iteratively solves the vehicle routing and scheduling problem, where in each iteration, the vehicle routing sub-problem is firstly solved by an ILS, followed by a GS for the vehicle scheduling sub-problem.

Note that the three above mentioned papers about vehicle routing and loading dock scheduling problems consider a vehicle routing problem where the trucks are loaded at the depot, deliver the products to the customers, and then return to the depot. Liao (2021) instead, proposed an integrated vehicle routing and scheduling problem where vehicles depart from the cross-dock, pickup shipments at suppliers, and deliver them at a cross-dock. The outbound vehicle's routes, dock-doors and shipments have been pre-determined in his model. To solve the problem, Liao (2021) proposed MIP formulation and a cooperative co-evolutionary decomposition-based algorithm. The author presented four types of heuristic methods. The method that combines Artificial Bee Colony (ABC) with SA for routing and Ant Colony Optimisation (ACO) with local search for scheduling, SAABC-HACO in short, was proven to perform the best in terms of both solution quality and computational time.

2.4 Research Gap

Optimising truck routes while simultaneously scheduling those trucks to different dock-doors at their destination hub involves multiple layers of complexity in the optimisation problem. This problem can be broken down into two sub problems: the VRP and the DAP. The VRP focuses on finding the most efficient routing solution for a fleet of trucks, while the DAP focuses the optimal scheduling of trucks to the available dock-doors. The integration of both problems introduces a further layer of complexity, as it is necessary to simultaneously optimise the routes of the trucks and the scheduling of their arrival at the dock-doors.

The literature on these problems is extensive, as discussed in the preceding sections. Studies have explored various variations of the VRP, including the basic VRP, SDVRP, VRP with Incompatible Products, and OVRP, alongside various approaches to the DAP. However, despite the wide range of studies, there remains a gap in research specifically combining the VRP with dock-door scheduling in the presence of VRP variations: split delivery, open routes, time windows and deadlines, and incompatible products. The interaction of these multiple variations make the problem challenging, as it requires an integration of both vehicle routing and scheduling decisions, with the goal of finding optimal or near-optimal solutions in a computationally efficient manner.

In this work, a model is developed for solving the integrated Vehicle Routing and Dock-door Scheduling Problem with the following key features:

- Split Delivery: Accommodating scenarios where demands exceed vehicle capacities, thus requiring multiple deliveries by different vehicles.
- Open Routes: Considering the case where vehicles do not need to return to the depot after completing their delivery routes or do not need to start at the depot before visiting the first location of their route.
- Time Windows: The vehicle routing and dock-door scheduling problem incorporates time windows that impose specific constraints on both vehicles and shipments. Each shipment has a fixed delivery deadline at the depot, and a defined availability time, specifying when the shipment is ready for pickup at its origin. Additionally, pickup locations in the network have closing hours during which vehicles are not allowed to visit.
- Incompatible Products: Some products cannot be transported together on the same vehicle due to incompatibilities. Furthermore, certain products require refrigerated environments during transport, while others do not.
- Limited Dock-doors: The depot has a limited number of dock-doors available for vehicle unloading activities. This restriction means that only a maximum number of vehicles can visit the depot simultaneously.

By developing both a MILP formulation and ALNS framework, this research aims to find a method that can produce near-optimal solutions to this integrated problem. The model objective is to minimise transportation costs which are a function of leg costs, truck cooling costs, and truck load factor costs. The MILP formulation will be useful for smaller problem instances as the problem itself is NP-hard. On the other hand, the heuristic method will be tailored for large complex real-world instances that will be able to provide balance between solution quality and computational efficiency.

3 Methodology

This section outlines the methodology followed in developing the MILP formulation and ALNS framework used to solve the problem. Section 3.1 will present the problem definition and the general assumptions made prior to the development of the models. Following that, Section 3.2 will provide an explanation of the MILP formulation, and Section 3.3 will describe each step of the ALNS framework. Finally, the differences between the two models will be discussed in Section 3.4.

3.1 Problem Definition and General Assumptions

The vehicle routing and dock-door scheduling problem with split delivery, incompatible products, time windows and open routes is an integration of a VRP and DAP. The VRP aims to determine an optimal set of routes for a heterogeneous fleet of trucks, ensuring that all transportation requests from the stations to the hub in the network are fulfilled while adhering to all operational constraints. The DAP seeks to find a feasible docking schedule for the trucks to unload all shipments. These problems are combined to determine a near-optimal set of routes for a given fleet of trucks, ensuring that all transportation requests are fulfilled. The solution should also ensure that trucks do not have to wait before they can begin unloading at the hub's dock-doors.

Before developing the models, a number of general assumptions were made:

(i) A limited number of dock-doors are available at the hub for unloading.

(ii) Each dock-door can serve one truck at the time.

(iii) Each truck needs a fixed amount of minutes for unloading all shipments, and this unloading process cannot be interrupted.

(iv) Each truck needs a fixed amount of minutes to make a stop at an extra location to load additional shipments.

(v) A location cannot be visited by a truck during its closing hours.

(vi) Trucks should be able to immediately start unloading after arriving at the hub, which means that waiting times at the hub are not allowed.

(vii) Each truck can start its route at a random location and should finish its route at the hub.

(viii) Each truck should undock from the dock-door before the earliest arrival deadline of any shipment it carries is reached.

(ix) Each truck has 4 cargo positions, each with 25% of the total truck volume capacity.

(x) The fleet of trucks is heterogenous, which means that each truck has a predefined volume based capacity and cooling type.

(xi) Weight based capacity of trucks is assumed not to be a limiting constraint.

(xii) There are three types of trucks: non-refrigerated, temperature controlled type 1 and temperature controlled type 2.

(xiii) Each shipment that needs to stay within a predefined temperature range should be placed in a truck with the corresponding temperature conditions. Shipments that do not need to be placed in a temperature controlled truck can be placed in every truck.

(xiv) Each shipment has an availability time, specifying when the shipment is ready for pickup. Trucks cannot collect a shipment if it arrives at a station before that time.

(xv) Shipments that are incompatible with each other cannot be transported by the same truck.

(xvi) A shipment can be split over multiple trucks.

(xvii) Constraints about truck driver times are neglected.

(xviii) The cost of each truck is calculated based on the individual legs of its route. This base cost is adjusted by a cooling factor if the truck is refrigerated and further modified by a load factor. The load factor is determined by the maximum number of cargo positions filled upon the truck's arrival at the hub (e.g., 1 position = 70%, 2 positions = 80%, 3 positions = 90%, 4 positions = 100%).

3.2 Mixed Integer Linear Programming Formulation

A novel MILP formulation has been developed based on the MIP formulation of the model presented in the paper of Liao (2021). This model served as the basis for designing the MILP formulation. The model has been linearised, with several constraints added or removed to align with the routing and scheduling assumptions described above.

In order to ensure that shipments are separated when they are not allowed to be transported together, shipment incompatibility constraints have been added to the model. The mathematical formulations of Manerba and Mansini (2015) and Gendreau et al. (2016) were used as references to incorporate these constraints. Their formulations contain pairwise incompatibility constraints to prevent the loading of two or more incompatible products into the same truck. Similar to those models, this model introduces a set that defines incompatibilities between product types, along with constraints that prevent incompatible product types from being placed in the same truck.

Furthermore, the model of Liao (2021) addresses a closed routing problem in which all trucks start and finish their route at a hub. The model has been adjusted to an open routing problem, which means that

each truck can start its route at any station in the network and finishes its route at the hub. Therefore, a dummy hub has been introduced to the model, as demonstrated in the paper of Vincent et al. (2016). This dummy hub has arcs connecting to all stations in the network with zero travel cost and zero travel time, and all combinations of legs starting at the dummy hub will be excluded from the cost calculation in the objective function. If the leg from the dummy hub to the actual hub is activated for a truck, it means that this truck will not be used in the network.

Moreover, the fleet described in the model by Liao (2021) has been modified from a homogeneous fleet to a heterogeneous fleet. Furthermore, the model has been adapted to a routing problem with split delivery, as described by Dror et al. (1994). This allows multiple trucks to visit the same station in the network instead of the restricted visits of one truck per station. In addition, multiple sets of time constraints are introduced to the model. An undocking deadline is set for each truck to ensure on-time delivery of all its shipments. This deadline is determined by the shipment with the earliest transfer time to its next mode of transport. Additionally, constraints are added that prevent a truck from arriving at or departing from a station outside of operating hours, and a constraint is introduced that prevents a truck from picking up a shipment before it has been become available at its origin station.

The next part of this section presents the novel MILP formulation and notation. Table 1, Table 2, and Table 3 respectively show the sets, decision variables, and parameters used in the mathematical formulation. After that, the mathematical formulation, including the objective function and constraints, will be presented.

Table 1: Model Sets.

Set	Description	
$\mathcal{T}_d$	Set of trucks including 2 dummy trucks (i=0 & i=-1)	$i \in \mathcal{T}_d$
$\mathcal{T}$	Set of trucks	$i \in \mathcal{T} \subseteq \mathcal{T}_d$
$\mathcal{T}_{nc}$	Set of non refrigerated trucks	$i \in \mathcal{T}_{nc} \subseteq \mathcal{T}$
$\mathcal{T}_{2-8deg}$	Set of $2-8$ deg refrigerated trucks	$i \in \mathcal{T}_{2-8deg} \subseteq \mathcal{T}$
$\mathcal{T}_{15-25deg}$	Set of $15-25$ deg refrigerated trucks	$i \in \mathcal{T}_{15-25deg} \subseteq \mathcal{T}$
$\mathcal{J}_d$	Set of stations including dummy hub (j=-1)	$j \in \mathcal{J}_d$
$\mathcal{J}_h$	Set of stations including hub (j=0)	$j \in \mathcal{J}_h$
$\mathcal{J}$	Set of stations excluding (dummy) hub	$j \in \mathcal{J}$
$\mathcal{P}$	Set of shipments	$p \in \mathcal{P}$
$\mathcal{P}_{2-8deg}$	Set of $2-8$ deg refrigerated shipments	$p \in \mathcal{P}_{2-8deg} \subseteq \mathcal{P}$
$\mathcal{P}_{15-25deg}$	Set of $15-25$ deg refrigerated shipments	$p \in \mathcal{P}_{15-25deg} \subseteq \mathcal{P}$
$\mathcal{P}_j$	Set of shipments in station j	$p, j \in \mathcal{P}_j \subseteq \mathcal{P}$
$\mathcal{D}$	Set of dock-doors	$d \in \mathcal{D}$
$\mathcal{B}$	Set of incompatible shipments	$(p1, p2) \in \mathcal{B} \subseteq \mathcal{P}$ x $\mathcal{P}$
$\mathcal{K}$	Set of truck positions	$k \in \mathcal{K}$
$\mathcal{CH}$	Set of closing hours	$(t^c, t^o) \in \mathcal{CH}$

Table 2: Model Decision Variables.

Decision Variable	Description
$z^{pos}_{i,j,j',k}$	$\begin{cases} 1 & \text{if truck } i \text{ travels from station } j \text{ to station } j' \text{ with } k \text{ positions filled,} \\ 0 & \text{otherwise} \end{cases}$
$z_{i,j,j'}$	$\begin{cases} 1 & \text{if truck } i \text{ travels from station } j \text{ to station } j', \\ 0 & \text{otherwise} \end{cases}$
$x_{i,i'}$	$\begin{cases} 1 & \text{if truck } i \text{ docks before truck } i' \text{ at the same dock,} \\ 0 & \text{otherwise} \end{cases}$
$y_{i,d}$	$\begin{cases} 1 & \text{if truck } i \text{ is assigned to dock } d, \\ 0 & \text{otherwise} \end{cases}$
$w_{i,p}$	$\begin{cases} 1 & \text{if truck } i \text{ collects (part of) shipment } p, \\ 0 & \text{otherwise} \end{cases}$
$pos_{i,k}$	$\begin{cases} 1 & \text{if truck } i \text{ has exactly } k \text{ positions filled,} \\ 0 & \text{otherwise} \end{cases}$
$btc_{i,j,(t^c,t^o)}$	$\begin{cases} 1 & \text{if truck } i \text{ finishes operations at station } j \text{ before closing time } t^c \text{ of the closing hours,} \\ 0 & \text{if truck } i \text{ finishes operations at station } j \text{ after opening time } t^o \text{ of the closing hours} \end{cases}$
$v_{i,p}$	Volume of shipment p collected by truck i
t^{dock}_i	Docking time of truck i at hub
t^{undock}_i	Undocking time of truck i at hub
$t^s_{i,j}$	Departure time of truck i from station j
$t^s_{i,0}$	Arrival time of truck i at the hub
l_i	Total volume (m^3) of shipments loaded in truck i

Table 3: Model Parameters.

Parameter	Description
Q^t_i	Capacity (m^3) of truck i
Q^p_k	Capacity (m^3) of k truck positions
V_p	Volume (m^3) of shipment p
$C^t_{j,j'}$	Transportation cost between station j and j'
C^p_k	Cost multiplier for k positions filled
C^c_i	Cost multiplier for refrigerated trucks for truck i
T^l	Loading time (h) at station
T^{ul}	Unloading time (h) at hub
$T^t_{j,j'}$	Transportation time (h) between station j and j'
T^{sa}_p	Time shipment p becomes available for pickup
T^{sd}_p	Time of delivery deadline of shipment p
M	Large positive number used in the Big M Method
E	Lowest volume (m^3) that a shipment can be split into

Objective Function:

$$\min \sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{J}} \sum_{\substack{j' \in \mathcal{J}_h, \\ j \neq j'}} \sum_{k \in \mathcal{K}} C^t_{j,j'} \cdot C^p_k \cdot C^c_i \cdot z^{pos}_{i,j,j',k} \tag{1}$$

subject to:

Routing Constraints:

$$\sum_{j \in \mathcal{J}_d} z_{i,j,0} = 1, \quad \forall i \in \mathcal{T} \tag{2}$$

$$\sum_{\substack{j \in \mathcal{J}_d, \\ j \neq h}} z_{i,j,h} - \sum_{\substack{j' \in \mathcal{J}_h, \\ j' \neq h}} z_{i,h,j'} = 0, \quad \forall i \in \mathcal{T}, h \in J \tag{3}$$

Capacity and Supply Constraints:

$$\sum_{p \in P} v_{i,p} = l_i, \quad \forall i \in \mathcal{T} \tag{4}$$

$$\sum_{p \in P} v_{i,p} \leq Q^t_i, \quad \forall i \in \mathcal{T} \tag{5}$$

$$\sum_{i \in \mathcal{T}} v_{i,p} = V_p, \quad \forall p \in \mathcal{P} \tag{6}$$

$$v_{i,p} \leq V_p \cdot \sum_{\substack{j' \in \mathcal{J}_h, \\ j \neq j'}} z_{i,j,j'}, \quad \forall i \in \mathcal{T}, j \in \mathcal{J}, p \in \mathcal{P}_j \quad (7)$$

Truck Position Constraints:

$$pos_{i,1} + pos_{i,2} + pos_{i,3} + pos_{i,4} = 1 - z_{i,\text{-}1,0}, \quad \forall i \in \mathcal{T} \quad (8)$$

$$l_i \leq Q_1^p + (1 - pos_{i,1}) \cdot Q_3^p, \quad \forall i \in \mathcal{T} \quad (9)$$

$$l_i \leq Q_2^p + (1 - pos_{i,2}) \cdot Q_2^p, \quad \forall i \in \mathcal{T} \quad (10a)$$

$$l_i > Q_1^p \cdot pos_{i,2}, \quad \forall i \in \mathcal{T} \quad (10b)$$

$$l_i \leq Q_3^p + (1 - pos_{i,3}) \cdot Q_1^p, \quad \forall i \in \mathcal{T} \quad (11a)$$

$$l_i > Q_2^p \cdot pos_{i,3}, \quad \forall i \in \mathcal{T} \quad (11b)$$

$$l_i > Q_3^p \cdot pos_{i,4}, \quad \forall i \in \mathcal{T} \quad (12)$$

$$z^{pos}_{i,j,j',k} \geq z_{i,j,j'} + pos_{i,k} - 1, \quad \forall i \in \mathcal{T}, j \in \mathcal{J}_d, j' \in \mathcal{J}_h, j \neq j', k \in \mathcal{K} \quad (13a)$$

$$z^{pos}_{i,j,j',k} \leq z_{i,j,j'}, \quad \forall i \in \mathcal{T}, j \in \mathcal{J}_d, j' \in \mathcal{J}_h, j \neq j', k \in \mathcal{K} \quad (13b)$$

$$z^{pos}_{i,j,j',k} \leq pos_{i,k}, \quad \forall i \in \mathcal{T}, j \in \mathcal{J}_d, j' \in \mathcal{J}_h, j \neq j', k \in \mathcal{K} \quad (13c)$$

Dock Time Constraints:

$$t_{i'}^{dock} \geq t_i^{undock} - M \cdot (1 - x_{i,i'}), \quad \forall i, i' \in \mathcal{T}, i \neq i' \quad (14)$$

$$t_i^{dock} \leq M \cdot (1 - z_{i,\text{-}1,0}), \quad \forall i \in \mathcal{T} \quad (15a)$$

$$t_i^{dock} \geq -M \cdot (1 - z_{i,\text{-}1,0}), \quad \forall i \in \mathcal{T} \quad (15b)$$

$$t_i^{undock} = t_i^{dock} + T^{ul} \cdot (1 - z_{i,\text{-}1,0}), \quad \forall i \in \mathcal{T} \quad (16)$$

Docking Deadline Constraint:

$$T_p^{sd} + M \cdot (1 - w_{i,p}) \geq t_i^{undock}, \quad \forall i \in \mathcal{T}, p \in \mathcal{P} \quad (17)$$

Dock Allocation Constraints:

$$\sum_{d \in \mathcal{D}} y_{i,d} = 1 - z_{i,\text{-}1,0}, \quad \forall i \in \mathcal{T} \quad (18)$$

$$x_{i,i'} - 1 \leq y_{i,d} - y_{i',d}, \quad \forall i, i' \in \mathcal{T}, i \neq i', d \in \mathcal{D} \quad (19)$$

$$y_{i,d} - y_{i',d} \leq 1 - x_{i,i'}, \quad \forall i, i' \in \mathcal{T}, i \neq i', d \in \mathcal{D} \quad (20)$$

$$\sum_{\substack{i \in \mathcal{T}_d, \\ i \neq -1, \\ i \neq i'}} x_{i,i'} = 1 - z_{i',\text{-}1,0}, \quad \forall i' \in \mathcal{T} \quad (21)$$

$$\sum_{\substack{i' \in \mathcal{T}_d, \\ i' \neq 0, \\ i' \neq i}} x_{i,i'} = 1 - z_{i,\text{-}1,0}, \quad \forall i \in \mathcal{T} \quad (22)$$

$$\sum_{i' \in \mathcal{T}} x_{0,i'} \leq |D| \quad (23)$$

$$\sum_{i \in \mathcal{T}} x_{i,\text{-}1} \leq |D| \quad (24)$$

$$x_{0,i} + x_{0,i'} + y_{i,d} + y_{i',d} \leq 3, \quad \forall i, i' \in \mathcal{T}, i \neq i', d \in \mathcal{D} \quad (25)$$

Pairwise Incompatibility Constraints:

$$w_{i,p} \cdot E \leq v_{i,p}, \quad \forall i \in \mathcal{T}, p \in \mathcal{P} \quad (26)$$

$$v_{i,p} \leq Q_i^t \cdot w_{i,p}, \quad \forall i \in \mathcal{T}, p \in \mathcal{P} \quad (27)$$

$$w_{i,p1} + w_{i,p2} \leq 1, \quad \forall i \in \mathcal{T}, (p1, p2) \in \mathcal{B} \quad (28)$$

Truck Cooling Constraints:

$$\sum_{i \in \mathcal{T}_{2-8deg}} \sum_{p \in \mathcal{P}_{15-25deg}} v_{i,p} = 0 \quad (29)$$

$$\sum_{i \in \mathcal{T}_{15-25deg}} \sum_{p \in \mathcal{P}_{2-8deg}} v_{i,p} = 0 \quad (30)$$

$$\sum_{i \in \mathcal{T}_{nc}} \sum_{p \in (\mathcal{P}_{15-25deg} \cup \mathcal{P}_{2-8deg})} v_{i,p} = 0 \quad (31)$$

Sub-tour Elimination / Station Service Time Constraints:

$$t_{i,j}^s + T_{j,j'}^t + T^l \cdot (1 - z_{i,j,0}) - M \cdot (1 - z_{i,j,j'}) \leq t_{i,j'}^s, \quad \forall i \in \mathcal{T},\ j \in \mathcal{J},\ j' \in \mathcal{J}_h,\ j \neq j' \quad (32a)$$

$$t_{i,j}^s + T_{j,j'}^t + T^l \cdot (1 - z_{i,j,0}) + M \cdot (1 - z_{i,j,j'}) \geq t_{i,j'}^s, \quad \forall i \in \mathcal{T},\ j \in \mathcal{J},\ j' \in \mathcal{J}_h,\ j \neq j' \quad (32b)$$

$$t_{i,j}^s \leq M \cdot \sum_{\substack{j' \in \mathcal{J}_h, \\ j \neq j'}} z_{i,j,j'}, \quad \forall i \in \mathcal{T}, j \in \mathcal{J} \quad (33)$$

$$t_{i,0}^s = t_i^{dock}, \quad \forall i \in \mathcal{T} \quad (34)$$

Shipment Availability at Station Arrival Time Constraint:

$$t^s_{i,j} - T^l \geq T^{sa}_p - M \cdot (1 - w_{i,p}), \quad \forall i \in \mathcal{T}, j \in \mathcal{J}, p \in \mathcal{P}_j \quad (35)$$

Out Station Time Window Constraints:

$$t^s_{i,j} \leq t^c + M \cdot (1 - btc_{i,j,(t^c,t^o)}), \quad \forall i \in \mathcal{T}, j \in \mathcal{J}, (t^c, t^o) \in \mathcal{CH} \quad (36a)$$

$$t^s_{i,j} \geq t^o + T^l - M \cdot btc_{i,j,(t^c,t^o)}, \quad \forall i \in \mathcal{T}, j \in \mathcal{J}, (t^c, t^o) \in \mathcal{CH} \quad (36b)$$

Domain Decision Variables:

$$z^{pos}_{i,j,j',k} \in \{0,1\}, \quad \forall i \in \mathcal{T}, j \in \mathcal{J}_d, j' \in \mathcal{J}_h, j \neq j', k \in \mathcal{K} \quad (37)$$

$$z_{i,j,j'} \in \{0,1\}, \quad \forall i \in \mathcal{T}, j \in \mathcal{J}_d, j' \in \mathcal{J}_h, j \neq j' \quad (38)$$

$$x_{i,i'} \in \{0,1\}, \quad \forall i, i' \in \mathcal{T}, i \neq i' \quad (39)$$

$$y_{i,d} \in \{0,1\}, \quad \forall i \in \mathcal{T}, d \in \mathcal{D} \quad (40)$$

$$w_{i,p} \in \{0,1\}, \quad \forall i \in \mathcal{T}, p \in \mathcal{P} \quad (41)$$

$$pos_{i,k} \in \{0,1\}, \quad \forall i \in \mathcal{T}, k \in \mathcal{K} \quad (42)$$

$$btc_{i,j,(t^c,t^o)} \in \{0,1\}, \quad \forall i \in \mathcal{T}, j \in \mathcal{J}, (t^c, t^o) \in \mathcal{CH} \quad (43)$$

Equation 1 is the objective function and aims to minimise the sum of the costs per truck used in the network, which is based on the truck route, truck load factor and truck cooling type factor. Constraint 2 ensures that every truck completes its route at the hub. Constraint 3 ensures that each truck leaves a station as many times as it arrives at that station, thereby creating a continuous and uninterrupted route.

Constraints 4 - 7 is the group of capacity and supply constraints. Constraint 4 ensures that the total volume of shipments in each truck equals the sum of the individual shipment volumes. The volume of shipments in each truck is limited to the volume capacity of each truck in constraint 5. Constraint 6 ensures that each shipment is picked up from its origin station. Furthermore, constraint 7 ensures that a truck can only pick up a shipment if it departs from the station supplying the shipment.

Constraints 8 - 13c handle the decision variables that involve determining the amount of positions filled in each truck, i.e. 1, 2, 3, or 4 positions. Constraint 8 ensures that if a truck is activated in the network, only 1, 2, 3, or 4 positions can be filled. Otherwise, all corresponding truck position decision variables are set to zero. Constraints 9, 10a & 10b, 11a & 11b, and 12 determine whether 1, 2, 3, or 4 positions are filled in a truck, respectively. Constraints 13a - 13c ensure that if truck i travels between station j and j' and k positions are filled, decision variable $z_{ijj'k}$ should be set to 1, 0 otherwise.

Constraints 14 - 17 determine the correct and feasible docking times and undocking times of the the trucks used in the network. In these constraints, the big M represents the latest delivery deadline among all shipments. Constraint 14 ensures that if a truck docks after an another truck at the same dock-door, the second truck docks after that the first truck has undocked from the dock-door. This ensures that the trucks do not have overlapping times at the same dock-door. The docking time of an unused truck is set to 0 by constraints 15a and 15b. The undocking time of a truck is determined by constraint 16. Similar to the docking time of a truck, its undocking time is set to 0 if a truck has not been assigned a route. Finally, constraint 17 ensures that a truck meets the delivery deadlines of the shipments loaded into it by undocking before the earliest deadline of any of the shipments has passed.

The group of constraints 18 - 25 ensures a correct and feasible schedule for the allocation of trucks to dock-doors. Constraint 18 ensures that if a route is assigned to a truck it should be allocated to one of the dock-doors at the hub. Furthermore, constraints 19 and 20 allocates two trucks to the same dock-door if they are scheduled to dock after each other at the same dock-door. Constraints 21 and 22 ensure that each truck allocated to a dock-door has, respectively, a preceding truck or dummy truck and a succeeding truck or dummy truck at the dock-door. Two dummy trucks are introduced for each available dock-door at the hub. The dummy trucks are placeholders used to manage situations at the start and end of the scheduling sequence, serving to initiate and close the docking sequence. This is ensured by constraints 23 and 24. In addition, constraint 25 ensures that if two trucks have a dummy truck preceding them at their allocated dock-door, they must be assigned to two different dock-doors.

Constraints 26 - 31 represent the pairwise incompatibility and truck cooling constraints, which ensure that incompatible shipments are not loaded into the same truck and that shipments are loaded into trucks with the corresponding environmental cooling conditions. A binary decision variable w is introduced to indicate whether a shipment is loaded into a truck ($w = 1$) or not ($w = 0$). If the volume of a shipment exceeds the parameter E, the shipment is considered to be loaded into the truck. This implicitly introduces a lower bound (E) on the minimal volume of a shipment that must be loaded into a truck. Constraints 26 and 27 are introduced to determine the value of decision variable w. Moreover, constraint 28 ensures that

incompatible shipments are not loaded into the same truck using the previously described binary decision variable w. Constraints 29 - 31 ensure that shipments are only loaded into trucks that satisfy the shipments cooling requirements.

The constraints that determine the truck departure times at each station are presented by constraints 32a - 36b. Constraints 32a and 32b ensure the elimination of sub-tours and the correct determination of station departure times by using travel times between previously visited stations and loading times at those stations. If a truck does not travel to a station, the departure time from that station is set to zero by constraint 33. Furthermore, constraint 34 ensures that a truck immediately start its docking process at one of the dock-doors once it arrives at the hub. Constraint 35 ensures that a truck cannot collect a shipment if it arrives at the shipment's origin station before the shipment becomes available for pick up. The last two operational constraints prevent trucks from arriving or departing from stations during their closing hours. This is shown by constraints 36a and 36b. The big M used in constraint 35 represents the time when the last shipment becomes available for pickup among all shipments. The big M used in the other constraints represents the latest delivery deadline among all shipments.

The last group of constraints defines the domain of the decision variables used in the mathematical model. They are presented by constraints 37 - 43 and conclude the model's formulation.

3.3 Heuristic Algorithm: an Adaptive Large Neighbourhood Search

Numerous heuristic approaches have been proposed in the literature to solve VRPs. Among them, the Adaptive Large Neighbourhood Search (ALNS) algorithm, introduced by Ropke and Pisinger (2006), has demonstrated significant successes in addressing VRPs (Liang et al., 2023; Pisinger & Ropke, 2007; Schopka & Kopfer, 2015). The principles of ALNS were chosen as the foundation for designing the heuristic algorithm to solve the integrated VRP and DAP. To further improve its performance, the algorithm incorporates a SA acceptance criterion, which balances exploration and exploitation by probabilistically accepting inferior solutions to escape local optima.

3.3.1 Solution Representation

Before introducing the algorithm's framework, it is important to explain how the solution representation is constructed, as it provides the foundation for its execution. The principle of a dock-based route representation, introduced by Liang et al. (2023), is used to store and organise the solution structure during the algorithm's execution. Each dock-door is represented by a separate string, divided into equal docking time windows spanning the problem's timespan. Each window is allocated to a specific truck identified by a unique ID, illustrated by Figure 1. Furthermore, each truck has its own individual string representing a unique sequence of shipments assigned to that truck, captured by Figure 2. This sequence determines the truck's route based on the origin station of the shipments, shown at the bottom of Figure 2. The routes and station departure times of each truck are derived after the shipments are allocated to a truck. This is done in reverse, starting from the hub, as the docking time is pre-fixed. This ensures that every truck has a reserved docking slot and a planned route aligned with the overall schedule. This structure enables efficient problem-solving by coordinating truck-to-dock-door assignments and route planning.

Figure 1: Solution representation of the dock-door schedule.

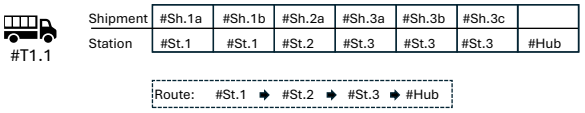


Figure 2: Solution representation of the shipment allocation and truck route.

3.3.2 Cost Calculation

Similar to the MILP formulation, the primary objective of the ALNS is to minimise the operational costs of the trucks within the network. The costs per truck in the network are based on the individual legs that make up the truck's route, the load factor of the shipments being transported, and an additional factor that is applied if the truck is equipped with a refrigerated trailer. The base cost of the truck route is determined by summing all separate leg costs of the legs in the truck route. This cost is multiplied by a cooling multiplier if the truck uses a refrigerated trailer. Furthermore, the truck's total cost is adjusted by a cost multiplier based on its load factor at the hub, i.e., when the truck carries its highest load factor during the trip. This multiplier is predefined for each number of filled positions.

In contrast to the MILP formulation, this ALNS approach introduces an extra type of cost for trucks with an infeasible time schedule. A penalty, in the form of an extra cost, is applied to these trucks. These time penalty costs are applied if the routing does not satisfy one or more of the following conditions:

1. A truck arrives at or departs from a station during operating hours;
2. A truck picks up a shipment from a station after the shipment is available for pickup.

The time penalty costs serve as a mechanism to convert the hard time constraints of the MILP formulation into soft time constraints within the ALNS framework. The soft constraints are introduced to allow for greater flexibility in navigating through the solution space without being overly constrained. By minimising the costs described above, the goal is to achieve a time penalty cost of zero, ensuring that the time constraints are fully satisfied and no longer violated.

The time penalty cost for each truck is determined for each station visit by calculating the time differences relative to feasible operational time conditions. For the first case, the time difference is determined by calculating the smallest time difference between the truck's infeasible operating times (arrival and departure) at the dock and the station's operating hours. For the second case, the time difference is calculated by identifying the latest release time of the shipments scheduled for pickup at each station along the truck's route. This time difference is then determined as the interval between the truck's arrival time at the station and the latest release time of any shipment at that station. Once the relevant time differences have been calculated, they are multiplied by a predefined time penalty factor. The resulting penalty is then added to the real trucking costs previously described.

3.3.3 Algorithm Framework

A flowchart of the ALNS framework is presented in Figure 3. The framework starts with the initialisation of the SA and ALNS parameters. After that, the input data is preprocessed to make it compatible with the model. Then an initial solution is created to begin with. This process is explained in Section 3.3.4.

In the next step, the ALNS selects one of the five available removal operators and one of the three insertion operators based on their respective operator weights. The functioning and selection of removal and insertion operators is explained in Section 3.3.5. After the selection of the operators, the temperature and iteration number are updated based on the SA principle.

The selected removal operator selects a number of shipments to be removed from the trucks. This number of shipments is a percentage of the total shipments in the problem, which is defined during the initialisation of the algorithm parameters. If the removal is successfully executed, the next step involves re-inserting the removed shipments based on the logic of the selected insertion operator. The shipments will be re-inserted into specific trucks at specific positions in the solution string of the truck. The sequence of shipments in the solution string of each truck determines the route sequence, as shown in Figure 2.

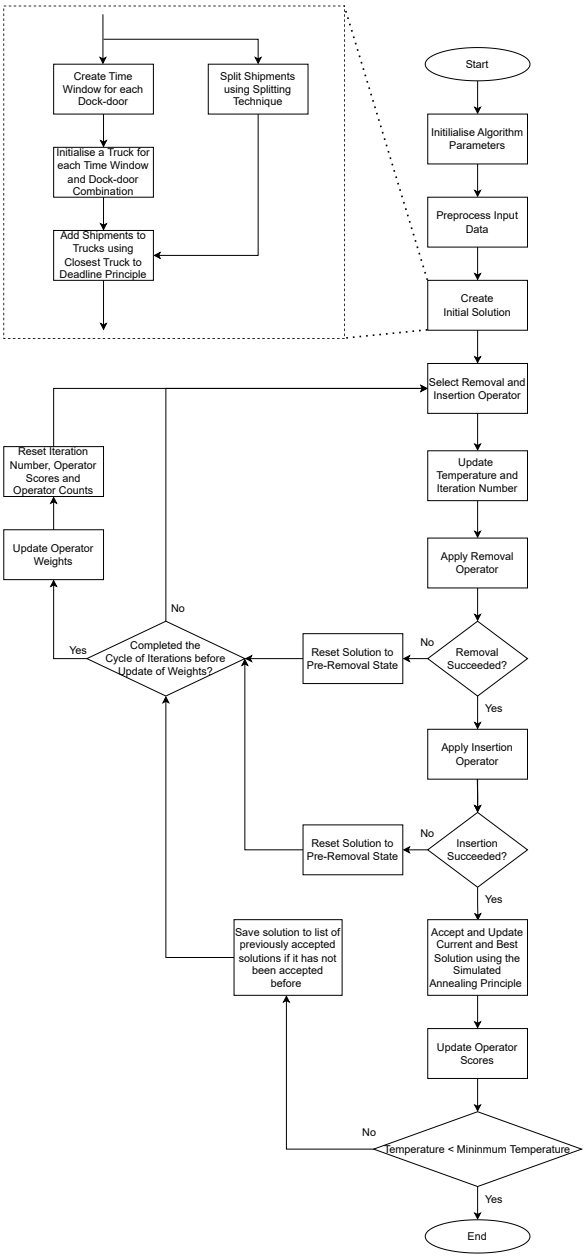


Figure 3: Flowchart of the ALNS Framework for the Vehicle Routing and Dock-door Scheduling Problem.

If the insertion of all removed shipments succeeds, the new solution will be accepted or rejected based on the SA principle, which is explained in Section 3.3.6. Following this, the scores of the selected operators are updated based on their performance during the current iteration. Then, if the temperature has dropped below the specified minimum temperature, the algorithm stops and outputs its best found solution.

If the temperature has not reached the minimum temperature threshold, or if the removal or insertion of shipments has not succeeded, a new iteration starts, and a new set of removal and insertion operators will be selected. Before moving to the next iteration, the algorithm checks whether a cycle of a predefined num-

ber of iterations has been completed. After completing a cycle of iterations, the operator weights are updated based on their performance scores obtained during the cycle, and the iteration parameters and operator performance scores are reset before starting a new cycle. Section 3.3.7 will explain how the operator weights are updated after the completion of one cycle.

3.3.4 Initial Solution and Volume Splitting

Before constructing the initial solution, the shipment volumes are divided using the splitting technique described by Chen et al. (2017). An a priori splitting strategy is applied based on the maximum capacity of the trucks. All shipments are divided into smaller portions based on the truck's maximum capacity. If the total shipment volume exceeds the truck's capacity, the first split creates a portion equal to half the trucks capacity. This process of halving continues until the remaining volume is less than 1 unit. Once the remaining volume is smaller than 1, the splitting process stops. This means for a truck with a volume capacity of $40m^3$, the "40/20/10/5/2/1" rule is applied to divide the shipment into as many large portions as possible, following the quantities defined by the rule.

The initial solution is created using the split shipments and the trucks assigned to each docking time window. By adding shipments to trucks with an undocking time closest to and before the shipment's arrival deadline. This process is carried out for each shipment individually. If a shipment cannot be added to a truck because its volume exceeds the truck's remaining capacity, or due to differing cooling requirements or incompatibilities with the shipments already assigned to that truck, the process is repeated with the next truck closest to the shipment's deadline.

3.3.5 Removal and Insertion Operators

After the initial solution has been generated, the algorithm starts improving the solution by iteratively destroying and repairing it. At the start of each iteration, a removal operator and an insertion operator are selected based on their weights. These weights are adjusted after a cycle of iterations has been completed, which will be described in Section 3.3.7. A roulette wheel selection principle is used to select a removal and insertion operator, where the probability of selecting a particular operator is proportional to its weight. The probability of choosing operator i is defined as the ratio of its weight to the sum of the weights of all operators, as shown in Equation 44 (Ropke & Pisinger, 2006):

$$P(\textit{Selecting operator } i) = \frac{w_i}{\sum_{i=1}^{k} w_i} \tag{44}$$

A total of five removal operators and three insertion operators have been added to the algorithm. While an endless number of operators can be used in the algorithm, their number should be limited to ensure that the algorithm remains efficient. Voigt (2024) conducted a literature review on ALNS for VRPs and ranked the operators based on their effectiveness. Based on the findings presented in his paper, a number of operators has been selected for the ALNS framework, along with two additional removal operators and one additional insertion operator. The choice of operators is made such that a balance is achieved between intensification and diversification. During each iteration a predefined percentage of the total shipments in the solution will be removed, after which the insertion operator will re-insert all removed shipments into the solution.

The following removal operators have been added to the model:

- Removal of random shipments (Ropke & Pisinger, 2006);
- Removal of shipments with highest cost (Ropke & Pisinger, 2006);
- Removal of a posteriori score related shipments (Shaw Removal) (Shaw, 1997);
- Removal of all shipments from random trucks;
- Removal of all shipments from random trucks with a penalty cost

Below, all five removal operators are described in detail. Note that, to successfully complete the removal phase, each removal operator must aim to remove the pre-specified percentage of shipments from the solution string.

Removal - Random Shipments
The random removal of shipments operator removes the shipments at random until the predefined percentage of shipments has been removed.

Removal - Shipments with Highest Cost
This operator focuses on removing shipments with the highest associated costs to improve overall cost efficiency. It identifies and calculates the removal costs for each shipment, ranks them based on these costs, and probabilistically selects the highest cost shipments for removal. The pseudocode for this operator can be found in the paper of Ropke and Pisinger (2006)

Removal - A Posteriori Score-related Shipments
This operator, also known as the Shaw Removal Heuristic, aims to remove clusters of related shipments. It removes shipments based on their relatedness to improve the solution's flexibility and diversity. It starts by randomly selecting an initial shipment and iteratively adds the most related shipments to a removal set by using a similarity measure. The similarity between shipments is calculated using a combination of five shipment characteristics: the distance between shipments' origins, the travel cost between origins, the differences in shipment arrival deadlines and availability times, and the difference in shipment types. Shipments are then probabilistically selected for removal, favouring those with higher similarity. This

operator aims to remove clusters of related shipments. The pseudocode for this operator can be found in the paper of Ropke and Pisinger (2006). The procedure for determining the shipments' similarity is shown in pseudocode in Algorithm 1.

Algorithm 1 Shipment Similarity Determination

Input: $Shipment1$, $Shipment2$
Output: $Similarity\_factor$
$S1 \leftarrow Shipment1$
$S2 \leftarrow Shipment2$
$vol\_diff \leftarrow |S1.volume - S2.volume|$
Normalise vol_diff
if S1 and S2 are split from the same shipment **then**
  $Similarity\_factor \leftarrow 1/(1 + vol\_diff)$
  return $Similarity\_factor$
end if
if S1.type = S2.type **then**
  $type\_diff \leftarrow 0$
else
  $type\_diff \leftarrow 1$
end if
$dl\_diff \leftarrow |S1.deadline - S2.deadline|$
$avail\_diff \leftarrow |S1.availability - S2.availability|$
Normalise dl_diff and $avail\_diff$
if S1.origin = S2.origin **then**
  $Similarity\_factor \leftarrow 1/(1 + vol\_diff + dl\_diff + avail\_diff + type\_diff)$
else
  $time \leftarrow$ travel time between $S1.origin$ and $S2.origin$
  $cost \leftarrow$ travel cost for travelling between $S1.origin$ and $S2.origin$
  Normalise $time$ and $cost$
  $Similarity\_factor \leftarrow 1/(1 + vol\_diff + dl\_diff + avail\_diff + type\_diff + time + cost)$
end if
return $Similarity\_factor$

Removal - All Shipments from Random Trucks
This operator selects random trucks from which all shipments are removed. The operator continues to select new trucks until the number of shipments to be removed is reached. All shipments will be removed from the last truck, even if the number of shipments to be removed has been reached.

Removal - All Shipments from Random Trucks with a Penalty Cost
For this operator, a list is created that includes all trucks in the current solution that add a penalty cost to the objective function. This operator randomly selects trucks from this list from which all shipments are removed. Similarly, to the operator described above, this operator continues to select trucks until the correct percentage of shipments has been removed. If the required percentage cannot be reached due to insufficient trucks in the list, the removal phase ends prematurely, and the insertion phase reinserts the removed shipments back into the solution string. Furthermore, note that if no trucks with a penalty cost exist at the beginning, this operator cannot execute. In such cases, a different removal and insertion operator is selected.

After the removal phase is completed, the selected insertion operator will reinsert the removed shipments and repair the solution. The available insertion operators are listed below:

- Insertion in random order at best position (Greedy Insertion) (Lei et al., 2011);
- Insertion in random order into random truck at best position
- Insertion highest position 2 regret at best position (Qu & Bard, 2012);

During the insertion phase each insertion operator performs the same feasibility check to ensure that the shipment can be inserted into the selected truck. This feasibility check verifies the insertion's compliance with truck capacity limits, shipment incompatibilities, truck cooling requirements, and arrival deadlines. The remaining part of this section explains each of the insertion operator in more detail.

Insertion - Random Order at Best Position
This insertion operator, known as Greedy Insertion, randomly selects a shipment from the set of removed shipments. It makes a list of trucks suitable for loading the shipment, ensuring the solution remains feasible. The operator then inserts the shipment into one of these trucks at the position that minimises the additional cost to the objective function. This process is repeated until all shipments have been reinserted.

If the removal operator used to remove the shipments to be reinserted is the 'Removal of all shipments from random trucks with a penalty cost', the first shipment to be reinserted will be placed in the best position within the solution string of an empty truck. This allows all shipments to potentially be moved to the same new truck without incurring a time penalty. However, if the first shipment is placed in a non-empty truck, it removes the opportunity to allocate all shipments into the same truck, as the insertion into a non-empty truck is typically cheaper. The primary goal of this removal operator is to eliminate the penalty cost.

Insertion - Random Order into Random Trucks at Best Position
This insertion operator randomly selects a shipment from the list of removed shipments and selects a random truck that is suitable for loading the shipment. Then, it will try to insert the selected shipment into the selected truck at the position which adds the lowest cost to the objective function. This process is repeated until all shipments have been reinserted. Similar to the insertion operator described before, the first shipment to be reinserted will be placed in the best position within the solution string of an empty truck if the

removal operator 'Removal of all shipments from random trucks with a penalty cost' was used to destroy the solution in that iteration.

Insertion - Highest Position 2 Regret at Best Position

The process of this insertion operator begins with determining the insertion costs of every shipment at every feasible position in the solution string. As explained, feasible positions are identified by considering arrival deadlines, truck capacity, cooling requirements, and shipment incompatibilities. The calculated insertion costs are then sorted in ascending order for each shipment. The regret value for each shipment is calculated as the difference between the costs of inserting the shipment at the best and second-best position. If there is only one feasible position, the regret value the insertion cost of that position. The operator then selects the shipment with the highest regret value among all the shipments. This approach minimises potential future regret by ensuring that the most critical shipment is inserted first. The selected shipment is inserted into the truck at the position with the lowest cost. This process repeats until all shipments are reinserted into the solution. Similarly to the insertion operators described above, the first shipment to be reinserted will be placed within the solution string of an empty truck if the removal operator 'Removal of all shipments from random trucks with a penalty cost' is selected. The procedure of this insertion heuristic is shown in pseudocode in Algorithm 2

Algorithm 2 Insertion - Highest Position 2 Regret at Best Position Algorithm

Input: *removed_shipments* (List of removed shipments), *full_truck_removal_with_penalty* (Boolean stating if shipments are removed from truck with time penalty)
Output: Insertion Costs
insertion_costs ← 0
while *removed_shipments* is not empty **do**
  Set *regret_values* to an empty list
  for each *shipment* in *removed_shipments* **do**
    Set *insertions* to an empty list
    if *full_truck_removal_with_penalty* and *shipment* is first removed from *removed_shipments* **then**
      possible_trucks_for_insertion ← list of empty trucks with feasible docking deadline
    else
      possible_trucks_for_insertion ← list of trucks with feasible docking deadline, truck capacity, cooling requirements and shipment incompatibilities
    end if
    for each *truck* in *possible_trucks_for_insertion* **do**
      for each *position* in *truck* **do**
        Calculate *insertion_cost*
        Add (*insertion_cost*, *truck*, *position*, *shipment*) to *insertions*
      end for
    end for
    Sort *insertions* by *insertion_cost* in ascending order
    if *insertions* is not empty **then**
      if there are at least 2 insertions in *insertions* **then**
        regret_value ← difference between *insertion_cost* of the first two insertions in *insertions*
      else
        regret_value ← *insertion_cost* of the insertion in *insertions*
      end if
      Append (*regret_value*, best insertion) to *regret_values*
    end if
  end for
  if *regret_values* is not empty **then**
    Shuffle *regret_values*
    max_regret_value ← first max value in *regret_values*
    insertion_cost, *truck*, *position*, *shipment* ← Extract details from *max_regret_value*
    Add *shipment* to *truck* at *position* in solution string
    total_insertion_costs ← *total_insertion_costs*+ *insertion_cost*
    remove *shipment* from *removed_shipments*
  else
    return None
  end if
end while
return *insertion_costs*

3.3.6 Simulated Annealing Acceptance Criterion

Within the ALNS framework, the principle of SA is applied. This helps to balance exploration and exploitation of the solution space by probabilistically accepting inferior solutions to escape local optima. Accepting only superior solutions may lead to being trapped in a local optimum, thereby removing the opportunity to move towards a global optimum. This technique is inspired by the process of material annealing.

The approach works as follows: after a removal operator has removed shipments from the trucks in the solution and a insertion operator has reinserted them into the solution, a new solution is generated. If the

new solution is better than the previously obtained solution, it is always accepted. However, if the new solution is worse, it is either rejected or accepted based on a probability determined by an acceptance probability function. For this minimisation problem, the probability of acceptance is calculated using Equation 45. For maximisation problems, the same equation can be used but the terms *f(new)* and *f(old)* need to be swapped.

$$P(\textit{accept new solution}) = \begin{cases} e^{-\frac{(f(new)-f(old))}{T_k}}, & \text{if } f(new) > f(old) \\ 1, & \text{if } f(new) \leq f(old) \end{cases} \quad (45)$$

If the objective value of the new solution ($f(new)$) is smaller than or equal to the objective value of the previous solution ($f(old)$), the probability of accepting the new solution is 1. If the new solution's objective value is greater than that of the old solution, the probability of acceptance is determined by the formula with the corresponding inequality requirement shown in Equation 45, with T_k representing the state temperature of iteration k.

After determining the probability of acceptance, a random continuous number between 0 and 1 (excluding 1) is picked and compared with this probability. If the probability of acceptance is smaller than the random number, the solution is rejected. If it is greater or equal to the random number, the solution is accepted and becomes the current solution for the next iteration $(k + 1)$.

An important feature of this method is the decreasing temperature T_k over time. Lowering the temperature T_k reduces the probability of accepting solutions with a higher objective value. As the number of iterations progresses, the model becomes less probable to accept these solutions as the current solution. This mirrors the process of material annealing, where the temperature of the material gradually decreases. After each iteration, a new temperature is calculated using Equation 46:

$$T_{k+1} = \alpha \cdot T_k \quad (46)$$

The temperature decreases per iteration based on the temperature coefficient α, which is predefined. The model ceases execution after the predefined minimum temperature is reached.

3.3.7 Adaptive Weight Adjustment

The idea of adaptive weight adjustment, as described in the paper of Ropke and Pisinger (2006), which characterises ALNS, updates the operator weights based on their performance during each cycle of iterations. Operators that lead to better solutions are given higher weights, while those that perform poorly are assigned lower weights. The goal of this approach is to increase the probability of using more effective operators over time, allowing the algorithm to focus on the best performing search strategies.

At the start of a new cycle of iterations, all operator scores are reset to zero. After each iteration in a cycle, the scores of the chosen removal and insertion operators are increased by σ_1, σ_2, or σ_3 based on the outcome of the remove-insert operation, as defined by the logic of Ropke and Pisinger (2006), presented in Table 4:

Table 4: Score Adjustment Parameters (Ropke & Pisinger, 2006).

Parameter	Description
σ_1	The last remove-insert operation resulted in a new global best solution.
$\sigma 2$	The last remove-insert operation resulted in a solution that has not been accepted before. The cost of the new solution is better than the cost of current solution.
σ_3	The last remove-insert operation resulted in a solution that has not been accepted before. The cost of the new solution is worse than the cost of current solution but the solution was accepted.

After the completion of iteration cycle, consisting of a predefined number of iterations, the operator weights are updated individually using Equation 47:

$$w_{i,j+1} = w_{i,j} \cdot (1 - r) + r \cdot \frac{\pi_{i,j}}{\theta_{i,j}} \quad (47)$$

If an operator has not been selected during a cycle, Equation 48 is used to update the operator weight:

$$w_{i,j+1} = w_{i,j} \cdot (1 - r) \quad (48)$$

In these equations, $w_{i,j}$ is the weight of operator i during the latest completed j^{th} cycle, and r is the reaction factor $(0 < r \leq 1)$, which controls the influence of the latest cycle's performance on the updated weight. A larger value of r makes the weight update more sensitive to the operator's performance in the most recent cycle, while a smaller value of r puts greater emphasis on the weight of the operator during last cycle.

Parameter $\pi_{i,j}$ is the total score obtained by operator i during the j^{th} cycle. This score is the sum of operator i's scores over all iterations in the j^{th} cycle, determined using the scoring logic described in Table 4. The accumulated score $\pi_{i,j}$ is normalised by $\theta_{i,j}$, which is the total number of times operator i was used during the j^{th} cycle. This normalisation ensures that operators that are used less frequently are not penalised for lower total scores.

3.3.8 Hyperparameter Tuning

As the algorithm uses multiple parameters that should be initialised before running the algorithm, a parameter configuration has to be selected that positively affects the performance of the ALNS algorithm.

Algorithm parameters that need to be tuned are shown in Table 5:

Table 5: Algorithm Parameters.

Algorithm Parameter
Initial Temperature (T_{start})
Minimum Temperature (T_{min})
Temperature Coefficient (α)
Iterations per Cycle
Score New Global Best Solution (σ_1)
Score New Best Current Solution (σ_2)
Score New Worse Solution Accepted (σ_3)
ALNS Reaction Factor (r)
Percentage of Shipments To Be Removed
Time Penalty Cost Factor per Hour

An automated machine learning (AutoML) hyperparameter optimisation (HPO) tool has been selected for tuning these parameters. The working of the tool, called SMAC (Lindauer et al., 2022), short for Sequential Model-based Algorithm Configuration, will be explained below briefly. The SMAC tool uses the concept of Bayesian optimisation and starts by sampling a set of initial configurations of hyperparameters. After that, the tool evaluates each configuration by running the ALNS algorithm and measuring its performance based on the objective value. After the initial evaluations, SMAC builds a surrogate model using Random Forest (RF) to approximate the performance landscape of the hyperparameter configurations. This surrogate model helps to predict the performance of new configurations without the need to run the actual ALNS algorithm. Additionally, SMAC uses an acquisition function that interplays with the surrogate model to select new configurations to evaluate. The acquisition function balances the exploration of new configurations and the exploitation of promising configurations identified by the surrogate model. This is an iterative process, where each new evaluation helps to refine the surrogate model. The iterations continue until the maximum number of function evaluations has been reached.

By using the SMAC package in Python, the parameter space can be explored such that a combination of parameter values can be found that positively affect the performance of the ALNS algorithm. This Python package allows for systematic and effective tuning of the ALNS algorithm's parameters.

3.4 Model Differences

Before presenting the performance results of both models, this section highlights some minor differences in their setup and programming. Firstly, there is a difference in how shipments are split. In the ALNS model, all shipments are split according to the technique presented by Chen et al. (2017). In constrast, the MILP model permits unlimited shipment splitting, with the volume lower bound determined by the parameter e in constraint 26. Secondly, the models differ in whether they allow a station to be visited multiple times within a route. The ALNS model allows a truck to visit the same station multiple times within a route, while the MILP model restricts each station to being visited only once per route. However, revisiting a station is generally inefficient and unlikely to occur. Furthermore, the models differ in how docking time windows are determined. The MILP model employs continuous time windows, allowing for more precise scheduling, while the ALNS model uses on discrete, predefined time windows.

These differences in model setup result in different solution spaces, which impact the potential solution quality and performance of each model. The MILP model, with its unlimited shipment splitting and continuous docking time windows, explores a broader solution space, potentially leading to a better solution quality. Note that, while the restriction on station visits may seem limiting, it is unlikely to have a significant impact, as repeated visits lead to inefficient routes. In contrast, the ALNS model operates within a more constrained solution space due to predefined time windows and limited shipment splitting but benefits from increased computational efficiency. Consequently, while the MILP model may achieve solutions with a better quality, the ALNS model could provide more practical solutions within a reasonable computation time.

4 Description of the Case Study

To compare and evaluate the performance of the MILP and ALNS model, they are applied to different scenarios of a case study from an airline. The European trucking network for outbound air cargo of an airline has been used for this. The airline's network consists of approximately 50 stations located near airports in Europe.

At the hub airport, the airline has five available dock-doors which are connected to two cargo lifts. This means that two trucks can be processed at the same time. In addition, at most three other trucks can already start docking while the other two trucks are still being unloaded. When the other two trucks have finished unloading, the unloading of two other trucks that are already connected to a dock-door can be started immediately. Therefore, both models will consider that only two dock-doors are available but trucks can immediately start unloading once another preceding truck has finished unloading at a dock-door.

Shipments that need to be transported within the network have several time-bound constraints. Each shipment becomes available for pickup at a station in the network at a given time. However, shipments cannot be picked up at their origin stations at all times. This is determined by the station's opening hours. Although each truck can arrive 24/7 at the hub airport, each shipment must arrive at the hub airport before its predefined arrival deadline, which is determined by its departing flight. This is to ensure enough time is available to process the shipments at the hub airport and to transfer them to the correct flights.

Furthermore, the shipment data for each station is grouped by shipment type, with each group having a corresponding arrival deadline and pickup time. Since

three shipment types are considered in this case study, three shipment volumes per station are scheduled for daily transport, which can be distributed across multiple trucks.

Three types of trucks are considered in the scenarios of the case study: one non-refrigerated truck and two types of refrigerated trucks, each type having its own temperature range. The shipments scheduled for transportation must be assigned to a truck based on their specific temperature requirements. It is assumed that shipments which do not require refrigeration can be allocated to any of the three types of trucks. Furthermore, each truck has a volume capacity of $40m^3$ divided over four cargo positions, with each position assumed to have a capacity of up to $10m^3$ of cargo volume.

5 Results

After completing the development of the MILP and ALNS models and defining the case study, several simulations were run to analyse their performance and compare them. This section presents those results, beginning with Section 5.1, which details the hardware and software specifications used to obtain them. Subsequently, Section 5.2 discusses the model comparisons, followed by Section 5.3, which presents the results of running the ALNS model on a large instance.

5.1 Hardware and Software Specifications

The simulations that have been run to obtain the results presented in this section are executed on a Lenovo ThinkPad T16 Gen 2 with a 13th Gen Intel(R) Core(TM) i7-1365U processor running at 1.80 GHz and 32 GB of RAM. The laptop operates on a 64-bit Windows 10 operating system and has been used for all programming tasks and the execution of the algorithms.

The algorithms that are presented in the previous section are programmed in Python using version 3.12.2. The input data used in both models has been preprocessed using the Pandas 2.2.2 package, and the MILP formulation is solved by using the commercial solver Gurobi, version 11.0.3. Furthermore, the SMAC3 package (Lindauer et al., 2022), version 2.2.0, has been used for tuning the hyperparameters used in the ALNS algorithm.

5.2 MILP and ALNS Comparison

For the MILP and ALNS comparison, two datasets, each containing 28 data instances, are used to evaluate the performance of both models. Each instance contains 1 day of shipment requests from several stations in the network of the airline. The instances of dataset 1 contain shipment data from two market segments within the network, and the instances of dataset 2 includes two additional market segments. For all 56 instances, the truck routes and dock-door scheduling have been optimised using both models. First, Section 5.2.1 presents the results of the tuning process for the two parameter configurations, each optimised for one of the datasets. Section 5.2.2 describes the simulation stopping conditions for both models. After that, Section 5.2.3 presents the results of the model comparison.

5.2.1 Tuning Results for the ALNS Algorithm Parameter Configurations

As explained in Section 3.3.8, the parameters of the ALNS algorithm are tuned using the AutoML tool SMAC. For comparison of the MILP and ALNS models, the parameters of the ALNS model are tuned only once using one of the instances in each dataset. This approach is chosen because of the limited time available during the research project. To ensure effective functioning of the SA acceptance criterion, the instance with the highest initial cost is selected to tune the parameters. Tuning the model parameters using an instance with a lower initial cost could result in a parameter configuration with a relatively low starting temperature (T_{start}). Subsequently, for instances in the dataset with relatively high objective values, this could lead to a situation where the algorithm rarely accepts worse solutions throughout its execution. This would limit exploration of the solution space, which is something that should be avoided.

Table 6 presents the tuning ranges used during the tuning phase, along with the parameter configurations that resulted from running the SMAC tuning algorithm. After completing the tuning phase for the model comparison, the resulting configurations have been used for the simulations using the instances in their corresponding dataset.

Table 6: ALNS algorithm parameter tuning ranges and configurations for instance sets 1 and 2.

Algorithm Parameter	**Tuning Range Set 1**	**Tuning Range Set 2**	**Configuration for Instance Set 1**	**Configuration for Instance Set 2**
Initial Temperature (T_{start})	10,000-50,000	40,000-150,000	12,402	87,746
Minimum Temperature (T_{min})	5-500	10-10,000	55	1,395
Temperature Coefficient	0.9-0.999	0.99-0.999	0.9981	0.9990
Iterations per Cycle	5-20	5-20	5	12
Score New Global Best Solution (σ_1)	1-5	1-5	3.3489	2.6842
Score New Best Current Solution (σ_2)	0.1-1	0.1-1	0.6543	0.7625
Score New Worse Solution Accepted (σ_3)	0.1-1	0.1-1	0.1002	0.9998
ALNS Reaction Factor (r)	0.01-0.5	0.01-0.5	0.0101	0.0607
Percentage of Shipments to be Removed	0.001-0.1%	0.001-0.1%	7.245%	4.268%
Time Penalty Cost Factor per Hour	200-10,000€/h	400-10,000€/h	7,935€/h	7,183€/h

5.2.2 Simulation Stopping Conditions and Solution Stability

Before executing the MILP model, the running time and optimality gap must be constrained. As the model is NP-hard, a running time limit is needed to avoid long running times, despite using relatively small data instances for the comparison. The running time limit is set to 1800 seconds. Furthermore, the optimality gap constraint has been set to 3%. The MILP model has been run once for each instance. The ALNS model needs to be run multiple times because its solutions vary with each run due to the random factor that influences the selection of removal and insertion operators. To gather sufficient data and to evaluate the model's performance, the coefficient of variation is used to determine the number of simulation runs required. The coefficient of variation (c_v) is determined by using Equation 49:

$$c_v = \frac{\sigma(Avg\ Obj)}{\mu(Avg\ Obj)} \tag{49}$$

In this formula, $\sigma(Avg\ Obj)$ represents the standard deviation of the average objective value, measuring the spread or variability of the objective values around the mean, denoted by $\mu(Avg\ Obj)$. After each simulation run with a different random seed, the coefficient of variation is recalculated for each instance. When the coefficient of variation stabilises, it indicates that the required number of simulation runs has been reached to obtain a reliable estimate of the distribution of the model's average objective value output. For each instance, the algorithm was run at least 10 times until the change in the coefficient of variation of the objective value was less than 0.001 for at least five consecutive simulation runs.

5.2.3 Model Comparison Results

Table 7 and Table 8 show the results for optimising the truck routes and dock-door schedules of the instances in dataset 1 and 2, respectively. Both tables are divided in three parts: the instance characteristics, the optimisation results of the MILP model, and the optimisation results of the ALNS model. Furthermore, the last column in both the tables present the percentage difference between the minimum objective value obtained by the ALNS model and the best incumbent solution found using the MILP model, relative to the latter.

First of all, for all instances in dataset 1, both the MILP and ALNS model succeeded in finding a feasible solution. Furthermore, for all instances, the minimum objective values found by the ALNS are greater than or equal to the best bound found by the MILP model. If this had not been the case, ALNS found a solution that is better than what the MILP guarantees as achievable at that point. This would indicate potential issues with the constraints in the MILP model or the accuracy of the ALNS model's solutions, which could result in MILP solutions of lower quality or infeasible ALNS solutions.

For more than 70 percent of the instances, the MILP model was not able to reach the 3 percent optimality gap threshold before reaching the running time limit. Furthermore, the MILP model achieved relatively low computational times only for instances 3 and 4, with times of 5.24 and 1.04 seconds respectively. Moreover, for instances where the MILP model reached the defined optimality gap before reaching the running time limit, the combination of the number of stations in the network, number of shipments, and the total shipment volume was relatively low compared simulations of instances that were terminated after reaching the running time limit. This is expected, as the number of constraints in the MILP model significantly decreases with fewer stations, shipments, and a lower total shipment volume, which requires fewer trucks to be modelled.

The last column in the table shows the percentage difference between the minimum objective value found by ALNS model and MILP model's best incumbent solution, which highlights how closely ALNS approaches the MILP's objective values. In a significant amount of cases, this difference is relatively small ($< 2\%$). For instances with a high MILP optimality gap ($> 75\%$), ALNS provides solutions that are often close to the MILP's best incumbent solutions but with significantly less computational time required. Moreover, the ALNS model generally runs faster on average compared to the MILP model's running time limit. Furthermore, for instances where the MILP model finds the optimal solution, indicated by an optimality gap of zero (instances 9 and 23), the ALNS model also performs well and matches the MILP results.

The variability in ALNS solutions, indicated by the standard deviation of the objective value, indicates how consistent the heuristic is across multiple runs. This increases for more complex instances with a relatively high number of shipments, stations and total shipment volume. These instances also needed more runs for the coefficient of variation to converge.

Table 7: Simulation results of instance set 1 for the MILP and ALNS model comparison.

Instance	Instance Characteristics			MILP				ALNS					Min Objective - Best Incumbent (%)
	# Stations*	# Shipments	Total Shipment Volume (m^3)	Best Bound	Best Incumbent	Gap (%)**	Run Time (s)***	Min Objective	Max Objective	Average Objective	Standard Deviation	Average Run Time (s) / # Runs	
1	4	8	208.53	3,584.07	3,693.54	2.96	1,169.12	3,726.45	3,739.83	3,730.50 ± 5.67		26.01 10	0.89%
2	5	7	162.3	527.80	3,186.17	83.43	1,801.76	3,186.17	3,186.17	3,186.17 ± 0.00		39.17 10	0.00%
3	3	5	226.51	2,761.93	2,842.42	2.83	5.24	2,899.54	2,899.54	2,899.54 ± 0.00		45.49 10	2.01%
4	2	5	334.82	2,607.28	2,685.06	2.90	1.04	2,685.39	2,720.89	2,690.97 ± 12.60		17.46 11	0.01%
5	3	6	250.51	2,759.74	2,843.38	2.94	18.27	2,843.38	2,843.38	2,843.38 ± 0.00		28.93 10	0.00%
6	4	7	461.43	1,672.45	3,260.14	48.70	1,800.96	3,474.26	3,503.59	3,488.83 ± 12.99		29.20 10	6.57%
7	4	7	375.27	438.80	2,796.04	84.31	1,801.25	2,802.54	2,810.60	2,807.40 ± 3.51		33.10 10	0.23%
8	4	8	129.14	2,014.37	2,069.84	2.68	137.80	2,281.61	2,294.63	2,282.91 ± 4.11		39.51 10	10.23%
9	4	5	131.18	2,925.52	2,925.52	0.00	74.80	2,925.52	2,927.10	2,926.47 ± 0.81		40.75 10	0.00%
10	4	7	242.92	392.31	1,866.81	78.99	1,801.24	2,059.13	2,066.39	2,063.59 ± 3.62		25.24 10	10.30%
11	3	7	468.79	1,780.20	1,928.48	7.69	1,801.24	2,080.75	2,119.67	2,097.95 ± 16.45		35.52 10	7.90%
12	5	8	446.1	281.30	5,859.55	95.20	1,802.68	5,890.67	6,354.68	6,156.50 ± 208.67		40.09 20	0.53%
13	5	8	414.99	10.57	4,584.89	99.77	1,801.94	4,628.44	4,920.78	4,820.91 ± 109.03		41.21 19	0.95%
14	4	7	405.28	262.53	2,204.11	88.09	1,802.44	2,233.54	2,262.14	2,258.06 ± 10.39		36.89 14	1.34%
15	4	7	314.69	618.70	2,858.52	78.36	1,801.52	2,889.56	2,889.56	2,889.56 ± 0.00		28.34 10	1.09%
16	5	6	223.51	538.47	3,417.65	84.24	1,802.99	3,417.65	3,477.86	3,447.75 ± 31.44		48.12 12	0.00%
17	4	7	229.93	2,789.53	2,868.84	2.76	438.83	2,904.01	2,904.01	2,904.01 ± 0.00		28.82 10	1.23%
18	4	7	346.07	551.71	2,500.37	77.93	1,803.60	2,526.27	2,532.57	2,531.31 ± 2.66		44.45 10	1.04%
19	5	8	357.99	351.63	4,311.64	91.84	1,801.77	4,373.36	4,467.89	4,440.97 ± 42.18		46.38 11	1.43%
20	5	8	374.23	0.00	5,177.02	100.00	1,801.93	5,582.09	5,585.19	5,583.95 ± 1.60		45.35 10	7.82%
21	5	7	606.31	40.79	3,399.33	98.80	1,801.78	3,399.33	3,921.51	3,674.67 ± 234.80		41.93 23	0.00%
22	4	8	172.77	774.44	2,024.03	61.74	1,801.07	2,230.47	2,244.13	2,240.44 ± 5.61		60.66 10	10.20%
23	4	5	152.11	3,139.15	3,139.15	0.00	106.97	3,139.15	3,146.05	3,145.36 ± 2.18		47.85 10	0.00%
24	4	7	342.31	1,810.26	2,504.50	27.72	1,801.15	2,509.55	2,509.55	2,509.55 ± 0.00		48.91 10	0.20%
25	4	8	503.92	129.35	3,213.91	95.98	1,802.66	3,412.04	3,413.59	3,413.12 ± 0.75		44.94 10	6.16%
26	4	6	420.22	512.43	4,468.04	88.53	1,801.96	4,491.31	4,500.18	4,494.42 ± 2.41		45.46 10	0.52%
27	5	8	364.68	1,044.10	3,990.26	73.83	1,801.80	4,181.70	4,181.70	4,181.70 ± 0.00		53.08 10	4.80%
28	4	7	296.92	2,666.29	3,045.59	12.45	1,801.02	3,046.50	3,103.89	3,090.87 ± 18.12		53.19 15	0.03%

*Excluding depot, **Optimality gap threshold = 3%, ***CPU limit = 1800s

After the algorithmic comparison, a managerial comparison will be provided based on three network characteristics: the average truck load factor, the average number of legs per truck route, and the total number of shipments after splitting. The results used for the comparison are presented in three box plots in Figure 4.

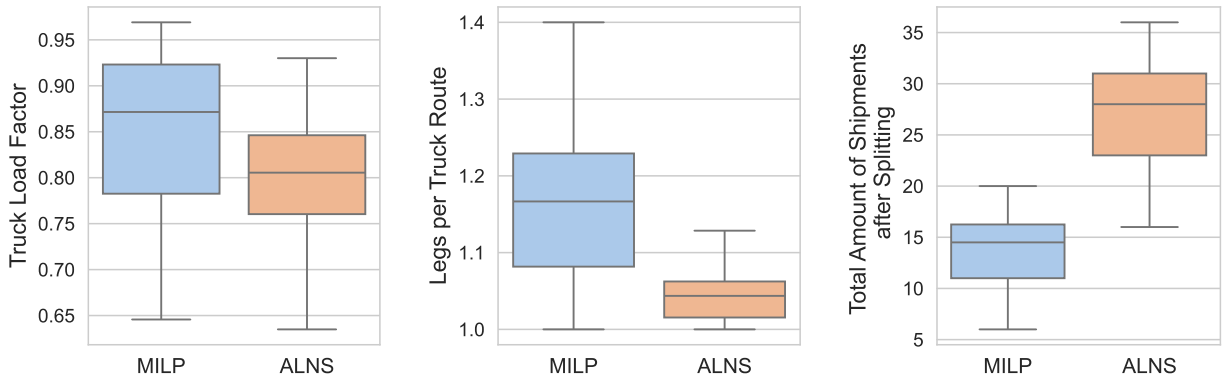
Figure 4a shows the average truck load factor resulting from the MILP and ALNS models. While the instance characteristics vary, the results indicate that MILP generally achieves a higher average truck load factor. This means that, for small instances, the MILP model optimises truck utilisation more effectively than the ALNS model, leading to better use of volume capacity in transportation.

Furthermore, Figure 4b presents the average number of legs per truck route for both the MILP and ALNS models. The box plot shows that the MILP model's solutions generally result in a higher number of legs per truck route compared to the results of the ALNS model. This means that the MILP model plans more truck routes with intermediate stops. In contrast, ALNS tends to generate routes with fewer legs, meaning that routes with more direct deliveries are used. Furthermore, the figure shows that both models solved instances using only truck routes with one leg per route, as indicated by the minimum value of 1.

Lastly, Figure 4c illustrates the total number of shipments after splitting. It is evident that the ALNS solutions use a significantly higher number of shipments compared to the MILP model. This can be explained by the differences in how both models operate. The ALNS model uses an a priori splitting technique based on a set of predefined volume thresholds, irrespective of the network characteristics. On the other hand, the MILP model splits the shipments during the optimisation process based on what splitting decisions contribute to achieving a lower objective value.

These results are in line with the results shown in Table 7. Overall, the MILP model performs similarly to or better than the ALNS model. A higher average

truck load factor and a larger number of legs per truck route contribute to more efficient utilisation of truck capacities, leading to lower objective values achieved by the MILP model.



(a) Average truck load factor.

(b) Average legs per truck route.

(c) Total amount of shipments after splitting.

Figure 4: Distribution of network characteristic results of instance set 1.

The second table, Table 8, presents the simulation results for the instances in dataset 2, which includes two additional market segments compared to dataset 1. For most of the instances, the MILP model did not succeed in finding a feasible solution within the given running time limit. Therefore, no best incumbent objective value is available for instances 32-56. A feasible solution was found only for the first three instances, all characterised by a relatively low total shipment volume ($< 400m^3$). Furthermore, the MILP model found a best bound for all instances, however, all except instance 31 have a value of zero.

Since no incumbent solution has been found for instances 32-56, the solver has not yet identified a feasible integer solution for any of those instances. The best bound of zero means that, in the relaxation of the continuous decision variables, the lowest possible objective value found so far is zero. However, this does not imply that a feasible integer solution achieving zero exists, only that the relaxation allows for such values. The use of the Big M formulation in multiple MILP constraints could be causing numerical instability or weak relaxations, making it harder for the solver to find feasible integer solutions or tighten the bounds effectively.

On the other hand, the ALNS model found a feasible solution for all instances within a maximum of four and a half minutes. Moreover, the majority of the instances was solved within two minutes. The ALNS outperforms the MILP model for all instances except for instance 29, for which the minimum objective value found is approximately 4.75% higher than the best incumbent solution of the MILP model.

Compared to the performance of the ALNS model on the instances of dataset 1, the number of simulations needed to converge to a stable coefficient of variation is generally higher, as is the standard deviation of the objective value found. However, from the results of datasets 1 and 2, it is evident that the more complex the instances are, the better the ALNS model performs in terms of computational time and objective value compared to the MILP model.

Table 8: Simulation results of instance set 1 for the MILP and ALNS model comparison.

Instance	Instance Characteristics			MILP				ALNS					Min Objective - Best Incumbent (%)
	# Stations*	# Shipments	Total Shipment Volume (m^3)	Best Bound	Best Incumbent	Gap (%)**	Run Time (s)***	Min Objective	Max Objective	Average Objective	Standard Deviation	Average Run Time (s)	# Runs	
29	9	17	344.54	0.00	13,316.34	100.00	1,801.19	13,949.10	15,472.02	14,438.52	± 469.40	153.82	29	4.75%
30	10	14	342.68	0.00	16,123.02	100.00	1,800.86	15,945.79	17,105.24	16,664.34	± 351.50	215.07	15	-1.10%

Table 8 continued from previous page

31	9	12	397.8	316.47	21,423.60	98.53	1,800.68	18,228.36	19,929.96	18,978.14 ± 381.09	107.81	21	-14.91%
32	9	16	532.5	0.00	-	-	1,801.15	17,734.88	18,932.52	18,197.46 ± 396.38	166.60	17	-
33	8	16	499.3	0.00	-	-	1,801.13	20,298.13	20,775.49	20,521.31 ± 160.88	177.50	11	-
34	9	16	654.92	0.00	-	-	1,801.05	16,592.49	18,085.11	17,127.11 ± 390.79	135.39	16	-
35	10	16	652.61	0.00	-	-	1,802.72	20,343.05	21,970.96	21,179.87 ± 472.17	152.28	17	-
36	10	19	295.8	0.00	-	-	1,801.07	14,784.42	16,267.47	15,815.33 ± 420.21	216.65	22	-
37	11	16	644.8	0.00	-	-	1,802.79	32,632.20	34,199.38	33,638.99 ± 533.05	250.23	15	-
38	12	18	458.46	0.00	-	-	1,802.19	22,252.60	23,282.66	22771.72 ± 384.59	217.96	16	-
39	8	13	733.75	0.00	-	-	1,801.01	16,427.63	17,029.68	16,703.42 ± 227.25	74.30	10	-
40	11	18	697.72	0.00	-	-	1,801.64	23,774.18	25,842.91	24,697.57 ± 683.66	130.71	21	-
41	10	18	757.68	0.00	-	-	1,802.80	23,829.42	25,520.45	24,847.09 ± 597.34	86.01	17	-
42	9	17	957.94	0.00	-	-	1,801.61	34,391.30	35,952.08	35,007.18 ± 518.14	85.62	13	-
43	10	21	546.99	0.00	-	-	1,810.88	21,635.13	22,892.19	22,551.3 ± 459.80	108.48	17	-
44	13	16	560.7	0.00	-	-	1,801.44	24,626.48	25,376.52	25,023.66 ± 212.74	122.12	10	-
45	11	17	492.1	0.00	-	-	1,801.87	24,216.61	24,364.69	24,331.33 ± 59.08	117.70	10	-
46	9	14	394.69	0.00	-	-	1,801.04	8,500.00	9,360.60	9,116.196 ± 250.53	89.87	20	-
47	9	17	549.25	0.00	-	-	1,801.16	17,940.64	19,353.82	19,076.73 ± 369.90	98.42	14	-
48	10	17	570.89	0.00	-	-	1,802.83	19,912.18	21,457.77	20,992.56 ± 486.48	84.86	21	-
49	10	17	876.3	0.00	-	-	1,809.31	22,320.17	23,532.83	22,876.89 ± 440.87	82.30	15	-
50	10	19	501.93	0.00	-	-	1,801.17	21,766.49	21,939.00	21,841.02 ± 66.78	120.15	10	-
51	11	15	296.75	0.00	-	-	1,801.13	17,529.87	18,098.53	17,878.87 ± 203.52	140.36	12	-
52	11	18	567.52	0.00	-	-	1,802.98	20,964.21	21,956.84	21,505.61 ± 362.57	126.05	11	-
53	10	16	627.35	0.00	-	-	1,801.64	17,479.59	18,326.07	17,802.78 ± 206.57	107.61	13	-
54	11	19	699.64	0.00	-	-	1,801.78	29,630.57	30,230.83	29,928.29 ± 180.21	122.54	10	-
55	10	18	640.54	0.00	-	-	1,806.87	21,166.90	22,045.81	21,414.82 ± 289.00	99.26	11	-
56	9	16	608.96	0.00	-	-	1,801.15	20,823.98	22,140.70	21,519.65 ± 409.86	97.01	17	-

*Excluding depot, **Optimality gap threshold = 3%, ***CPU limit = 1800s

5.3 ALNS Results for a Large Instance

This section analyses the performance of the ALNS algorithm using a large instance. Instance 57, covering a full week of all shipment requests across the entire network of the airline, is used for this analysis. The instance's network consists of 40 stations (including the depot) and contains 361 unique shipment requests with a total volume of 12,769m^3. Section 5.3.1 will present the parameter configuration used for running the ALNS model on instance 57. After that Section 5.3.2 and Section 5.3.3 will discuss the model's general performance and the performance of the operators used in the model. Finally, Section 5.3.4 presents a comparison between the airlines actual network and the model-generated network obtained using the ALNS algorithm.

5.3.1 Tuning Results for the ALNS Algorithm Parameter Configuration

The tuning process for this instance follows the same procedure as for the small instances, as described in Section 5.2.1. Table 9 presents the tuning ranges used during the tuning phase, along with the parameter configurations that resulted from running the SMAC tuning algorithm. The model results presented in the next section were obtained by running the model with this specific parameter configuration.

Table 9: ALNS algorithm parameter tuning range and configuration for a week of shipment transportation requests.

Algorithm Parameter	Tuning Range	Configuration
Initial Temperature (T_{start})	500,000-1,000,000	841,997
Minimum Temperature (T_{min})	1-400,000	56
Temperature Coefficient	0.99-0.999	0.9911
Iterations per Cycle	5-20	10
Score New Global Best Solution (σ_1)	1-5	4.9984
Score New Best Current Solution (σ_2)	0.5-1	0.9230
Score New Worse Solution Accepted (σ_3)	0.1-0.5	0.4099
ALNS Reaction Factor (r)	0.01-0.5	0.2819
Percentage of Shipments to be Removed	0.001-0.1%	3.912%
Time Penalty Cost Factor per Hour	500-10,000€/h	5,345€/h

5.3.2 General Algorithm Performance

To determine the number of runs needed for this analysis, the algorithm was run until the change in the coefficient of variation of the objective value was less than 0.00025 for at least five consecutive simulation runs. For this instance some simulations returned a best solution that still contained a time penalty cost, which means that the solution was infeasible. These run results were neglected in the calculation for the coefficient of variation. A total of 84 simulations were run, of which 38 found a feasible solution with a time penalty cost of 0. Table 10 summarises the general simulation results of the runs that resulted in a feasible solution.

Table 10: General simulation results of the runs that found a feasible solution for instance 57.

Variable	Result
Average Run Time ± st. dev.	4,112.64s ± 695.59s
Number of Iterations per Run	1,079
Number of Simulations	84
Simulations with Feasible Solution	38
Simulations with Infeasible Solution	46

The distribution of total costs, including the average, as well as the upper and lower bounds of the total costs for the feasible solutions found for instance 57, is presented in Figure 5 and Table 11.

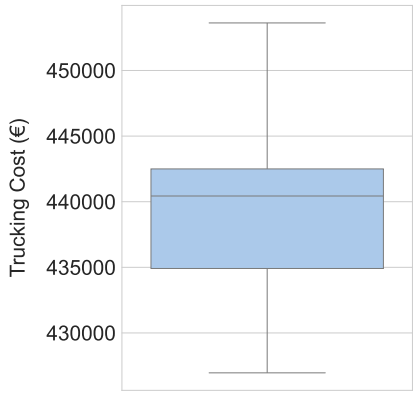


Figure 5: Distribution of the total costs of the feasible solutions found for instance 57.

Table 11: Summary of the total costs of the feasible solutions found for instance 57.

Variable	Result
Average Total Cost ± st. dev.	439,465.11€± 5,958.27€
Median Total Cost	440,435.10€
Minimum Total Cost	426,959.33€
Maximum Total Cost	453,623.04€

After running the model, the model outputs six graphs that illustrate the behaviour of various costs and operator weights as the number of iterations during a run progresses. Specifically, The first four graphs show the trucking cost, time penalty cost, the combined total of both costs and the direction of total cost changes over iterations. An example of these graphs is shown, on the next page, in Figure 7, representing the results from the run that found the minimum total cost.

Figure 7 shows that during the first phase of the run, the model tries to bring down the time penalty costs. This behaviour can be explained by the value of the time penalty cost factor, shown in Table 9. As the penalty per hour is relatively high compared to the trucking costs per leg in the network, the model can save more costs by removing shipments that impose a time penalty cost on the total costs. At the same time, the trucking costs increase as a result of moving shipments that imposed a time penalty cost to other trucks where they do not impose a time penalty cost. In this case, the gain from a decreasing time penalty offsets the increasing trucking cost.

After the majority of the initial time penalty costs have been reduced, the graphs show that the model begins to reduce the trucking costs. The graphs also indicate that at certain points, the model accepts solutions with a higher total cost compared to the total cost of the best solution found so far. However, the frequency of the model accepting worse solutions decreases as the number of iterations increases, which is shown by the last graph in Figure 7. This behaviour is a consequence of the simulated annealing concept used in the model.

Figure 7 also shows the iteration at which the total cost was last reduced by at least 0.1%, illustrated by the green vertical dotted-dashed line. Furthermore, the vertical red line indicates the first iteration where the solution with the minimum total cost was found.

The performance in terms of the two variables mentioned in the previous paragraph is further illustrated in Figure 6. This figure presents the distribution of both variables across all runs that resulted in a feasible solution, providing insight into their variability. As shown in Table 10, the total number of iterations per run is 1079. Figure 6 shows that in most runs the best solution is found near the end of the run, close to the final iteration. The other variable represents the last iteration in which the total cost of the best solution decreased by at least 0.1%. For this instance, based on the average trucking cost, this corresponds to a trucking cost reduction of approximately €440.

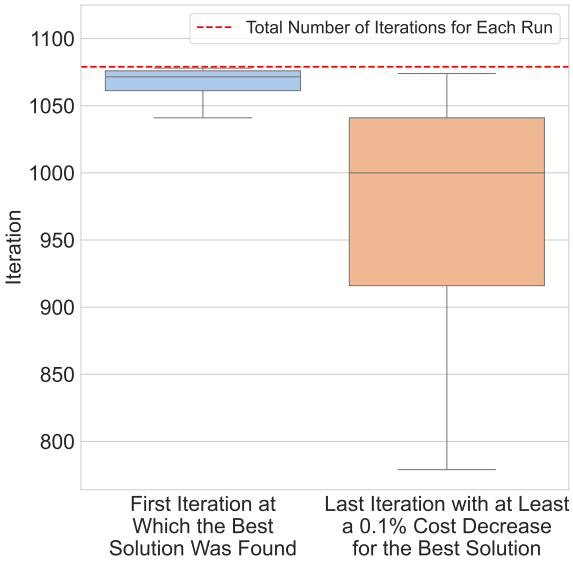


Figure 6: Distribution of iterations at which the best solution was first found and the last iteration where the best solution's total cost decreased by at least 0.1%, over all feasible solutions obtained for instance 57.

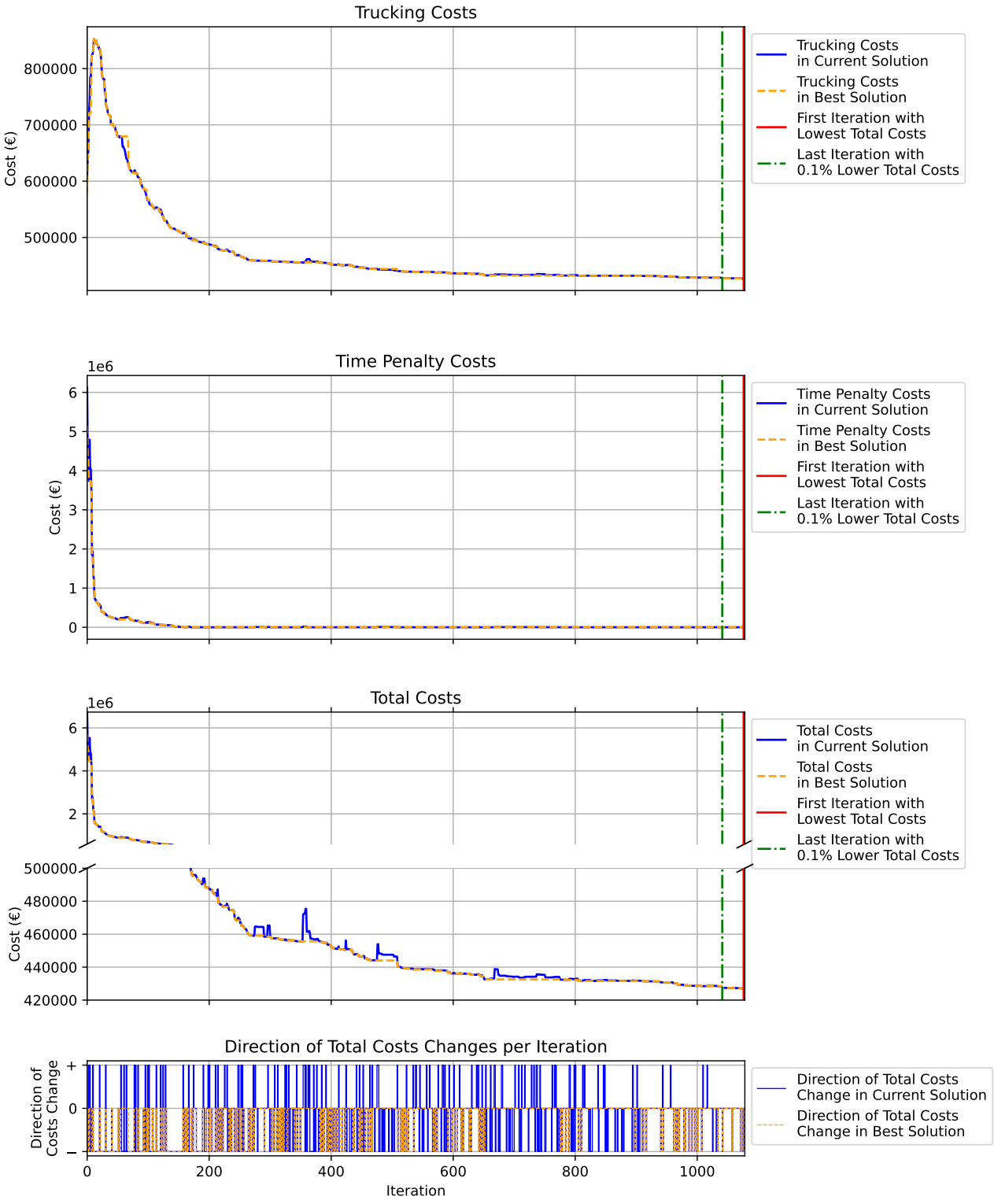


Figure 7: Cost progression during the run that found the minimum cost value for instance 57.

As shown in Table 10, 46 out of the 84 simulation runs resulted in a time penalty cost greater than zero, indicating unsuccessful attempts at finding a feasible solution. The time penalty cost is a measure of the total time difference relative to the feasible operational conditions for each truck and station combination, indicating a degree of infeasibility. Figure 8 shows the distribution of the time penalty costs of the 46 infeasible solutions found for instance 57. The figure shows a median time penalty cost of 1,292€, which corresponds to a total infeasible operational time difference of approximately 15 minutes. Given that the average travel time across all generated routes is around 13.5 hours, this represents a relatively low infeasible time difference. As a result, one could argue that time penalty costs in this order of magnitude could be acceptable in practical scenarios. So, while these solutions are technically infeasible under strict constraints, they may still

be viable in real-world operations where minor deviations can be accommodated.

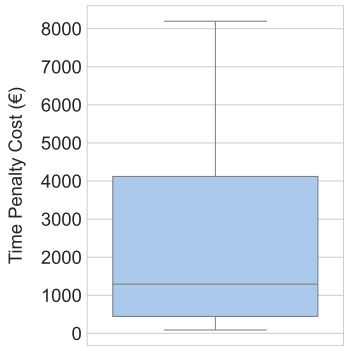


Figure 8: Distribution of the time penalty costs of the infeasible solutions found for instance 57.

To further examine the effect of changing the time penalty cost factor on the performance of solution feasibilities, the ALNS model has been run using the same seed as for the run that found the minimum total cost value but for different values of the time penalty cost factor. Table 12 presents the results of those runs. Although, the model has been run once for each value time penalty cost factor, shows that for values lower than 10,000€/h the model is more likely to found infeasible solutions.

Additionally, the results in the table show that while increasing the time penalty cost factor lowers the chance on finding infeasible solutions, the trucking cost increases. Relatively high penalty factors can restrict exploration of the solution space, which can lead to solutions of lower quality. However, lower penalty factors increase the likelihood of finding in infeasible solutions, which emphasises the importance of choosing an appropriate time penalty cost factor to balance feasibility and cost efficiency.

Furthermore, the lowest total cost found in these runs (423,208.31€ at a penalty factor of 10,000€/h) is lower than the solution found with the tuned penalty factor of 5,345€/h, which resulted in a total cost of 426,959.33€. This indicates that the tuning process may not have identified an appropriate or effective setting.

Table 12: Time penalty cost factor sensitivity for instance 57.

Time Penalty Cost Factor (€/h)	Total Cost (€)	Trucking Cost (€)	Time Penalty Cost (€)
1000	433,780.09	432,713.42	1,066.67
2500	435,745.62	435,745.62	0.00
4000	429,383.00	429,383.00	0.00
5000	439,883.63	439,050.30	833.33
6000	443,638.87	443,638.87	0.00
7500	458,185.12	441,685.12	16,500.00
10000	423,208.31	423,208.31	0.00
25000	444,862.44	444,862.44	0.00
50000	449,897.45	449,897.45	0.00
100000	446,378.91	446,378.91	0.00

5.3.3 Performance ALNS Operators

The performance and selection of each removal and insertion operator vary per run. The distribution of the total usage counts per operator over all the runs that found a feasible solution is captured in Figure 9. It illustrates the variation in selection frequency for each insertion and removal operator. The insertion operators, indicated by the green colour, show that the operators "Random Order at Best Position" and "Highest Position 2-Regret at Best Position" are the most frequently used. On the other hand, "Random Truck at Best Position" is rarely selected, indicating its limited success and effectiveness in constructing new solutions. Among the removal operators, shown in red, the operators "Random Shipments", "Highest Cost", and "Shaw" show a significant high usage count during the simulations. Note that the variability of the first two removal operators is significantly higher compared to the other operators. In contrast, the operators "All Shipments from Random Trucks" and "All Shipments from Random Trucks with Time Penalty" are used less frequently.

Figure 9: Distribution of total usage counts per operator over all the runs that found a feasible solution for instance 57.

As explained, the frequency of operator selection changes throughout each run. This selection is closely linked to the weights assigned to the operators, which are adjusted after each cycle of iterations. The adjustment interval is defined during the parameter tuning phase and set to 10 iterations before updating the operator weights.

In addition to the four graphs shown in Figure 7, two additional graphs illustrate how the removal and insertion operator weights change throughout a simulation run. Figure 10 presents an example of how the operator weights change after each cycle of iterations, specifically from the run that found the minimum total cost. The removal operator plot shows high initial weights for the "Random Shipments" and "Highest Cost" removal operators. However, their weights gradually decrease as iterations proceed. The "Shaw" operator and "All Shipments from Random Trucks" operator show fluctuating weights, highlighting their dynamic role in different stages of optimisation. The "All Shipments from Random Trucks" operator even approaches zero as the number of completed iterations increases. Despite high weights in the initial stage of the run, the 'All Shipments from Trucks with Time Penalty' operator quickly decreases and approaches zero, due to a significant reduction in time penalty costs, as shown in Figure 7. This decrease limits the number of shipments requiring removal, thus reducing the options for finding an improved solution. The insertion operator plot show that the "Random Order at Best Position" and "Highest Position 2-Regret at Best Position" operators initially have the highest weights, indicating their early effectiveness. However, their weights gradually decrease as the number of iterations increases. In contrast, the weight of the 'Random Order Random Truck at Best Position' operator decreases from the beginning and approaches zero, indicating its limited impact on solution quality.

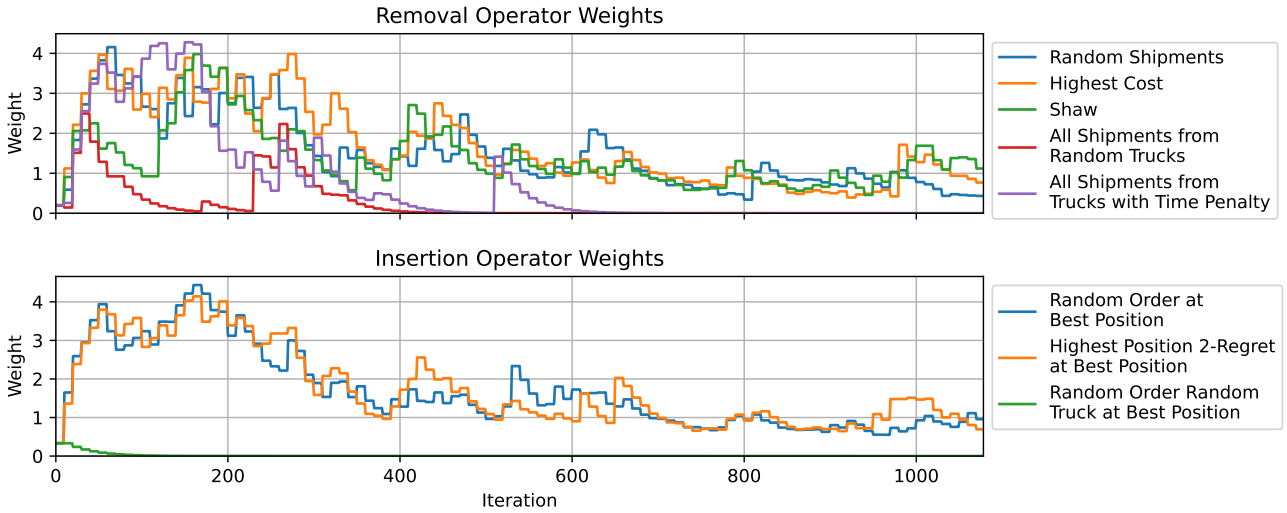


Figure 10: Operator weights during the run that found the minimum value for instance 57.

5.3.4 Network Characteristic Results

This subsection compares the historical routing data of the airline's actual network with the model-generated network obtained using the ALNS algorithm, based on the same historical shipment requests. As shown in Table 13, the optimised model significantly reduces trucking costs by up to 19.7%, decreases the number of required trucks by 31.2%, and increases the average load factor from 63.1% to 85.8%, demonstrating a more efficient routing network. Additionally, the average number of legs per route increases slightly from 1 to 1.153, indicating that every 6th to 7th truck adds an extra stop on its route to the hub. This outcome was expected, as the actual network did not use multi-leg routes. These optimisations lead to lower trucking costs, improved fleet utilisation, and reduced fuel consumption, as trucking costs are a function of the distance travelled. However, increasing the average load factor and decreasing the number of trucks used in the network does not necessarily result in lower trucking costs, as factors such as route length and detours can impact total costs.

The ALNS model balances cost efficiency and operational constraints and has the potential to improve the performance of the airline's trucking network. However, as discussed in Section 3.1, several assumptions are made in the routing and scheduling model that differ from real-world operations.

First, in the actual network, truck scheduling at dock doors has not been implemented, leading to waiting times that occur in practice but are not accounted for in the model. Second, while the model enforces a strict maximum volume capacity for trucks, deviations from this capacity sometimes occur in practice.

Additionally, the model allows for splitting total shipment volumes based on shipment type and origin station, whereas, in the actual network, shipment requests are handled individually. Since the model relies on forecasted total volumes per shipment type and origin, it does not have access to exact shipment-specific volumes.

Lastly, extra operational costs resulting from extra stops in a route are not considered in the model. Furthermore, some actual truck routes are not always feasible, as certain shipments did not arrive on time to meet their required delivery deadlines.

When comparing the results, it is important to keep the assumptions in mind. The model generates a routing and scheduling plan based on an idealised scenario that follows these assumptions. However, real-world operations often deviate from this idealised scenario.

Table 13: Comparison of actual network characteristics and model-generated network characteristics.

	Actual Network Characteristics	Model Generated Network Characteristics		
		Solution with Lowest Trucking Costs	Solution with Lowest Number of Trucks	Solution with Highest Average Load Factor
Trucking Costs (€)	531,399.74	426,959.32	437,640.66	437,640.66
Number of Trucks	542	372	359	359
Average Load Factor (%)	63.1	85.8	88.9	88.9
Average Number of Legs per Route	1.000	1.153	1.212	1.212
Average Route Distance (km)	694.32	738.38	773.47	773.47

6 Conclusions

This study focused on designing an integrated vehicle routing and dock-door scheduling model to optimise a airline's trucking network by increasing truck load factors, reducing operational costs, and ensuring on-time cargo delivery to the hub, to match with outbound flight legs. A novel Mixed Integer Linear Programming (MILP) model and Adaptive Large Neighbourhood Search (ALNS) model for a vehicle routing and dock-door scheduling problem with split delivery, incompatible products, time windows and open routes, was presented.

The comparative analysis of the MILP and ALNS models showed that while the MILP model performs better in achieving near-optimal solutions for small instances, the ALNS model outperforms the MILP model in terms of computational efficiency for more complex instances. However, for a significant amount of the small instances, the difference in objective value is between 0% and 2%, which is remarkable when considering that the ALNS model works with a smaller solution space compared to the MILP model because of the a priori splitting technique and pre-determined discrete time windows used in the ALNS model. Moreover, the ALNS model consistently finds feasible solutions within significantly less computational time, even for larger and more complex instances, where as the MILP model is not able to find feasible solutions within a reasonable computational time frame. The ALNS model shows a better performance in terms of running time and solution quality, making it a more useful alternative for large and complex instances where the MILP model results show high optimality gaps or do not find feasible solutions.

From the results of running the ALNS model for a large instance, it can be concluded that it is important to make a trade-off between solution quality and the likelihood of finding feasible solutions, which is influenced by the chosen time penalty cost factor in the hyperparameter configuration. Additionally, the results indicate that the last solution improvement of the trucking costs in most simulation runs occurred near the end of the simulation. This shows that there is still potential for improving the solutions by either increasing the temperature coefficient or lowering the minimum temperature in the simulated annealing process. Furthermore, the results show that the configuration of the hyperparameters derived from the tuning process did not yield the best-known solution. The best-known solution was found later during the investigation of the time penalty cost factor's sensitivity on the model's performance. This stresses the importance of determining an appropriate configuration setting as well as the significant influence of the chosen configuration on the model's performance.

The performance indicators of the removal and insertion operators in the ALNS model show that the insertion operators "Random Order at Best Position" and "Highest Position 2-Regret at Best Position" are most frequently used, whereas insertion operator "Random Truck at Best Position" is rarely selected, indicating its limited effectiveness, caused by its random nature. The removal operators, "Random Shipments", "Highest Cost", and "Shaw" are frequently used, with significant variability in their usage. In contrast, the other removal operators, "All Shipments from Random Trucks" and "All Shipments from Random Trucks with Time Penalty", are used less frequently and tend to drop in weights as the number of iterations progresses. This can be explained by the fact that these operators disrupt the solution by removing all shipments from a truck, which may already be forming a relatively low cost toward the end of the simulation runs. Additionally, the number of trucks with a time penalty are more likely to approach zero, making the latter operator less useful. Furthermore, the dynamic adjustments of the operator weights during a simulation highlight the importance of tuning the corresponding operator scores and iteration cycle parameters to enhance the model's performance.

In the ALNS model, the insertion operators "Random Order at Best Position" and "Highest Position 2-Regret at Best Position" are most frequently used, while "Random Truck at Best Position" is rarely se-

lected due to its limited effectiveness. Frequently used removal operators include "Random Shipments, "Highest Cost, and "Shaw, with significant variability in their usage. In contrast, "All Shipments from Random Trucks" and "All Shipments from Random Trucks with Time Penalty" are less frequently used and their weights decrease as the number of iterations progresses. This is because these operators destroy solutions by removing all shipments from a truck, which may already have a low cost towards the end of simulations. Additionally, trucks with time penalties tend to diminish, reducing the latter operator's usefulness. Dynamic adjustments of operator weights emphasise the need for tuning operator scores and iteration parameters to enhance model performance.

Finally, when comparing the model-generated network with the historical routing data of the airline's actual network, the solutions produced by the ALNS model show substantial improvements in cost efficiency, fleet utilisation, and truck load factors. Modelling the network with a week of shipment requests show that it can reduce trucking costs by up to 19.7%, decrease the number of required trucks by 31.2%, and increase the average truck load factor from 63.1% to 85.8%. Furthermore, the results indicate that increasing the average load factor and decreasing the number of trucks used in the network does not necessarily lead to lower trucking costs, as route length and detours impact total expenses. However, the comparison also highlights several assumptions in the model that differ from real-world operations. These include the absence of dock-door scheduling in the actual network, the models strict adherence to maximum truck volume capacity, and its allowance for shipment splitting. These assumptions should be kept in mind, as they create an idealised scenario that does not fully capture the complexities, deviations, and disturbances present in actual airline operations.

Based on the conclusions drawn and the assumptions made during the development of the model, the final part of this section will present recommendations to improve the models presented in this paper. For the ALNS model, it is crucial to ensure robustness and adaptability for each new instance of similar scale. To achieve this, a more extensive sensitivity analysis is recommended. This will provide a deeper understanding of the sensitivity of the model's hyperparameters, which helps to improve the robustness of the model.

Furthermore, the current MILP model does not account for transitions between summer and winter time. This was not a problem with the data instances used for this study, as they did not encounter changes between summer and winter times. However, implementing these seasonal adjustments will improve the accuracy and reliability of the scheduling and routing solutions.

Additionally, both model inputs do not account for the uncertainty of transit times between different stations in the network and driver rest times. According to European-wide regulations, drivers must take mandatory rest periods after daily driving periods. By introducing variable transit times that account for traffic delays and seasonal variations, and incorporating driver rest times, the model can provide a more precise simulation of route durations, which helps to reflect real-world conditions more accurately.

Moreover, the influence of new network configurations on customer choices has not been taken into account. It is vital to model and analyse how changes in the network affect customer behaviour to ensure that the solutions do not result in a loss of customers opting for the airline to ship their goods.

Finally, the models should be designed to be robust and not overly tuned for specific instances. This will ensure broad applicability and practical utility across various scenarios, making the models more versatile and useful in different operational contexts. By addressing these recommendations, the models can be significantly improved, providing more accurate, reliable, and practical solutions for dock scheduling and shipment routing.

References

Archetti, C., Savelsbergh, M. W., & Speranza, M. G. (2006). Worst-case analysis for split delivery vehicle routing problems. *Transportation science*, *40*(2), 226–234.

Archetti, C., & Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *International transactions in operational research*, *19*(1-2), 3–22.

Bianchessi, N., Irnich, S., & Tilk, C. (2021). A branch-price-and-cut algorithm for the capacitated multiple vehicle traveling purchaser problem with unitary demand. *Discrete Applied Mathematics*, *288*, 152–170.

Chen, P., Golden, B., Wang, X., & Wasil, E. (2017). A novel approach to solve the split delivery vehicle routing problem. *International Transactions in Operational Research*, *24*(1-2), 27–41.

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

Dondo, R., & Cerdá, J. (2014). A monolithic approach to vehicle routing and operations scheduling of a cross-dock system with multiple dock doors. *Computers & Chemical Engineering*, *63*, 184–205.

Dondo, R., & Cerdá, J. (2015). The heterogeneous vehicle routing and truck scheduling problem in a multi-door cross-dock system. *Computers & Chemical Engineering*, *76*, 42–62.

Dror, M., Laporte, G., & Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics*, *50*(3), 239–254.

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

Eroglu, D. Y., Gencosman, B. C., Cavdur, F., & Ozmutlu, H. C. (2014). Introducing the mchf/ovrp/sdmp: Multicapacitated/heterogeneous fleet/open vehicle routing problems with split deliveries and multiproducts. *The Scientific World Journal*, *2014*.

Gendreau, M., Manerba, D., & Mansini, R. (2016). The multi-vehicle traveling purchaser problem with pairwise incompatibility constraints and unitary demands: A branch-and-price approach. *European Journal of Operational Research*, *248*(1), 59–71.

Grangier, P., Gendreau, M., Lehuédé, F., & Rousseau, L.-M. (2021). The vehicle routing problem with cross-docking and resource constraints. *Journal of Heuristics*, *27*, 31–61.

Gromicho, J., Van Hoorn, J., Schutten, J. M., et al. (2012). Vehicle routing with restricted loading capacities.

Gu, W., Cattaruzza, D., Ogier, M., & Semet, F. (2019). Adaptive large neighborhood search for the commodity constrained split delivery vrp. *Computers & Operations Research*, *112*, 104761.

IATA. (2025). Global air cargo demand achieves record growth in 2024 [Accessed: 12-03-2025]. https://www.iata.org/en/pressroom/2025-releases/2025-01-29-02/

Karadgi, S., & Hiremath, P. (2022). Job scheduling on parallel machines with precedence constraints using mathematical formulation and genetic algorithm. *International Conference on Robotics, Control, Automation and Artificial Intelligence*, 835–847.

Khayya, E., Medarhri, I., & Zine, R. (2024). A survey of the vehicle routing problem and its variants: Formulations and solutions. *Mathematical Modeling and Computing*, *11*(1), 333–343.

Khmelev, A., & Kochetov, Y. (2015). A hybrid vnd method for the split delivery vehicle routing problem. *Electronic Notes in Discrete Mathematics*, *47*, 5–12.

Konstantakopoulos, G. D., Gayialis, S. P., & Kechagias, E. P. (2022). Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Operational research*, *22*(3), 2033–2062.

Kurt, A., & Çetinkaya, F. C. (2024). Unrelated parallel machine scheduling under machine availability and eligibility constraints to minimize the makespan of non-resumable jobs. *International Journal of Industrial Engineering and Management*, in–press.

Lei, H., Laporte, G., & Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, *38*(12), 1775–1783.

Lenstra, J. K., & Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, *11*(2), 221–227.

Li, M., Hao, J.-K., & Wu, Q. (2024). A flow based formulation and a reinforcement learning based strategic oscillation for cross-dock door assignment. *European Journal of Operational Research*, *312*(2), 473–492.

Liang, Y., Wang, X., Luo, Z., & Zhang, D. (2023). Integrated optimisation of loading schedules and delivery routes. *International Journal of Production Research*, *61*(16), 5354–5371.

Liao, T. W. (2020). Integrated outbound vehicle routing and scheduling problem at a multi-door cross-dock terminal. *IEEE Transactions on Intelligent Transportation Systems*, *22*(9), 5599–5612.

Liao, T. W. (2021). Integrated inbound vehicle routing and scheduling under a fixed outbound schedule at a multi-door cross-dock terminal. *IEEE Transactions on Intelligent Transportation Systems*, *23*(8), 13217–13229.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., & Hutter, F. (2022). Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, *23*(54), 1–9. http://jmlr.org/papers/v23/21-0888.html

Liu, X., Chen, Y.-L., Por, L. Y., & Ku, C. S. (2023). A systematic literature review of vehicle routing problems with time windows. *Sustainability*, *15*(15), 12004.

Maecker, S., Shen, L., & Mönch, L. (2023). Unrelated parallel machine scheduling with eligibility constraints and delivery times to minimize total weighted tardiness. *Computers & Operations Research*, *149*, 105999.

Manerba, D., & Mansini, R. (2015). A branch-and-cut algorithm for the multi-vehicle traveling purchaser problem with pairwise incompatibility constraints. *Networks*, *65*(2), 139–154.

Palma-Blanco, A., González, E. R., & Paternina-Arboleda, C. D. (2019). A two-pheromone trail ant colony system approach for the heterogeneous vehicle routing problem with time windows, multiple products and product incompatibility. *International Conference on Computational Logistics*, 248–264.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, *34*(8), 2403–2435.

Qu, Y., & Bard, J. F. (2012). A grasp with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, *39*(10), 2439–2456.

Rahbari, A., Nasiri, M. M., Werner, F., Musavi, M., & Jolai, F. (2019). The vehicle routing and scheduling problem with cross-docking for perishable products under uncertainty: Two robust bi-objective models. *Applied Mathematical Modelling*, *70*, 605–625.

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.

Schiewe, P., & Schöbel, A. (2022). Integrated optimization of sequential processes: General analysis and application to public transport. *EURO Journal on Transportation and Logistics*, *11*, 100073.

Schopka, K., & Kopfer, H. (2015). An adaptive large neighborhood search for the reverse open vehicle routing problem with time windows. In *Logistics management: Contributions of the section logistics of the german academic association for business research, 2015, braunschweig, germany* (pp. 243–257). Springer.

Schrage, L. (1981). Formulation and structure of more complex/realistic routing and scheduling problems. *Networks*, *11*(2), 229–232.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, *46*.

Torkzaban, N., Gholami, A., Baras, J. S., & Golden, B. L. (2024). Aasa: A priori adaptive splitting algorithm for the split delivery vehicle routing problem. *Algorithms*, *17*(9), 396.

Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications*. SIAM.

Vincent, F. Y., Jewpanya, P., & Redi, A. P. (2016). Open vehicle routing problem with cross-docking. *Computers & Industrial Engineering*, *94*, 6–17.

Voigt, S. (2024). A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems. *European Journal of Operational Research*.

Wang, Z., Li, Y., & Hu, X. (2015). A heuristic approach and a tabu search for the heterogeneous multi-type fleet vehicle routing problem with time windows and an incompatible loading constraint. *Computers & Industrial Engineering*, *89*, 162–176.

Zhang, H., Ge, H., Yang, J., & Tong, Y. (2022). Review of vehicle routing problems: Models, classification and solving algorithms. *Archives of Computational Methods in Engineering*, 1–27.