

Optimization of Container Transportation for Fixed-Schedule Block Trains with Optional Round Trips in Collaborative Logistics

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Lukas Felician Krasel, BSc

Matrikelnummer 00071426

an der Fakultät für Informatik
der Technischen Universität Wien

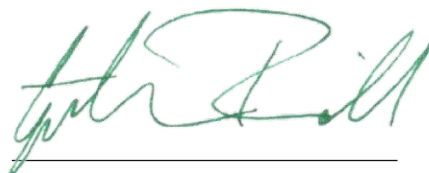
Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl

Mitwirkung: Dipl.-Ing. Georg Brandstätter, BSc

Dipl.-Ing. Ulrike Ritzinger, BSc, PhD

Wien, 2. Dezember 2022

Lukas Felician Krasel



Günther Raidl

Optimization of Container Transportation for Fixed-Schedule Block Trains with Optional Round Trips in Collaborative Logistics

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Lukas Felician Krasel, BSc

Registration Number 00071426

to the Faculty of Informatics
at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl

Assistance: Dipl.-Ing. Georg Brandstätter, BSc
Dipl.-Ing. Ulrike Ritzinger, BSc, PhD

Vienna, 2nd December, 2022

Lukas Felician Krasel

Günther Raidl

Erklärung zur Verfassung der Arbeit

Lukas Felician Krasel, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 2. Dezember 2022

Lukas Felician Krasel

Acknowledgements

First of all, thanks a lot to Georg Brandstätter and Ulrike Ritzinger, my supervisors at the AIT Austrian Institute of Technology, for their great advice, guidance and excellent support during the whole work process. Thanks to Günther Raidl, my supervisor at the TU Wien for the uncomplicated and good support whenever needed. Thanks to my colleagues at the AIT, and especially to Matthias Prandtstetter, the team leader of my unit at the AIT, for creating a respectful and warm environment, including me in their working team, and supporting me with their knowledge.

Thanks to Jovan Zivanovic and Roland Wallner, two of my fellow students and dear friends, who studied alongside me since school and always supported me on the way. Thanks to Sofia for providing first aid to my overcomplicated English sentences and my emotional state. And, of course, thanks to my family and friends for their emotional and personal support.

Without all of you, this work wouldn't have been possible!

Kurzfassung

In Zeiten des Klimawandels und fortschreitender Globalisierung wird der Transportsektor aufgrund steigenden Bedarfs laufen ausgebaut. Gleichzeitig ist er für einen großen Teil des Energie- und CO₂-Verbrauchs in Europäischen Union verantwortlich. Die Nutzung von Güterzügen ist dabei eine der klimafreundlichsten und auch preisgünstigsten Optionen.

Zwei verschiedene Zugverkehrstypen werden derzeit oft eingesetzt: Beim Wagenladungsverkehr setzen sich die Züge aus einzelnen Waggons zusammensetzen, während beim Ganzzugverkehr (auch “Blockzugverkehr”) Züge mit gleichbleibenden Waggons immer direkt von Start zum Ziel ohne Zwischenhalte fahren. Während ersterer mehr Flexibilität bietet, ist letzterer kostengünstiger, denn das Zusammensetzen der Waggons zu Zügen an sogenannten Rangierbahnhöfen ist aufwändig. In dieser Arbeit untersuchen wir einen alternativen Ansatz, der die Vorteile beider Zugverkehrstypen kombiniert. Wir optimieren den Transport von 20’ Standardcontainern und betrachten Ganzzüge, die ähnlich wie Passagierzüge fahren: Sie haben fixe Fahrplänen, wiederholte Fahrten — sogenannten “Round Trips” — und mehrfache Stops. Die Idee ist, dass Leerfahrten oder schlecht ausgelastete Zugfahrten vermieden werden, indem wir die geladenen Container auf andere Züge (oder Fahrten) verlagern und dadurch die gesamte Zugfahrt weglassen und einsparen können.

Das dazugehörige Problem heißt “Multiple Collaborative Round Trip Problem” (MCRP), stammt aus dem Bereich der kollaborativen Logistik und wird im Rahmen des Forschungsprojekts PhysICAL untersucht. Containerpfade und der Einsatz der Züge werden aus Sicht einer Spedition optimiert. Durch die globale Perspektive sind potentiell Einsparungen möglich, wodurch die Spedition Kosten spart. Diese kann dann wiederum bessere Preise für Kund:innen anbieten.

Wir stellen eine formale Problembeschreibung auf und repräsentieren das Problem mit Hilfe eines Graphen, der die zeitliche Dimension über seine Struktur darstellt (“Time-expanded Network”). Um das Problem exakt zu lösen, modellieren wir ein ILP (“Integer Linear Program”). Außerdem beweisen wir die Komplexität des MCRP in seiner Entscheidungsvariante.

Wir generieren künstliche Instanzen, um die Performance unseres Ansatzes und die Eigenschaften der Lösungen zu den Instanzen zu untersuchen. Die Instanzen basieren auf einem echtem Schienennetzwerk, das zehn relevante Städte des europäischen Güterverkehrs verbindet. Wir nehmen eine Speichenarchitektur an und lösen ein ILP, um die verbindenden Längen der Gleise zu minimieren. Außerdem verwenden wir für die Parame-

ter unserer Instanzen fundierte Werte, die auf Literatur und unseren Projektpartner:innen beruhen.

Wir untersuchen unseren Algorithmus auf Laufzeit und andere Qualitätskriterien. Des Weiteren untersuchen wir, welche Parameter die Komplexität des Problems beeinflussen und welche Auswirkungen sie auf die Lösungsqualität haben. Im speziellen betrachten wir eine einzelne beispielhafte Lösung, um einen tieferen Einblick in die Lösungsstrukturen unserer Instanzen zu erhalten. Zuletzt vergleichen wir die künstlichen Instanzen mit der derzeitigen Praxis und untersuchen die Anwendbarkeit unseres Problems in der Praxis.

Abstract

In times of globalization and climate change, the freight transportation sector faces increasing demand. At the same time, it is responsible for a major part of the energy consumption and CO₂ emissions in the European Union. Rail freight transport appears to be one of the best choices when considering economical and ecological aspects.

In current practice, two types of rail freight are common: In so-called *wagonload freight*, individual trains are composed of individual wagons, while *block trains* have a fixed set of wagons and directly connect their source to their destination without intermediate stops. Wagon load trains allow more flexibility, but the uncoupling and rearranging operations of wagons are rather costly. In this study, we investigate an alternative approach to combine the advantages of both operation techniques and potentially increase the efficiency of the freight sector.

We optimize the shipment of individual 20' containers, which are carried by block trains. These block trains operate on fixed schedules with potentially recurring round trips and multiple stops, similar to passenger trains. The idea is to avoid empty runs and underutilized round trips by shifting the respective containers to other trains or round trips. These empty round trips then do not have to take place, hence costs are saved.

The corresponding problem is called “Multiple Collaborative Round Trip Problem” (MCRP). It arises in the context of the research project PhysICAL from the point of view of a freight forwarder in collaborative logistics. If the container paths are optimized from a global point of view, the operational cost of the freight forwarder can be reduced. In return, the operator can then offer better prices to its customers.

We formally define the MCRP and represent it with the use of a time-expanded network, state an Integer Linear Program (ILP) to solve it in an exact way, and prove the NP-completeness of its decision variant.

To investigate and benchmark the problem, we generate a set of artificial instances. They are based on a real-world rail network and connect a subset of ten relevant cities for current rail freight transport in Europe. The individual instance parameters and values are researched or provided by our project partners.

We analyze the performance of our algorithm, the hardness of the problem, and the characteristics of the solutions for a variety of instance parameters. We also investigate a single solution in detail to provide deeper insights into the composition of the solutions. At last, we compare our method to the state of the art and draw managerial insight from our results.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
Table of Abbreviations	xv
Table of Mathematical Symbols	xvi
1 Introduction	1
1.1 The Physical Internet (PI) and the PhysICAL Project	1
1.2 Problem Description	2
1.3 Research Questions and Scientific Contributions	3
1.4 Thesis Outline	3
2 Fundamentals and Related Work	5
2.1 Prior Knowledge and Notation – Mathematical and Computer Scientific Background	5
2.2 Literature Review — Related Work	9
3 The Multiple Collaborative Round Trip Problem	13
3.1 Problem Definition	13
3.2 Problem Representation — Time-Expanded Network	17
3.3 ILP Problem Formulation	23
3.4 Complexity	28
4 Instance Generation and Experimental Setup	33
4.1 Rail-Network	34
4.2 Instance Parameters	36
4.3 Experimental Setup	44
5 Results	45
5.1 Performance and Solution Characteristics	45
5.2 Investigating an Exemplary Solution	50
5.3 Investigation of Results	57
	xiii

5.4 Summary of Results and Practical Insight	82
6 Conclusion	89
6.1 Future Work	90
Appendix	93
Bibliography	99

Abbreviations

ILP	Integer Linear Program
LP	Linear Program
MCRP	Multiple Collaborative Round Trip Problem
MILP	Mixed Integer Linear Program
MST	Minimum Spanning Tree
NP	Nondeterministic polynomial (problems), see Section 2.1.3
P	Polynomial (problems), see Section 2.1.3
PhysICAL	Physical Internet through Cooperative Austrian Logistics
PI	Physical Internet
SAT	Satisfiability (Problem), see Section 3.4

Mathematical Symbols

Below, we list all mathematical symbols used in this work. We only include symbols that appear in more than one place in this work, as otherwise the symbol's definition is given at their occurrence anyhow.

Hints regarding notation We usually use the hat-symbol $\hat{\cdot}$ or the visually similar single-dot notation $\dot{\cdot}$ to indicate that a symbol is related to *terminals* (e.g., \hat{V} , \hat{A} , \hat{C}_a , or \dot{C}_j^t).

In contrast, we often use the bar-symbol $\bar{\cdot}$ or the double-dot notation $\ddot{\cdot}$ to emphasize a relation to *trains* (e.g., \bar{V} , \bar{A} , \bar{C}_a , \bar{W}_a , or \ddot{C}_l^t , \ddot{W}_l^t).

The tilde-symbol $\tilde{\cdot}$ is used to indicate a relation to lifting and halting of trains at terminals (e.g., \tilde{A} , \tilde{L}).

A	Arcs of a graph (often being our time-expanded network, introduced in Section 3.2.3)
A_p^{ld}	Arrival time of the p -th stop π_p^{ld} of train l 's d -th round trip, see Section 3.1
$\bar{A}(l)$	Transport arcs of train l where it is not halting and which are part of an active round trip, used for calculating the transport and weight capacity utilization, see Section 5.1
\check{A}	Source and sink arcs, see Section 3.2.3
$\check{A}(i)$	Source and sink arcs of container i , see Section 3.3.1
\hat{A}	Storage arcs, see Section 3.2.3
$\hat{A}(i)$	Storage arcs of container i , see Section 3.3.1
$\hat{A}(j)$	Storage arcs of terminal j , see Section 3.2.3
\bar{A}	Transport arcs, see Section 3.2.3
$\bar{A}(i)$	Transport arcs of container i , see Section 3.3.1
$\bar{A}(l)$	Transport arcs of train l , see Section 3.2.3
$\bar{A}_R(l, d)$	Round trip d 's transport arcs of train l , see Section 3.3.1
\tilde{A}	Lifting arcs, see Section 3.2.3
$\tilde{A}(i)$	Lifting arcs of container i , see Section 3.3.1
$\tilde{A}(l)$	Lifting arcs of train l , see Section 3.2.3
$\hat{A}(j, t)$	Terminal lifting arcs of action time point t and terminal j , see Section 3.3.1
$\hat{A}(i)$	Placeholder arcs of container i , see Section 3.3.1
B_p^{ld}	Departure time of the p -th stop π_p^{ld} of train l 's d -th round trip, see Section 3.1
buff_i	Number of days of the buffer time windows of container i , see Section 4.2.3
C	Set of <i>city</i> terminals, see Section 4.1
\dot{C}_j^t	Storage capacity of terminal j at time point t , introduced in Section 3.1
\hat{C}_a	Storage capacity of terminal j along arc $a \in \hat{A}(j)$, see Section 3.2.3
\ddot{C}_l^t	Transport capacity of train l at time point t , introduced in Section 3.1
\bar{C}_a	Transport capacity of train l along arc $a \in \bar{A}(l)$, see Section 3.2.3
$c(a, i)$	Costs along arc a for storing, lifting, or transporting container i , see Section 3.2.3

c_j^L	Lifting costs of terminal j , introduced in Section 3.1
c_j^S	Storage costs of terminal j , introduced in Section 3.1
\tilde{c}_l^t	Transport cost of train l at time point t , introduced in Section 3.1
\bar{c}_d^l	Round trip costs of train l for making round trip R_d^l , introduced in Section 3.1
D_i	Deadline of container i , introduced in Section 3.1
D^l	Index set $D^l = \{1, \dots, d_{\max}^l\}$ of the round trips of train l , see Section 3.1
d	Sink vertex, <i>destination</i> of our commodity flow, see Section 3.2.2
d_i	Destination terminal of container i , introduced in Section 3.1
d_i^*	Destination vertex $d_i^* = d_i^{t_D^D}$ of container i , see Section 3.2.2
d_{\max}^l	Total number of round trips $d_{\max}^l = R^l $ of some train l , see Section 3.1
$d(e)$	Distance of edge e , see Section 4.1.2
E_i	Expected shipping time of a container, see Section 4.2.3
h	Length of the considered time horizon in days, see Section 4.2
I	Set of containers, see Section 3.1
J	Set of terminals, see Section 3.1
j_p^{ld}	Halting terminal of the p -th stop π_p^{ld} of train l 's d -th round trip, see Section 3.1
L	Set of trains ("lines"), see Section 3.1
N	Time-expanded network $N = (V, A)$, introduced in Section 3.2
NP	Class of problems whose solutions are verifiable in polynomial time, see Section 2.1.3
$\mathcal{O}(\cdot)$	Landau symbol for complexity, see Section 2.1.3
o	Source vertex, <i>origin</i> of our commodity flow, see Section 3.2.2
o_i	Origin terminal of container i , introduced in Section 3.1
o_i^*	Origin vertex $o_i^* = o_i^{t_R^R}$ of container i , see Section 3.2.2
P	Class of problems which are solvable in polynomial time, see Section 2.1.3
P	Set of <i>port</i> terminals, see Section 4.1
p_{\max}	Total number of stops $p_{\max} = R_d^l $ of a round trip R_d^l , see Section 3.1
q_j^t	Lifting capacity of terminal j at time point t , introduced in Section 3.1
R_i	Release time of container i , introduced in Section 3.1
R^l	Set of (potential) round trips of train l , see Section 3.1
r_i	Revenue of container i , introduced in Section 3.1
$\text{succ}_j(t)$	Successor function providing the subsequent time point of t w.r.t. terminal action time points $\hat{T}(j)$, see Section 3.2.1
$\text{succ}_l(t)$	Successor function providing the subsequent time point of t w.r.t. train action time points $\bar{T}(l)$, see Section 3.2.1
T	Set of time points of the considered time horizon, introduced in Section 3.1
T'	Set of time points of our time horizon, excluding the last time point: $T' = T \setminus \{t_{\max}\}$
$\hat{T}(j)$	Terminal action time points of terminal j , see Section 3.2.1
$\hat{T}'(j)$	Reduced terminal action time points of terminal j , excluding the last action time point, see Section 3.2.1

$\bar{T}(l)$	Train action time points of train l , see Section 3.2.1
$\bar{T}'(l)$	Reduced train action time points of train l , excluding the last action time point, see Section 3.2.1
t_{\max}	The last (greatest) time point of our considered time horizon T , introduced in Section 3.1
t_i^D	Destination (“deadline”) time point of container i , see Section 3.2.2
t_i^R	Origin (“release”) time point of container i , see Section 3.2.2
V	Vertices of a graph (often being our time-expanded network, introduced in Section 3.2.2)
\check{V}	Source and sink vertices $\check{V} = \{o, d\}$, introduced in Section 3.2.2
\hat{V}	Terminal vertices, see Section 3.2.2
\bar{V}	Train vertices, see Section 3.2.2
\bar{W}_l^t	Weight capacity of train l at time point t , introduced in Section 3.1
\bar{W}_a	Weight capacity of train l along arc $a \in \bar{A}(l)$, see Section 3.2.3
w_i	Weight of container i , introduced in Section 3.1
x_a^i	Binary ILP variable indicating if container i is shipped along arc a , introduced in Section 3.3.2
y_i	Binary ILP variable indicating if container i is shipped at all, introduced in Section 3.3.2
z_d^l	Binary ILP variable indicating if round trip d of train l is active, introduced in Section 3.3.2
Δ_a	Duration of transport arc $a = (l^1, l^{t_2})$, introduced in Section 5.1
η^l	Home terminal of train l , introduced in Section 3.1
Π^l	Plan of train l containing all its stops π_p^{ld} , see Section 3.1
π_p^{ld}	The p -th stop $\pi_p^{ld} = (j_p^{ld}, A_p^{ld}, B_p^{ld})$ of the d -th round trip of train l , see Section 3.1
τ	The maximum allowed length of a detour w.r.t. its original distance (in percent), see Section 4.1.2



Introduction

Every day, thousands of containers are shipped around the globe by various modes of transportation, most often carried by ships, trains, trucks and planes. In the European Union, freight transportation has increased by 7.8 percent within the years 2010 to 2017 to a total of 3.731 billion tonne-kilometers [63, p. 8]. According to the European Court of Auditors [23, p. 11], freight transportation is responsible for about a third of the energy consumption and CO₂ emissions in the European Union.

In times of climate change and ongoing globalization, the ecological and economical aspects of transportation demand both cost efficiency and climate friendly ways of transportation. Out of all the means of motorized transportation, the usage of trains is the most environmentally friendly choice, as, e.g., stated by [2]. For this reason, the European Union aims to shift the transportation of goods from road to rail (see, e.g., European Court of Auditors [23, p. 11f]). Nevertheless, 76 percent of all goods in the European Union are transported by road (see Eurostat [24, p. 57]), and between 2010 and 2017 the road transport sector in the European Union increased faster (9.2 percent) than the rail sector (7.0 percent) (see [63, p. 8]). One of the reasons are capacity limitations in the rail sector, originating from shared infrastructure between passenger and freight trains. Therefore, the improvement of efficiency in the rail freight transport sector plays an important role in reaching climate targets and increasing the overall throughput of the transport sector.

1.1 The Physical Internet (PI) and the PhysICAL Project

The research project PhysICAL (Physical Internet through Cooperative Austrian Logistics), see [29], has been started in 2020 and aims at a more efficient and ecologically sustainable transport sector in Austria through cooperative logistics and the implementation and development of the *Physical Internet* (PI).

The idea of the PI was first introduced by Ballot et al. [6] as follows: “Like the Digital Internet that conveys data, the concept is to connect and synchronize all logistics

networks to create a collaborative and robust physical network of networks, capable of continually optimizing the shipment of ‘encapsulated’ goods of many types and sizes [...] by optimizing both the operator’s and the customer’s economic models [...]” [6] An integral part of this process is the implementation of a digital twin, the digital representation of the physical objects such as trains, containers, etc., to be able to then apply, e.g., optimization algorithms to logistic transport problems that may arise.

Four pilot projects in different sectors, developed by 17 project partners, are part of the PhysICAL project and aim at demonstrating the economical, environmental and socioeconomic advantages of cooperative logistic. The contribution of this work is part of the pilot project “Pilot 2.0”. As stated on the PhysICAL project web page [29], it focuses on the development of an open transport management platform. The goal is to replace unimodal (typically road-bound) services of transportation that are most common and often lead to inefficiency and empty runs. Instead, more intermodal modes of transportation shall be offered by the platform through digitalization of the transport chain, leading to an easier order placement due to better presentation of the offers, better transparency in the business of transportation without the disclosure of data of transport companies, and to an efficiency increase of the whole physical transportation process.

1.2 Problem Description

In context of the Pilot 2.0 project of PhysICAL, we consider a new problem that arises when optimizing the scheduling of trains and containers from the point of view of the transport management platform: The “Multiple Collaborative Round Trip Problem” (MCRP) is an optimization problem that was first introduced by Prandtstetter et al. [54]. The transport management platform functions as a freight forwarder (and potentially carrier). It handles shipping requests of containers booked by customers (shippers) and is responsible for the transportation of containers using trains.

Informally, the problem can be described as follows: We are considering a number of rail terminals, connected by different train lines, usually operated between cities (and ports). Containers are located at these terminals and have to be shipped to another one within a given schedule. Each train is operated on fixed routes, along which it travels in so called (potential) *round trips*, with a fixed schedule. The goal is to plan a cost-optimal transportation schedule for all containers w.r.t. the given trains and their routes. Furthermore, we have the option to cancel individual round trips if they do not yield enough reward.

The underlying idea of the problem is to save costs for both the transport manager though better utilization of the trains, and the shippers due to being offered reduced prices caused by the collaborative logistics approach. Currently, the typical scenario is that a freight forwarder orders trains, while shippers order individual slots of containers on these trains. If the overall costs of each train are reduced, the shippers can be charged less. This can be achieved by making use of a fourth party logistic provider (the transport management platform), leading to increased utilization of the managed trains as the

transport management platform tries to omit unnecessary train trips in order to save costs and avoid wear on trains.

Today's freight trains are usually operated as wagonload or block trains. Wagonload trains are composed of individual wagons, while block trains (unit trains) have a fixed set of wagons. Wagonload freight offers more flexibility in the scheduling process, but the arrangement of wagons in shunting yards is rather costly.

Our alternative approach (the MCRP) combines the advantages of both types. In contrast to the common practice of operating trains directly between two cities without intermediate stops, we make use of block trains that operate on round trips with more than two stops and follow a predefined schedule, similar to the operation of passenger trains.

1.3 Research Questions and Scientific Contributions

In this section, we specify our research questions and scientific contributions.

- The MCRP was introduced by Prandtstetter et al. [54], but only an informal problem description was given. Therefore, we ask:

How can we formally define the problem s.t. it suits its context in the PhysICAL project?

- How can we solve the problem in an exact way?
- As we investigate a problem that is new and an alternative approach of rail freight transport, it is unclear how real-world instances would look like. Furthermore, to the best of our knowledge, neither are there similar instances available that could be easily adapted to our problem, nor does there exist a collection of instance parameters or related values in the context of rail freight transport. Therefore, we gather relevant information and investigate the following questions:

What are suitable instances to benchmark our problem? How are these instances structured, and what are appropriate values for our instance parameters?

- How does our algorithm perform in practice, and what instance parameters influence the hardness of our instances?
- What do our solutions look like, and what managerial insight can we draw from our benchmark?

1.4 Thesis Outline

We present the prerequisite knowledge that is necessary to understand this work in Chapter 2, followed by a review of related work and literature.

Chapter 3 contains a detailed description of the MCRP, including a formal definition of the problem, and a suitable problem representation using a time-expanded network.

We further propose an ILP model for solving the problem, and follow with the proof of its NP-completeness (considering its decision variant).

In Chapter 4 we describe the generation process for our instances, their parameters, and our experimental setup for benchmarking.

Chapter 5 contains the results of our study. We introduce solution characteristics to be able to investigate the large number of benchmarked solutions. Next, we present a single exemplary solution for detailed understanding of the structure of the solution and the calculation of the solution characteristics. Then, we analyze the effects of different instances parameters on these solution characteristics, and finally draw some managerial insights.

Finally, Chapter 6 contains a summary and conclusion of our study, including an outlook on possible future research.

CHAPTER 2

Fundamentals and Related Work

In this section, we briefly discuss prior knowledge that is relevant for the understanding of this work, and present a literature review of related work and similar problems.

2.1 Prior Knowledge and Notation – Mathematical and Computer Scientific Background

This section contains relevant mathematical notations and concepts, followed by background information on complexity and (Mixed-)Integer Linear Programming.

2.1.1 Interval Notation

We use interval notation for discrete and continuous sets, where square brackets include, while round brackets exclude the specified limit. For clarification, considering sets of natural numbers, we have, e.g., $[1, 3] = \{1, 2, 3\}$, $(1, 3) = \{2\}$, and $[1, 3) = \{1, 2\}$.

2.1.2 Graphs

In this section, we present relevant background information associated with graphs. In general, a graph $G = (V, E)$ consists of *vertices* $v \in V$ (also called “nodes”), and undirected *edges* $\{v_1, v_2\} \in E$ (or directed edges, so-called *arcs* $(v_1, v_2) \in E$), connecting pairs of vertices. For further properties regarding graphs, see, e.g., [27, p. 3].

Minimum Spanning Tree

This problem asks the following: Given a connected graph $G = (V, E)$, and non-negative weights $w(e)$ for edges $e \in E$, we want to find a Minimum Spanning Tree (MST), i.e., a cycle-free simple graph $T = (V, E')$ spanning all vertices by edges $E' \subseteq E$ with minimum

weights. Two traditional algorithms are widely used, both starting by sorting the edges by their weight, and then incrementally considering and adding edges:

- Kruskal's Algorithm [45], introduced in 1956, starts with an (incomplete) tree $T = (V, E' = \emptyset)$, then considers edges in-order and adds them to E' when they don't introduce a cycle. This is repeated until no further edge can be added.
- Prim's Algorithm [55], introduced in 1957, starts with any vertex $v \in V$, and adds it to T . We now consider incident edges $\{u, v\}$ (in-order) w.r.t. the already included vertices in our tree, and add the edge (including the connected vertex) if it does not introduce a cycle, i.e., if u or v are not yet part of our tree. We repeat this until all vertices are part of our tree.

Both algorithms yield optimal results and have slightly different runtime properties w.r.t. the density of the given input graph G , but for our study, both algorithms can be applied when requiring a MST.

Shortest Path – Dijkstra

Given a graph $G = (V, E)$, edge weights $w(e) > 0$ for all edges $e \in E$, and two specific vertices $s, t \in V$, we want to find the shortest path from s to t w.r.t. G . Dijkstra's algorithm [21], proposed in 1959, solves this greedily and works as follows:

For all nodes, we initialize the following variables: We set all nodes to *unvisited*, we set the *predecessor* of each node $v \in V$ to $p(v) := \emptyset$, and the (accumulated) *distance* $d(v)$, i.e., the distance to reach node v from s , to $d(v) := \infty$, except for node s , where $d(s) := 0$. While t is *unvisited*, we choose the *unvisited* node v with minimum *distance* $d(v)$ and perform the following steps: (1) We set the node v to *visited*, (2) and for all *unvisited* neighbors n of v , we calculate the *distance* $d'(n)$ from s to n via v , i.e., $d'(n) := d(v) + w(\{v, n\})$, and if $d'(n) < d(n)$, we update the *distance* $d(n) := d'(n)$, and set the *predecessor* of n to v , i.e., $p(n) := v$.

When we are done, we get the shortest path from s to t with length $d(t)$ by following the chain of *predecessors* of t ($t, p(t), p(p(t)), \dots, s$), and so on, until we reach $p(v) = s$.

Note that this algorithm only works for positive edge weights. Other algorithms for solving this problem are, e.g., the Bellman-Ford-Algorithm [7, 27, 52] (for negative weights), or the Floyd-Warshall-Algorithm [25, p. 345] (for negative weights and all node pairs).

Time-Expanded Networks

When considering some problems with temporal dimensions, time-expanded networks, sometimes also called "time-space networks", are often used for modeling. A time-expanded network $N = (V, A)$, where V are the nodes and A are the arcs (or edges), can be viewed as having multiple copies of an underlying network for each point in time that we consider, see, e.g., the book by Ahuja et al. [1, p. 737].

2.1.3 Complexity

In this section, we sum up background information on computational complexity and reductions. Most of the following content can be found in, e.g., [18].

When considering problems from a theoretical perspective, we are usually interested in the hardness of a problem. This can be classified by the running time required for obtaining the solution in the worst case, or the required memory for storing or checking a solution in the worst case, w.r.t. the size of the input n .

Landau Symbol – Big-O Notation

Originally introduced by Bachmann [5] in 1894, and adapted by Landau [47] in 1909, the Landau symbol $\mathcal{O}(\cdot)$ is used for describing the order, upper bound, or growth rate of a function, following [18, p. 47]: Let $f(n), g(n)$ be two functions $f, g : \mathbb{N} \mapsto \mathbb{R}$, then $f \in \mathcal{O}(g)$ means that f does not grow *significantly* faster than g , formally:

$$f(n) \in \mathcal{O}(g(n)) \iff \exists C > 0 \exists n_0 > 0 \forall n > n_0 : |f(n)| \leq C \cdot |g(n)|$$

“Easy” Problems: P

We often refer to the class of problems that have a polynomial runtime $\mathcal{O}(n^c)$, for some constant c w.r.t. the size of the input n , as *easy* or *efficient*, forming the class of problems P. An important property of these problems is that solving the problem and checking a solution both takes polynomial time. Examples for such problems are sorting of lists or finding MSTs. Note that in practice these problems can still be hard to solve for large instances.

“Harder” Problems: NP and Beyond

For problems in the class NP, there exists a polynomially checkable certificate. This means that given a solution, we can check its correctness in polynomial time. Nevertheless, it appears that there is no polynomial-time algorithm for solving these problems and efficiently obtaining a solution (unless $P=NP$) (see, e.g., [28, 31]). Therefore, verifying a solution of an NP problem is “easy” (as checking is in P), but finding a solution is “hard”, i.e., taking into account the even stronger Exponential Time Hypothesis, see [38], many of these problems can only be solved in exponential time ($\mathcal{O}(c^n)$ for some constant c), see, e.g., [19]. Note that $\mathcal{O}(n^c) \subset \mathcal{O}(c^n)$ for some constant c , e.g., $\mathcal{O}(n^{100}) \subset \mathcal{O}(2^n)$.

Furthermore, there are much harder problems lying outside of NP, e.g., chess, (when generalizing to a reasonable $n \times n$ -board and ignoring the threefold repetition and the 50-move rules, see, e.g., [60]). Intuitively, when playing chess we cannot decide if a played move is best after playing it. Instead, we know this only after checking all other possibilities and their consequences as well, expanding the whole search tree.

Reductions

For showing the hardness of a problem, we use the concept of reductions: Assume we have a new problem A , and we already know the hardness of another problem B . We can use the concept of *reduction* to show the hardness of problem A . The idea is to *reduce* A to B (sometimes denoted as $A \leq B$), and solve problem A by solving the known problem B . This is done by transforming the input of problem A in polynomial time to an input for problem B , solve B with our new input, and then use the output of B to transform it into a solution for A (again in polynomial time). We now know that A is *not harder* than B , as we could solve A by using B .

If we show a reduction in the other direction (as we will later do for our problem under consideration), i.e., we solve B by using our problem A , we can conclude that A is at least as hard as B .

NP-Hard and NP-Complete Problems

The notion of a problem B being NP-hard means that it is *at least as hard as the hardest problems in the class NP*. This means that every problem A in NP can be reduced to B (formally $A \in \text{NP}$, B is NP-hard $\implies A \leq B$), meaning that we can solve all NP-problems by using an algorithm for problem B .

Note that the *membership* of NP only tells us that the problem has a polynomial certificate, but it could still be member of P. Knowing that problem B is NP-hard ensures that it is not a member of P (unless $P=NP$).

Nevertheless, knowing that our problem B is NP-hard does not tell us if it is indeed in the class of NP. Instead, it could also be a member of a superclass of NP, hence be much harder. Therefore, we also have the notion of NP-completeness, which means that a problem it is both a member of NP (not harder than NP) and (at least) NP-hard, hence among the hardest problems in NP.

After Cook showed the NP-completeness of the SAT-problem (see Section 3.4) in 1971, Karp [43] used reductions in 1972 to show that 21 combinatorial and graph theoretical problems can be reduced to each other, hence they are all NP-complete.

2.1.4 (Mixed-)Integer Linear Programming

One of these NP-complete problems is the 0-1 Integer Programming Problem (also called “Binary Integer Program”), formulated as a decision problem. The optimization version of this problem, allowing non-binary variables, is called Integer Linear Programming (ILP), and extending this to non-integer values results in a Mixed-Integer Linear Program (MILP). Following the book “Integer Programming” by Wolsey [65], an MILP is defined

as follows:

$$\max c_1 x_1 + c_2 x_2 \quad (2.1)$$

$$A_1 x_1 + A_2 x_2 \leq b \quad (2.2)$$

$$x_1 \geq 0 \quad (2.3)$$

$$x_2 \geq 0 \text{ and integer} \quad (2.4)$$

In the above, c_1 (c_2) is an n -dimensional (p -dimensional) row vector, x_1 (x_2) is an n -dimensional (p -dimensional) column vector, A_1 (A_2) is an $m \times n$ ($m \times p$) matrix, and b is an m -dimensional column vector. The idea is to describe the set of feasible solutions for a given problem as a set of variables and linear inequalities.

MILPs are often used for modeling NP-hard problems, as they provide a mathematical representation for the problem and many well-performing solvers, e.g., CPLEX [36], are available.

The idea of solving MILPs, originally introduced by Land and Doig [46], is to relax the integrality constraints of the integer variables, resulting in a Linear Program (LP), and then perform a branch-and-bound algorithm on the values of variables with violated integrality constraints. In every node of the branch-and-bound tree, a LP is solved. If this node cannot be pruned (due to its bound, infeasibility, or optimality), and there exists a violated integrality constraint for some variable $x_i = v$ with value $v \notin \mathbb{Z}$, we *branch* and create two subproblems with constraints $x_i \leq \lfloor v \rfloor$ and $x_i \geq \lceil v \rceil$ respectively.

For solving LPs, Dantzig's simplex method, see, e.g., [20], performs well in practice, although yielding an exponential runtime in the worst case. Nevertheless, there do exist polynomial-time algorithms, such as the interior-point method (see [42]) or algorithms based on the ellipsoid method (see [44]), but in practice these are often outperformed by the simplex method.

2.2 Literature Review — Related Work

Although there is ongoing research in the sector of railway transportation and the issue of improving efficiency, to our knowledge, just a few studies tackle similar problems to the one considered here, and all problems differ significantly. Archetti and Peirano [4, p. 1] state that “Despite [freight forwarding companies] being one of the most relevant figures in international multimodal transportation, freight forwarding companies optimization problems did not receive much attention from the research community.” As the transport management platform has the role of a freight forwarder company, our problem falls into this category.

2.2.1 Similar Problems

One of the first problems to be considered in the context of transportation of goods is the *transportation problem*, first formalized by Hitchcock [35] in 1941, and approached by, e.g., Ford and Fulkerson [26]. It considers the optimal (direct) shipment of goods from several

source nodes with specified supplies to a number of sink nodes with specified demands, respecting transportation costs and capacities. It was extended to the *transshipment problem* by Orden [53] in 1956, also considering intermediate transshipment centers.

Both these problems can be formulated as the more general *minimum-cost multi-commodity flow problem*, originally introduced by Tomlin [61]. In this problem, multiple commodities are sent out and received by various nodes in a graph, respecting the expected demand of source and sink nodes and edge capacities, while shipping costs along edges are minimized. In contrast to the transportation problem, a single node can be both source and sink node, and we consider multiple commodities. In some sense, our problem is a more general version of the minimum-cost multi-commodity flow problem with unsplittable flows on a specialized time-expanded network. Additional constraints have to be added to encode potential round trips. Nevertheless, the construction of the time-expanded graph is non-trivial by itself, as we have to respect (1) train schedules and container release times and deadlines, and (2) capacity constraints that are not defined along single arcs, but can include multiple arcs (see lifting constraints in Section 3.3.3). The minimum-cost multi-commodity flow problem and other related flow problems can be found in, e.g., “Network Flows” by Ahuja et al. [1].

Guastaroba et al. [34] present a literature overview on freight forwarder transportation problems with a focus on intermediate facilities (e.g., warehouses, transshipment centers — or terminals in our case), which separate different stages of the supply chain. They identify three major branches of study: vehicle routing problems, transshipment problems and service network design problems [4, p. 2]. Our problem under consideration falls into the category of “many-to-many transshipment problems”, so in the following we list the most closely related works in this survey:

Chen et al. [13] consider the *cross-docking problem* with the goal to “find a minimum cost distribution plan involving cross-docks based on anticipated supplies and demands” [13, p. 45]. Lim et al. [49] extends the classical transshipment problem by introducing characteristics arising from cross-dock networks such as hard time windows, inventory holding capacities and costs of the transshipment centers, and a set of fixed or flexible shipping schedules of transportation, also including further capacity constraints and costs (see [34, p. 777]). Their problem formulation as a minimum-cost flow problem using a time-expanded network seems to be quite closely related to ours, but uses splittable units, not individual containers, lacks the concept of round trips, and does not capture our co-dependent lifting constraints (see Section 3.3.3). Another study by Chang [12] examines an intermodal operational issue: “How an international carrier selects best routes for shipment through the international intermodal network.” [12, p. 2877], and they formulate it as a multi-objective multimodal multi-commodity flow problem with time windows and concave costs.

Archetti and Peirano [4] tackle the *air transportation freight forwarder service problem* from the point of view of a freight forwarder. They consider different transportation services to satisfy a set of shipments and minimizing the total cost. A considered route consists of truck transportation from factories to warehouses or directly to an airport, air transportation to a destination airport, and the final delivery in the destination country,

typically handled by foreign agents.

Despite many similarities of already mentioned related work, all of these problems do not consider railway transportation systems and are not suited for modeling our setting-specific problem. There exist various other distantly related problems in the research field of intermodal transportation, but we will focus now on railway transport related problem only.

2.2.2 Railway Transportation Problems

Many studies have been conducted on problems like the *blocking problem*, the *block-to-train assignment problem*, or the *shipment-to-block-assignment* of wagonload trains, e.g., [66], [41], [30], [67], or [50]. These problems consider how to block wagons of wagonload trains together to minimize (un-)coupling operations and route blocks of trains efficiently thorough the network. Xiao and Lin [66] mention the three typical phases for forming a train formation plan: “(1) determining the assignments of shipments to blocks, (2) specifying the train routing and scheduling, (3) determining the assignments of blocks to trains.” [66, p. 220]. In this context, further problems such as the optimization of operations at shunting yards arise, e.g., considered by [11].

Nevertheless, these problems are not directly relevant for our work due to the fact that we do not consider wagonload trains, but block trains that carry containers (see Section 3.1.2).

More closely related to our work are problems arising in rail transshipment yards, considering scheduling of trains within the yard for optimizing reloading of containers, see, e.g., [14] or [32]. Cicheński et al. [14] divide the process into five steps: “(1) Schedule time slots to service the trains. (2) Assign a railway track to each train. (3) Decide on the outbound positions of the containers on trains. (4) Assign container moves to gantry cranes. (5) Create a schedule for cranes to move all containers to their target outbound positions.” [14, p. 2]. Although in our problem, we do consider reloading of containers at terminals of our considered set of trains, we do not directly optimize any operations required in the individual yards for lifting, as this is the task of the yard operator and not the freight forwarder. Instead, we just assume a fixed lifting capacity for each terminal.

In the context of collaborative logistics, Moll et al. [51] investigate the potential of collaborations of freight railways and shippers, and review the problems and current best practice based on interviews with the Swiss freight railway and their customers, “complemented with quantitative analyses on short-term shipment forecasts and operated trains, supporting the interpretation and validation of interview results.” [51, p. If] They also describe the process of how to order weekly block trains from the point of view of a freight railway operator, but in contrast to our problem, block trains are ordered as a whole without considering individual containers [51, p. 48ff].

To sum up the above review, it appears that all closely related problems differ from our problem for two reasons: (1) Our problem does not exist in practice (yet), hence was not investigated so far, and (2) most investigated problems are either too general or too specialized for problems occurring in practice. Thus, they in- or exclude important

2. FUNDAMENTALS AND RELATED WORK

considered requirements, e.g., they lack concepts to encode potential round trips or train schedules, container release times and deadlines are seldom respected, our commodities (being single containers) cannot be split, and we have co-dependent constraints (w.r.t. lifting) on arcs.

The Multiple Collaborative Round Trip Problem

In this chapter, we define our problem formally and clarify its context. We represent it by using a time-expanded network, formulate it as an ILP model, and prove the NP-hardness of its decision variant.

3.1 Problem Definition

Our considered optimization problem, the Multiple Collaborative Round Trip Problem (MCRP), is formally described in this section. Recall the informal introduction from Section 1.2. The MCRP is defined as follows:

We have a discretized time period, represented by the set $T = \{1, 2, \dots, t_{\max}\}$, where every pair of consecutive *time points* $t, t+1 \in T$ is a fixed *time unit* μ apart. We further define the set T' that contains all but the last time point: $T' = T \setminus \{t_{\max}\}$. We are given:

- A set of rail **terminals** J . Each terminal $j \in J$ has the following properties:
 - **Storage cost** $c_j^S \in \mathbb{Q}$ for storing a single container for one time unit (time interval $[t, t+1)$) at terminal j .
 - **Lifting cost** $c_j^L \in \mathbb{Q}$ for moving a single container from or onto some train at terminal j .
 - **Storage capacity** $\dot{C}_j^t \in \mathbb{N}$ is the maximum number of containers that can be stored at terminal j during time interval $[t, t+1)$, with $t \in T'$.
 - **Lifting capacity** $q_j^t \in \mathbb{Q}$ is the maximum number of containers that can be lifted from or onto trains at terminal j during a single time unit, i.e., time interval $[t, t+1)$ with $t \in T'$.

- A set of **trains** L . For each train (“line”) $l \in L$, we have:
 - **Transport cost** $\check{c}_l^t \in \mathbb{Q}$ is the price per tonne transported on train l for time interval $[t, t + 1)$ with $t \in T'$.
 - **Transport capacity** $\check{C}_l^t \in \mathbb{N}$ is the maximum number of containers that can be transported on train l at time interval $[t, t + 1)$ with $t \in T'$.
 - **Weight capacity** $\check{W}_l^t \in \mathbb{N}$ is the maximum weight that train l can load and carry during time interval $[t, t + 1)$ with $t \in T'$.
 - **Home terminal** $\eta^l \in J$ is the terminal where train l starts and ends its round trips.
- A set of potential **Round trips**

$$R^l = \{ R_1^l, R_2^l, \dots, R_{d_{\max}^l}^l \}$$

that train l can take. The d -th **round trip** (with $d \in D^l = \{1, \dots, d_{\max}^l = |R^l|\}$) is defined by a sequence of stops π_p^{ld} , with $p_{\max} = |R_d^l|$:

$$R_d^l = (\pi_1^{ld}, \pi_2^{ld}, \dots, \pi_{p_{\max}}^{ld})$$

Each **stop** $\pi_p^{ld} = (j_p^{ld}, A_p^{ld}, B_p^{ld}) \in R_d^l$ has

- * **Halting terminal** $j_p^{ld} \in J$, i.e., the terminal where train l stops at,
- * **Arrival time** $A_p^{ld} \in T$ and leaves at, and
- * **Departure time** $B_p^{ld} \in T$.

This yields the precedence $A_p^{ld} < B_p^{ld} < A_{p+1}^{ld}$. Time $[A_p^{ld}, B_p^{ld})$ is the *dwell time* of stop π_p^{ld} of train l at halting terminal j_p^{ld} .

Each round trip R_d^l starts and ends at its home terminal, i.e., for any $d \in D^l$ we have $j_1^{ld} = j_{p_{\max}}^{ld} = \eta^l$. Furthermore, for two consecutive round trips $R_d^l, R_{d+1}^l \in R^l$, we have that the last stop of R_d^l is the first stop of R_{d+1}^l , formally: $\pi_{p_{\max}}^{ld} = \pi_1^{ld+1}$. Additionally, each round trip must contain at least three stops: The two stops at its home terminal, and at least one additional stop elsewhere, formally $|R_d^l| \geq 3$.

The union of all round trips gives us the train’s **plan** Π^l :

$$\Pi^l = \bigcup_{d \in D^l} R_d^l = \{ \pi_1^l, \pi_2^l, \dots, \pi_{p_{\max}}^l \}$$

For each of these round trips, we are further given

- * **Round trip costs** $\check{c}_d^l \in \mathbb{Q}$ that have to be paid when train l makes the round trip R_d^l .

- A set of **containers** I . Each container $i \in I$ has

- An **origin terminal** $o_i \in J$ where it is initially placed,
- **Destination terminal** $d_i \in J$ to which it has to be delivered,
- **Release time** $R_i \in T$ that is the earliest availability at o_i , and
- **Deadline** $D_i \in T$, i.e., the time at which container i has to be at d_i ,
- **Weight** $w_i \in \mathbb{N}$, and
- **Revenue** $r_i \in \mathbb{Q}$ that is collected when we deliver container i as specified.

3.1.1 Goal

We want to find a shipping schedule, that — given the implicit constraints — maximizes the collected revenues minus the required shipping and round trip costs. Informally, we want to deliver as many container as possible, as long as the round trip costs of the used trains and the shipping costs of the delivered containers pay off.

Notation

We usually use the hat-symbol $\hat{\cdot}$ or the visually similar single-dot notation $\dot{\cdot}$ to indicate that a symbol is related to *terminals* (e.g., \hat{V} , \hat{A} , \hat{C}_a , or \dot{C}_j^t).

In contrast, we often use the bar-symbol $\bar{\cdot}$ or the double-dot notation $\ddot{\cdot}$ to emphasize a relation to *trains* (e.g., \bar{V} , \bar{A} , \bar{C}_a , \bar{W}_a , or \ddot{C}_l^t , \ddot{W}_l^t).

Furthermore, we use the tilde-symbol $\tilde{\cdot}$ to indicate a relation to lifting and halting of trains at terminals (e.g., \tilde{A} , \tilde{L} , see later).

3.1.2 Real-World Aspects and Remarks

In this brief section, we want to escape the scientific ivory tower and bring the theoretical and mathematical formulation stated above into the view of the real world and highlight related practical aspects. Our problem was defined in this way to capture the real-world as realistically as possible — including, e.g., the lifting of containers, their availability time windows, and lifting and storage capacities — while keeping the level of details reasonable for solving it in practice (ignoring, e.g., delays of trains).

Changing Values Over Time

Note that we defined storage, transport, and weight capacities, as well as the transport costs, in such a way that their value can change over time. This enables us to model the involvement of third parties that might also use the considered terminals or trains, hence restricting the respective capacities.

Block Trains

In contrast to so-called wagon load trains, which are a composite of multiple single wagons, we assume that our trains are operated as block trains (sometimes called “unit

trains”). Block trains are operated as a whole, i.e., no wagons are (un-)coupled at any stops, and containers can be lifted onto and off the individual wagons. We consider block trains because ordering this type of trains is cheaper in most cases (on average per container) than ordering individual container transportation, see, e.g., [63, p. 21]. Schönemann [58] states that “Block trains ideally perform as shuttle trains with a fixed wagon set with regular recurring operating times” [58, p. 19].

In contrast to current practice, the block trains that we consider are not operated between only two stops, but instead halt at additional terminals where containers can be lifted onto or off the train. This enables more flexible routing of containers, and allows for more stops to be covered by a single train, potentially avoiding additional trains or container shipment to these stops by trucks.

Single Stacking of Containers

We only consider single-stacked containers on trains, as double-stacking is currently not possible on European trains (see, e.g., Rodrigue and Notteboom [56, p. 501]).

Round Trips

Informally, the *potential round trips* mentioned above look as follows: Some train leaves a certain terminal, its “home terminal”, in the beginning, travels along intermediate stops to another target stop, and then returns to its origin terminal. For instance, if we consider a train connection between Hamburg (Germany) and Vienna (Austria) with a single intermediate stop at Berlin (Germany), and we assume that Hamburg is the train’s home terminal, there are three different round trips that we can possibly construct: Hamburg-Berlin-Vienna-Berlin-Hamburg, Hamburg-Vienna-Berlin-Hamburg, and Hamburg-Berlin-Vienna-Hamburg.

Note that round trips do not have to take place, hence we called them “potential” round trips. If we indeed decide that a round trip takes place, we call it an *active* round trip.

For non-active round trips, we assume that the train is parked or can be used for other purposes during that time, hence no additional costs accumulate (or have to be respected separately).

Note that in theory, our model allows different round trips per train, i.e., a train could visit different locations during different round trips. Nevertheless, we are considering the case where all round trips are identical, as usually block trains are operated as shuttle trains with recurring schedules.

Post-Processing — Saving Storage Costs with Cooperating Customers

When considering a solution to our problem, it is often the case that containers arrive at their origin terminal (at their release time) some time before they are actually shipped, or are shipped to their destination terminal earlier than picked up (at their deadline). This causes storage costs until these containers are first shipped (or finally picked up).

In a post-processing optimization step, we can offer customers a discount if they are able to deliver (or pick up) these containers later (earlier), as we can save the related storage costs.

Post-Processing — Overestimating Number of Lifts

Another post-optimization step can be done by observing that containers can sometimes be lifted directly from one train onto another, requiring only one lifting operation instead of two. In these cases, we can save the costs for one of the lifting operations if the respective terminal allows it. Overestimating the number of lifts seems appropriate due to the fact that in the real-world, direct reloading of containers from one train to another is not common.

Remarks on Wording

In the following, we use the term “shipping” in contrast to “transportation” of containers: The latter refers to a container being transported on a specific train, while “shipping” describes the overall process of shipping the container, including lifting, storage and transportation (on trains). Note that we only consider train shipping and *neglect* how containers arrive or are picked up at terminals, most likely by trucks or ships¹.

Similarly, we use the term “terminal” in contrast to “stop”: A terminal (or station) has a specific location, while a stop describes the state of a train halting at some terminal. So a stop always has an associated time, train and terminal, while a terminal is not associated with times or trains directly.

3.2 Problem Representation — Time-Expanded Network

In this section, we present a mathematical representation for our problem. We use a time-expanded directed network with multiple (parallel) arcs to model it. The general idea is to construct the network in such a way that the shipping schedule of containers is represented by a *path* from a source to a sink vertex in our network, i.e., containers will “travel” along arcs. We will model this by sending a commodity flow for each container along its path through the network.

3.2.1 Time Points

We start by introducing the following *action time point sets*:

- For terminals $j \in J$, we define the set of **terminal action time points** $\hat{T}(j)$ that contains the time points where some train $l \in L$ arrives or leaves terminal j :

$$\hat{T}(j) = \{ A_p^l, B_p^l \mid l \in L, (j_p^l, A_p^l, B_p^l) \in \Pi^l : j = j_p^l \} \subseteq T$$

¹Note that our model is potentially able to capture some of these trips as well if we are able to encode their respective trips by our notion of round trips.

- Analogously, for trains $l \in L$, we define the **train action time points** $\bar{T}(l)$ that contain the time points where train l arrives or leaves some terminal j :

$$\bar{T}(l) = \{ A_p^l, B_p^l \mid (j_p^l, A_p^l, B_p^l) \in \Pi^l \} \subseteq T$$

Furthermore, we define the **reduced time point sets** $\hat{T}'(j)$ and $\bar{T}'(l)$ that contain all but the last time points, formally $\hat{T}'(j) = \hat{T}(j) \setminus \{\max \hat{T}(j)\}$, and $\bar{T}'(l) = \bar{T}(l) \setminus \{\max \bar{T}(l)\}$.

We further introduce the **successor functions** $\text{succ}_j(t) : \hat{T}'(j) \rightarrow \hat{T}(j)$ and $\text{succ}_l(t) : \bar{T}'(l) \rightarrow \bar{T}(l)$ that give us the subsequent time point of a given time point t in one of the action time point sets $\hat{T}(l)$ and $\bar{T}(l)$ (formally $\text{succ}_j(t) = \min \hat{T}(j) > t$, and analogously $\text{succ}_l(t) = \min \bar{T}(l) > t$). Note that for trains, we have that $\text{succ}_l(A_p^l) = B_p^l$ (for stop $\pi_p^l \in \Pi^l$), and $\text{succ}_l(B_p^l) = A_{p+1}^l$ (for $\pi_p^l \in \Pi^l \setminus \{\pi_{p_{\max}}^l\}$). In the general case ($|L| > 1$), this does not hold for terminals.

but the successor function is not defined on the last time point, we define the

3.2.2 Vertices

We can now define the time-expanded directed network with multiple arcs $N = (V, A)$. We define the vertex set $V = \check{V} \cup \hat{V} \cup \bar{V}$:

- The **source and sink vertices** \check{V} are exactly two vertices, the **source vertex** o (“origin”) and the **sink vertex** d (“destination”) of all container paths:

$$\check{V} = \{ o, d \}$$

- The **terminal vertices** \hat{V} contain one vertex for each terminal and each of its respective terminal action time points:

$$\hat{V} = \{ j^t \mid j \in J, t \in \hat{T}(j) \}$$

- The **train vertices** \bar{V} contain one vertex for each train and each of its respective train action time points:

$$\bar{V} = \{ l^t \mid l \in L, t \in \bar{T}(l) \}$$

Before being able to define the arc set A formally, we have to introduce further time points and vertices:

Origin and Destination Time Points and Vertices

Recall that each container i has a release time R_i and a deadline D_i . Note that in general, time points R_i and D_i are not directly represented by any vertex in our network (except when R_i or D_i happen to be action time points).

The earliest time point for a container to be moved is the earliest time point in $\hat{T}(o_i)$ after R_i at origin terminal o_i . We define the **origin time** (point) $t_i^R = \min \{t \in \hat{T}(o_i) \mid t \geq R_i\}$. Similarly, the latest time point for a container to be at its destination terminal d_i is the **destination time** (point) t_i^D that is the last possible time before D_i , i.e., $t_i^D = \max \{t \in \hat{T}(d_i) \mid t \leq D_i\}$. We call the time interval $[t_i^R, t_i^D]$ the **container's time horizon**.

Additionally, we extend our notation for simplicity reasons: We let the **origin vertex** o_i^* be the vertex that corresponds to the origin terminal o_i at origin time point t_i^R , i.e., $o_i^* = o_i^{t_i^R}$. We can omit the time point t_i^R in our notation as it is clear by the definition of the vertex itself w.r.t. the underlying network. Analogously, we have the **destination vertex** d_i^* that corresponds to the destination terminal d_i at destination time point t_i^D , formally $d_i^* = d_i^{t_i^D}$.

3.2.3 Arcs

We now define the arcs $A = \check{A} \cup \hat{A} \cup \bar{A} \cup \tilde{A}$ of our network $N = (V, A)$:

- **Source and sink arcs** \check{A} contain two arcs for each container $i \in I$: The first arc connects source vertex o to the origin vertices o_i^* . The second arc connects destination vertex d_i^* to sink vertex d . These arcs are required as we want to construct a path for each container that starts at o and has to reach d . Formally:

$$\check{A} = \{ (o, o_i^*)^i, (d_i^*, d)^i \mid i \in I \}$$

Note that this arc set might contain multi-arcs that can be distinguished by the superscript i .

- **Storage arcs** \hat{A} are the arcs representing that a container remains at a terminal for two consecutive terminal action time points:

$$\hat{A} = \bigcup_{j \in J} \hat{A}(j)$$

where $\hat{A}(j)$ are the storage arcs of an individual terminal j :

$$\hat{A}(j) = \{ (j^{t_1}, j^{t_2}) \mid t_1 \in \hat{T}'(j), t_2 = \text{succ}_j(t_1) \}$$

- **Transport arcs** \bar{A} are the arcs representing containers remaining on trains for two consecutive train action time points:

$$\bar{A} = \bigcup_{l \in L} \bar{A}(l)$$

where $\bar{A}(l)$ are the transport arcs of a single train $l \in L$:

$$\bar{A}(l) = \{ (l^{t_1}, l^{t_2}) \mid t_1 \in \bar{T}'(l), t_2 = \text{succ}_l(t_1) \}$$

A container is either being transported on a train (in between stops), or it remains on a train during halting.

- **Lifting arcs** \tilde{A} are the arcs representing containers being lifted from (or onto) a train to (or from) a terminal, i.e.,

$$\tilde{A} = \bigcup_{l \in L} \tilde{A}(l)$$

The arc set $\tilde{A}(l)$ of train $l \in L$ are the arcs that represent containers to be lifted from or onto train l . For each stop $\pi_p^l = (j_p^l, A_p^l, B_p^l) \in \Pi^l$, we create multiple arcs from train l to terminal j at the train's arrival time A_p^l , and the other way around — from terminal j to train l — at the train's departure time B_p^l .

The multiple parallel arcs are required for modeling lifting capacity constraints when multiple trains are present at the same terminal during overlapping time intervals (see Section 3.3). For a specific stop π_p^l of train l , we create an arc pair for each terminal action time point t within the dwell time of train l at stop π_p^l , i.e., every time another train arrives or leaves during the duration of the stop, we introduce a pair of arcs, indicated by the superscript t , at arrival and departure time of train l and stop π_p^l :

$$\begin{aligned} \tilde{A}(l) = \{ (l^{t_1}, j^{t_1})^t, (j^{t_2}, l^{t_2})^t \mid \\ (j_p^l, A_p^l, B_p^l) \in \Pi^l, j = j_p^l, t_1 = A_p^l, t_2 = B_p^l, t \in \hat{T}(j) \cap [t_1, t_2] \}. \end{aligned}$$

Note that we have at least two arcs for each stop $\pi_p^l = (j_p^l, A_p^l, B_p^l) \in \Pi^l$ of train l , i.e., $(l^{A_p^l}, j^{A_p^l})^{A_p^l}$ and $(j^{B_p^l}, l^{B_p^l})^{B_p^l}$, as these lifting arcs are required even if there is no other train present during stop π_p^l . Furthermore, note that we excluded departure time B_p^l from the considered time horizon in the set construction, as otherwise we would have two additional arcs when, e.g., considering this simple case with just a single train stopping at some terminal.

As we do not want to optimize the reloading/lifting process at each terminal — this is a complex problem on its own —, we simply assume that all containers that should be loaded are lifted onto trains at its departure time. Similarly, all containers that should be unloaded are lifted off the train at its arrival time. Furthermore, we assume that we have to pay for the storage of unloaded containers at terminals starting with the arrival time of the train. The transport costs of a container start with the departure of the train, and end with its arrival.

Together with the source and sink arcs, the lifting arcs are the only arc set in our representation that potentially contains multiple parallel arcs.

Example Fig. 3.1 shows an exemplary part of a time-expanded network for a single terminal j and two trains l_1, l_2 with overlapping dwell times. The nodes correspond to the action time points of the respective terminal and trains. The four considered action time points are the following: The first action time point $t_1 = A^{l_1} \in \hat{T}(j), \bar{T}(l_1)$ corresponds to the arrival of the first train l_1 at terminal j . The second train l_2 arrives

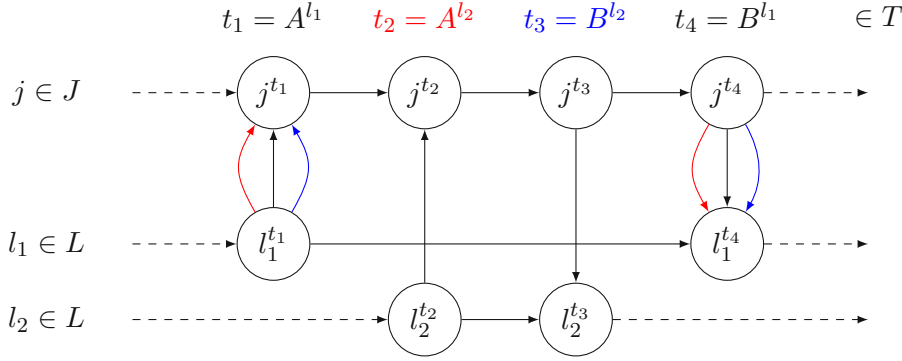


Figure 3.1: Example of storage, transport and lifting arcs for a single terminal j and two trains l_1, l_2 with overlapping dwell times. The red arcs are introduced because the arrival of train l_2 at time point t_2 lies within the dwell time of train l_1 , which halts at the same terminal. The blue arcs are introduced for the same reason with respect to the departure of train l_2 .

at the terminal at action time point $t_2 = A^{l_2} \in \hat{T}(j), \bar{T}(l_2)$, and departures again at $t_3 = B^{l_2} \in \hat{T}(j), \bar{T}(l_2)$. At last, the first train l_1 departs at $t_4 = B^{l_1} \in \hat{T}(j), \bar{T}(l_1)$.

The horizontal arcs are the storage arcs for terminal $j \in J$, and the transport arcs for the two trains $l_1, l_2 \in L$. The vertical (or bent) arcs are the lifting arcs. The overlapping dwell times of the two trains cause multiple (parallel) lifting arcs for the first train l_1 . During the dwell time of l_1 , the arrival of train l_2 takes place at time point t_1 . Therefore, the red arcs are introduced. Similarly, the blue arcs are introduced as the departure of train l_2 also lies within train l_1 's dwell time. As there are no trains that arrive or leave during train l_2 's dwell time, no additional arcs are added for this train.

Costs Along Arcs

Recall that we want to represent container schedules by paths (or commodity flows) through our network. Therefore, we define costs (and capacities) along arcs that have to be respected when they are used on the path of container i . These arcs represent storing, lifting or the transportation of containers:

If $a = (o, o_i^*)^i \in \check{A}$	$c(a, i) = c_{o_i}^S \cdot (t_i^R - R_i)$	(source costs)
If $a = (d_i^*, d)^i \in \check{A}$	$c(a, i) = c_{d_i}^S \cdot (D_i - t_i^D)$	(sink costs)
If $a = (j^{t_1}, j^{t_2}) \in \hat{A}$	$c(a, i) = c_j^S \cdot (t_2 - t_1)$	(storage costs)
If $a = (l^{t_1}, l^{t_2}) \in \bar{A}$	$c(a, i) = w_i \cdot \sum_{t \in [t_1, t_2] \subseteq T} \check{c}_l^t$	(transport costs)
If $a = (l^{t_1}, j^{t_1})^t \vee (j^{t_2}, l^{t_2})^t \in \tilde{A}$	$c(a, i) = c_j^L$	(lifting costs)

Recall that due to the introduction of vertices related to time points, time intervals $[R_i^i, t_i^R]$ and $(t_i^D, D_i]$ w.r.t. container i are not represented by storage arcs. Thus, the

related storage costs for container i during these time intervals are not captured directly by the storage arcs. Instead, we collect them via the source (and sink) arcs, which are the storage costs accumulating in the not yet considered time interval $[R_i, t_i^R)$ (and $(t_i^D, D_i]$).

No transport costs during halting When a train is halting, we set the transport costs to zero, formally:

$$\check{c}_l^t = 0 \quad \forall l \in L, \forall t \in \{t \in [A_p^l, B_p^l) \mid \pi_p^l = (j_p^l, A_p^l, B_p^l), \pi_p^l \in \Pi^l\}$$

Capacities Along Arcs

Note that we defined the terminal's storage capacities \dot{C}_j^t ($j \in J$) and each train's transport capacities \check{C}_l^t ($l \in L$), and weight capacities \ddot{W}_l^t it can carry per time interval $[t, t+1)$. Analogously to the costs along arcs, the capacity and weight restrictions can be considered along arcs between two time points $t_1 < t_2$, omitting time points in between. Therefore, we define:

$$\begin{aligned} \hat{C}_a &= \min_{t \in [t_1, t_2]} \dot{C}_j^t, \forall j \in J, \forall a = (j^{t_1}, j^{t_2}) \in \hat{A}(j) && \text{(storage capacity along arcs)} \\ \bar{C}_a &= \min_{t \in [t_1, t_2]} \check{C}_l^t, \forall l \in L, \forall a = (l^{t_1}, l^{t_2}) \in \bar{A}(l) && \text{(transport capacity along arcs)} \\ \bar{W}_a &= \min_{t \in [t_1, t_2]} \ddot{W}_l^t, \forall l \in L, \forall a = (l^{t_1}, l^{t_2}) \in \bar{A}(l) && \text{(weight capacity along arcs)} \end{aligned}$$

3.2.4 Objective

Given a specific problem instance, we can specify our objective w.r.t. the time-expanded network $N = (V, A)$:

For each container $i \in I$, we try to find a path from o to d in our network, while respecting all implicit (capacity and time) constraints and costs. We only collect the revenue of containers for which we found a path, while only using arcs of round trips that are active (i.e., that we paid for). We maximize the overall collected revenues while simultaneously minimizing the costs.

3.2.5 Size of the Time-Expanded Network

Theorem 1. *The time-expanded network is polynomial in its size.*

Proof. We claim that our time-expanded network $N = (V, A)$ is polynomial in its size w.r.t. the input parameters $t_{\max}, |J|, |L|, |I|$, and the specified round trips. In the worst case, the number of vertices $|V| \leq 2 + t_{\max} \cdot (|J| + |L|)$ is clearly polynomial. Therefore, the number of non-multiple arcs is also polynomial. The only arcs that allow multiple arcs are source and sink arcs, and lifting arcs. The number of source and sink arcs is at most $|I|$ (each). Completely overestimating the number of lifting arcs still yields a

polynomial size: Assume that all trains overlap at all time points, and for all trains, we introduce two additional lifting arcs for all other trains at all time points (and two just for the lifting of the considered train).² This still yields a polynomial bound: $t_{\max}^2 \cdot 2 \cdot |L|^2$, hence our network is polynomial in its size. \square

3.3 ILP Problem Formulation

In this section, we present an ILP model for our problem that uses a commodity flow formulation to model the paths of containers in our time-expanded network. For that reason, we have to introduce further sets before we can define our variables and the necessary constraints.

3.3.1 Required Definitions

As it is impossible for containers to be shipped before their release time or after their deadline — or, more specifically, before their respective origin and after their destination time —, we introduce **container arcs** $A(i)$ for each container $i \in I$. It only contains those arcs that have to be considered w.r.t. the container's time horizon, i.e., arcs within origin time t_i^R and destination time t_i^D of container i . Later, we will use this set to define container arc variables only along the arcs that are relevant for the container. The container arcs $A(i)$ are the union of the following sets:

$$A(i) = \check{A}(i) \cup \hat{A}(i) \cup \bar{A}(i) \cup \tilde{A}(i)$$

Thus, for every container i , we have:

- **Container source and sink arcs** $\check{A}(i)$ are the two source and sink arcs specifically created for container i :

$$\check{A}(i) = \{ (o, o_i^*), (d_i^*, d) \} \subseteq \check{A}$$

- **Container storage arcs** $\hat{A}(i)$ are the storage arcs within container i 's time horizon, representing that the container remains at some terminal $j \in J$:

$$\hat{A}(i) = \{ (j^{t_1}, j^{t_2}) \in \hat{A} \mid t_1, t_2 \in [t_i^R, t_i^D] \}$$

- **Container transport arcs** $\bar{A}(i)$ are the arcs within container i 's time horizon, representing that the container is transported on some train $l \in L$:

$$\bar{A}(i) = \{ (l^{t_1}, l^{t_2}) \in \bar{A} \mid t_1, t_2 \in [t_i^R, t_i^D] \}$$

- **Container lifting arcs** $\tilde{A}(i)$ are the lifting arcs representing container i being lifted from or onto some train $l \in L$ at a specific terminal $j \in J$, while the corresponding action time point as well as the arc itself lie within the container's time horizon:

$$\tilde{A}(i) = \{ (l^{t_1}, j^{t_1})^t \in \tilde{A} \mid t, t_1 \in [t_i^R, t_i^D] \} \cup \{ (j^{t_2}, l^{t_2})^t \in \tilde{A} \mid t, t_2 \in [t_i^R, t_i^D] \}$$

²We actually introduce far less lifting arcs.

Note that these container arcs are not the path that a container takes. Instead, these arcs are the *possible* arcs that a container *might* be sent along, i.e., the arcs that have to be considered when planning a path for the container. We will later define constraints related to individual containers along these arcs only.

Furthermore, we define some more arc sets in order to concisely formulate all constraints of our ILP model:

- The **terminal lifting arcs of action time point** $\hat{A}(j, t)$ of terminal j and action time point $t \in \hat{T}(j)$ contain the lifting arcs — potentially selected out of multiple parallel arcs — that are related to action time point t . It contains exactly those lifting arcs associated with t and all trains that are present at terminal j at that time point t . We will later define lifting capacity constraints along these arcs. Formally, we have:

$$\hat{A}(j, t) = \{ (l^{t_1}, j^{t_1})^t \in \tilde{A}(l) \} \cup \{ (j^{t_2}, l^{t_2})^t \in \tilde{A}(l) \}$$

- The **round trip transport arcs** $\bar{A}_R(l, d)$ capture the transport arcs of round trip R_d^l of train l . These are the transport arcs that lie within the train's arrival time A_1^{ld} of the round trip's first home stop $(j_1^{ld}, A_1^{ld}, B_1^{ld}) = \pi_1^{ld} \in R_d^l$ and the departure time $B_{p_{\max}}^{ld}$ of the round trip's last home stop $(j_{p_{\max}}^{ld}, A_{p_{\max}}^{ld}, B_{p_{\max}}^{ld}) = \pi_{p_{\max}}^{ld} \in R_d^l$:

$$\bar{A}_R(l, d) = \{ (l^{t_1}, l^{t_2}) \in \bar{A}(l) \mid t_1, t_2 \in [A_1^{ld}, B_{p_{\max}}^{ld}] \}$$

- The **container's placeholder arcs** $\hat{A}(i)$ of each container i are required to model terminal capacity constraints correctly. For each container, there are at most two of these arcs: the incoming storage arc of the origin vertex, and the outgoing storage arc of the destination vertex, which are arcs that lie outside the container's time horizon. Along these arcs, we have to reserve space for container i in case it is shipped. As these arcs lie outside the container's time horizon, they are not part of the container arcs $A(i)$ (and their related constraints that we will define). Therefore, we will need to consider them separately within the relevant capacity constraints. In detail, the first placeholder arc $(j_1^{t_1}, j_1^{t_2})$ is the storage arc spanning from a predecessor vertex (o_i^t) to the container i 's origin vertex $o_i^{t_i^R} (= o_i^*)$. Thus, it can be also represented by as $(o_i^t, o_i^{t_i^R})$. The second arc $(j_2^{t_3}, j_2^{t_4})$ is the storage arc spanning the container i 's destination vertex $d_i^{t_i^D} (= d_i^*)$ to the successor terminal vertex $(o_i^{t'})$, hence it can be written as $(d_i^{t_i^D}, d_i^{t'})$. Formally:

$$\begin{aligned} \hat{A}(i) = & \{ (j_1^{t_1}, j_1^{t_2}), (j_2^{t_3}, j_2^{t_4}) \in \hat{A} \setminus \hat{A}(i) \\ & \mid j_1 = o_i, j_2 = d_i, t_2 = t_i^R, t_3 = t_i^D, t_4 = \text{succ}_{j_1}(t_1), t_4 = \text{succ}_{j_2}(t_3) \} \end{aligned}$$

Note that this set can contain just a single arc (or can even be empty) when $R_i = t_i^R$ (and) or $t_i^D = D_i$, but contains at most two arcs.

Additionally, we require the set of **relevant containers of an arc** $I(a)$, which contains the containers $i \in I$ for which arc a is included in the respective container arc set $A(i)$:

$$I(a) = \{ i \in I \mid a \in A(i) \}$$

3.3.2 Variables

Before introducing the ILP model, we define the following binary variables:

- Binary **container arc variable** x_a^i or x_{uv}^i is 1 if container $i \in I$ is shipped along arc $a = (u, v) \in A(i)$ on its path, i.e., it is transported/stored/lifted via that arc, and 0 otherwise.
- Binary **container variable** y_i is 1 if container $i \in I$ is *delivered*, and 0 otherwise. This means we have found a path from its origin to its destination terminal within the underlying time-expanded network.
- Binary **round trip variables** z_d^l is 1 if train $l \in L$ makes round trip R_d^l , i.e., we can ship (transport and lift) containers on the respective stops of the round trip, and we pay for the round trip. If $z_d^l = 1$, we refer to it as being an *active* round trip.

3.3.3 The Model

Finally, we can formulate our ILP model:

$$\max \sum_{i \in I} \left(y_i \cdot r_i - \sum_{a \in A(i)} x_a^i \cdot c(a, i) \right) - \sum_{l \in L} \sum_{d \in D^l} z_d^l \cdot \bar{c}_d^l \quad (3.1)$$

$$\text{s.t.} \quad - \sum_{(u,v) \in A(i)} x_{uv}^i + \sum_{(v,w) \in A(i)} x_{vw}^i = \begin{cases} y_i & v = o \\ -y_i & v = d \\ 0 & \text{otw.} \end{cases} \quad \forall v \in V, \forall i \in I \quad (3.2)$$

$$\sum_{i \in I(a)} x_a^i + \sum_{i \in I: a \in \hat{A}(i)} y_i \leq \hat{C}_a \quad \forall a \in \hat{A} \quad (3.3)$$

$$\sum_{i \in I(a)} x_a^i \leq \bar{C}_a \quad \forall a \in \bar{A} \quad (3.4)$$

$$\sum_{i \in I(a)} x_a^i \cdot w_i \leq \bar{W}_a \quad \forall a \in \bar{A} \quad (3.5)$$

$$\sum_{i \in I} \sum_{a^t \in \tilde{A}(i) \cap \hat{A}(j, t)} x_{a^t}^i \leq \sum_{t' \in [t, \text{succ}_j(t))} q_j^{t'} \quad \begin{matrix} \forall j \in J, \\ \forall t \in \hat{T}'(j) \end{matrix} \quad (3.6)$$

$$x_a^i \leq \sum_{d \in D^l: a \in \bar{A}_R(l, d)} z_d^l \quad \begin{matrix} \forall l \in L, \forall i \in I, \\ \forall a \in \bar{A}(i) \cap \bar{A}(l) \end{matrix} \quad (3.7)$$

$$x_a^i \in \{0, 1\} \quad \forall i \in I, \forall a \in A(i) \quad (3.8)$$

$$y_i \in \{0, 1\} \quad \forall i \in I \quad (3.9)$$

$$z_d^l \in \{0, 1\} \quad \forall l \in L, \forall d \in D^l \quad (3.10)$$

The **objective** function (3.1) maximizes container revenues while minimizing induced costs. The left term of the sum collects the revenues r_i of all delivered containers i (if $y_i = 1$). From this, we subtract the containers' shipping costs of all arcs that are used along each delivered container's path, as well as the costs of *active* round trips (if $z_d^l = 1$).

Constraints (3.2) are the **commodity flow/conservation** constraints: If container $i \in I$ is delivered, we send out $y_i = 1$ unit of flow from its origin vertex o_i^* . At the same time, its destination vertex d_i^* has to receive exactly the same (1) amount of flow. For non-origin and -destination vertices, we have the classic flow conservation constraints, i.e., the in-flow of the vertex has to be equal to its out-flow. For undelivered containers, we don't send out (or receive at d_i^*) any flow (as $y_i = 0$), hence there can be no flow within the network.

Note that the objective alone is enough to avoid unnecessary costs caused by cycles in any graph, as the existence of a (cost-positive) cycle violates the optimality of our solution. Nevertheless, it is also cycle-free by construction, hence we can guarantee that if $y_i = 0$ for some container i , there cannot exist flow on any arc a ($x_a^i = 1$).

Constraints (3.3), (3.4) and (3.5) ensure that **storage**, **transport** and **weight capacities** are not exceeded at any point. The additional term on the left-hand side of constraints (3.3) is responsible for reducing the storage capacities along arcs that are part of the placeholder arc set of any delivered container.

Constraints (3.6) ensure that the **lifting capacities** of every terminal are never exceeded. We consider each terminal j 's action time points $t \in \hat{T}'(j)$ with the corresponding spanning time interval $[t, succ_j(t))$. On the right-hand side of the inequality, we simply sum up the lifting capacities over all time points within the considered interval, as this corresponds to the maximum lifting capacity during that time period. The left-hand side simply counts the number of lifts of all container's lifting arcs $a^t \in \tilde{A}(i)$ associated with action time point t and terminal j .

With constraints (3.7), we only allow the shipping of containers along transport arcs of train l when a corresponding **round trip** is active. Note that restricting transport arcs also results in a corresponding restriction of the associated lifting arcs due to flow conservation constraints (3.2). Also, the sum on the right-hand side usually contains just one, and at most two variables. The latter is the case when we consider arcs within arrival and departure time of the first or the last stop of a round trip, i.e., when the train is at its home terminal. Recall that these stops appear in two round trips, e.g., the last stop is also present in the first stop of the subsequent round trip. Therefore, the sum captures both round trips as when either is active, the considered transportation arc can be used. In all other cases, the sum contains just a single variable, i.e., the one corresponding to the round trip containing the respective transport arc.

Finally, we have constraints (3.8), (3.9) and (3.10) defining the binary **domains** of the x -, y - and z -variables.

Flexibility of our Model

Note that our model is quite flexible, i.e., we can encode many problem variants.

For example, we can force the delivery of *all* containers by setting the container variables $y_i = 1$. In this case, our model just returns infeasibility if it is not possible to deliver all containers.

We can also force the model to try to deliver as many containers as possible by setting the container revenues to sufficiently large values. For example, we can choose the container revenues $r_i \geq M$, where M exceeds the maximum costs for operating all trains and shipping the most expensive container on its worst path. An appropriate value is, e.g., $M = \sum_{l \in L} \sum_{d \in D^l} z_d^l + \max_{i \in I} \sum_{a \in P(i)} c(a, i)$, where $P(i)$ is the maximum arc path of container i from o to d w.r.t. to costs $c(a, i)$ along the path (note that $a \in A(i)$).

We can also extend our model to other types of transportation, e.g., ships, not restricting it to rail. If these other modes are operated in round trips, it is easy to encode them in our model. Note that we can also respect (potential) single trips by just encoding them as a single round trip.

3.4 Complexity

To show the NP-hardness (of the decision variant³ of) the MCRP, we show a reduction of the well-known NP-complete SATISFIABILITY PROBLEM (SAT problem) to the MCRP. The NP-completeness of SAT was shown by Cook [17] in 1973, and independently by Levin [48] in 1973.

Recall the SAT problem for a propositional formula in conjunctive normal form: Given n variables $X = \{x_1, \dots, x_n\}$, corresponding literals $L = \{x, \bar{x} \mid x \in X\}$, and m clauses $C = \{C_1, \dots, C_m\}$, where each clause $C_\iota = \{l_1^\iota, \dots, l_{\max}^\iota\}$ contains some literals $l_1^\iota, \dots, l_{\max}^\iota \in L$, we want to find a variable assignment (“interpretation”) $\mathcal{I} : X \mapsto \{0, 1\}$ s.t. all clauses in C are satisfied, i.e., the formula $F = F_1 \wedge \dots \wedge F_m$, where $F_\iota = (l_1^\iota \vee \dots \vee l_{\max}^\iota)$ for $\iota \in \{1, \dots, m\}$, evaluates to true⁴.

3.4.1 Multi-Train Reduction

Given an arbitrary SAT instance, we construct an instance of our problem, formulated as a decision variant, as follows:

We introduce an *origin terminal* and a *destination terminal*, denoted by j^O and j^D . For each variable $x \in X$ we have *blocking terminals* j_x^B , and for each clause C_ι , we introduce *clause terminals*, denoted by j_ι^C . Each terminal has infinite storage and lifting capacity $\check{C}_j^t = q_j^t = \infty$ (or sufficiently large capacities, e.g., $m + n$).

We further introduce $2n$ trains, one for each positive and negative literal $l \in L$. All trains start at home terminal j^O , go along some clause terminals, arrive at j^D , and — in order to complete their round trips — end at home terminal j^O . Each of these $2n$ trains has an infinite capacity $\check{C}_l^t = \infty$ (or some sufficiently large number, e.g., m) and a constant maximum total container weight capacity of $\check{W}_l^t = m$.

We place a single *satisfying container* i_ι^S at each clause terminal j_ι^C (i.e., $o_{i_\iota^S} = j_\iota^C$) that has to be delivered to the destination terminal j^D (i.e., $d_{i_\iota^S} = j^D$). Furthermore, there is a *blocking container* i_x^B located at each blocking terminal j_x^B (i.e., $o_{i_x^B} = j_x^B$) that also have to be delivered to the destination terminal j^D (i.e., $d_{i_x^B} = j^D$). Note that we have exactly n satisfying and m blocking containers, so in total $n + m$ containers. Each satisfying container i_ι^S has weight $w_{i_\iota^S} = 1$, while the blocking containers i_x^B have weight $w_{i_x^B} = m$.

Recall that we have $2n$ trains, one for each positive or negative literal. Note that each train has maximum total weight capacity of $\check{W}_l^t = m = w_{i_x^B}$, hence they can either pick up exactly one blocking container, or up to all satisfying containers.

Now, we construct the round trips of the trains. Each train corresponding to a single (positive or negative) literal l and its corresponding variable x . Every train has a single

³In the decision variant, we don’t ask for the optimal objective. Instead, we ask if it is possible to obtain an objective that is higher than some predefined value.

⁴A positive literal $x \in L$ evaluates to true if $\mathcal{I}(x) = 1$, while its negative counterpart $\bar{x} \in L$ evaluates to true if $\mathcal{I}(x) = 0$. A clause C_ι and its corresponding formula F_ι evaluate to true if at least one of its literals $l \in C_\iota$ evaluates to true. The whole formula F evaluates to true if all subformulas F_ι evaluate to true.

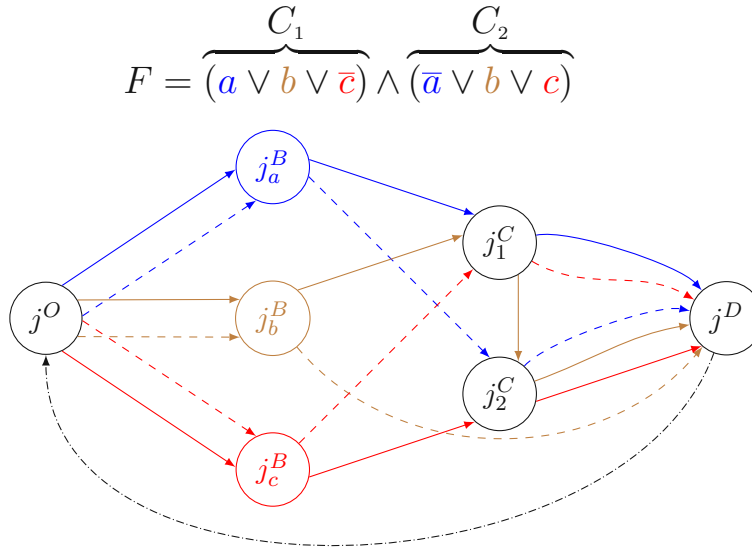


Figure 3.2: Example reduction for a Boolean formula with just two clause. The solid edges correspond to the trains used for the positive literal, the dashed edges are the negative literal trains. The black dash-dotted edge is the same for all (positive and negative) literal trains.

round trip with the following stops: It starts at j^O , continues to blocking terminal j_x^B , visits (in arbitrary order) all clause terminals j_i^C in whose clause C_i literal l appears, and finally reaches j^D . Afterwards, it returns to its home terminal. The construction of the rail network is illustrated in Fig. 3.2 for an small exemplary Boolean formula with two clauses.

In order to deliver all containers to the destination terminal j^D , the blocking container of each variable has to be transported by one of its two corresponding trains (positive and negative literal), while the other train can pick up satisfying containers from clause terminals, yielding a solution for SAT based on the trains transporting satisfying containers.

Note that we did not yet define our considered time horizon $T = [1, t_{\max}]$, release times and deadlines, or any other time-related values such as time points of each stop. We just have to choose t_{\max} to be sufficiently large (e.g., $2(3+m+n)$) to contain all stops of each each train, and we let all containers be available from the beginning ($R_i = 1$) to the end ($D_i = t_{\max}$).

We omit all costs (set them to zero), and let the revenue of each container $r_i = 1$. We claim now that the SAT-instance is satisfiable if and only if our constructed problem instance has an objective value of $m+n$, i.e., all containers can be delivered to the destination terminal j^D .

Correctness of the Reduction

As usual, we proof the correctness of our reduction by showing both directions of the reduction, i.e., we show that a positive SAT instance implies a positive instance of our problem, and the other way around. Note that for this section, “positive instance of our problem” refers to an objective of exactly $m + n$, i.e., we indeed delivered all containers.

Proof. “ \Rightarrow ” Given formula F , assume we have a positive instance for SAT. Then there exists some variable assignment $\mathcal{I}(x) \in \{0, 1\}$ for each variable $x \in X$ such that every clause C_l is satisfied, i.e., at least one of its literals is mapped to 1, i.e., $\mathcal{I}(l) = 1$, for some $l \in C_l$. Note that for complementary literals of variable $x \in X$, we trivially have $\mathcal{I}(x) = 1 - \mathcal{I}(\bar{x})$. In order to construct a positive MCRP instance from the SAT instance, we can now assign containers to trains in the following way: If variable $x = 1$ ($x = 0$) for some $x \in X$, we let the train associated to the positive (negative) occurrence of x pick up all still available satisfying containers i_l^S from the clause terminals it visits on its way and deliver it to j^D , while train associated to the complementary literal, i.e., the negative (positive) literal, picks up and delivers the blocking container i_x^B . Note that this is possible as every train has the capacities to transport either a single blocking, or up to all satisfying containers. We claim now (again) that we can indeed deliver all $m + n$ containers.

Assume otherwise. Then there is some container that was not delivered, being either a satisfying or blocking container. Assume it is a blocking container, i.e., one of the n blocking containers was not delivered. This assumption contradicts the fact that for each of the n variables, one of the two trains associated with the two complementary literals picks up the blocking container.

Assume now that instead, a satisfying container i_l^S could not be delivered. This is contradicted by the fact that some literal $l \in C_l$ is true, hence the corresponding train will have picked up this container.

As a result, we can indeed deliver all containers of our corresponding MCRP instance, given a positive SAT instance.

“ \Leftarrow ” For the second direction, we are given a positive instance of the MCRP, which is based on a SAT instance with given formula F . We show that we can construct a variable assignment \mathcal{I} satisfying F from the solution of this MCRP instance. Recall that we delivered all $n + m$ containers. This means that each blocking container i_x^B associated to variable x is delivered to j^D by a train associated to some literal l , while each train associated to the complementary literal l' delivers some satisfying containers. We choose the variable assignment $\mathcal{I}(x) = 1$ if and only if the train associated to the negative literal l' of variable x delivered container i_x^B . For example, if i_x^B is delivered by the train associated to positive literal l , we have $\mathcal{I}(x) = 0$. We claim now that formula F evaluate to true under interpretation \mathcal{I} .

Assume otherwise. Then there exists some clause C_l that is not satisfied. This implies that all literals $l \in C_l$ are falsified by \mathcal{I} , i.e., $\mathcal{I}(l) = 0$ for $l \in C_l$. But this contradicts the fact that container i_l^C was successfully delivered to j^D , as one of the trains associated to

literal l^v has to have picked it up due to the fact that we have a positive instance of our problem. As a consequence, F is indeed satisfiable. \square

To complete the correctness of the reduction, it remains to argue that our MCRP is indeed polynomial in its size w.r.t. the input SAT problem with m clauses and n variables. We have already shown the polynomial size of the used time-expanded graph in Section 3.2.5. Following the construction of our instance, this is clearly the case as the number of terminals $|J| = 2 + n + m$, the number of trains is $2n$, each with a single round trip with at most $3 + m$ stops, and the number of containers $|I| = n + m$, yielding a proper polynomial-sized reduction.

As a consequence, we have proven the following theorem:

Theorem 2. *The MCRP (as a decision problem) is NP-hard.*

Considering the existence of an (polynomial) ILP model for our problem, and knowing that (the decision variant of) Integer Linear Programming is itself NP-complete, we immediately get the following result:

Theorem 3. *The MCRP (as a decision problem) is NP-complete.*

Note that we can transform every decision problem into an optimization problem by using binary search (e.g., see Benoit et al. [8, p. 135]).

3.4.2 Single Train Reduction

We have seen a reduction using $2n$ trains. Instead, we can also use a single train that repeats round trip — $j^O, j_1^C, \dots, j_l^C, j^D$ and back to j^O — $2n$ times (i.e., it visits every clause terminal once for each (positive and negative) literal). All blocking containers are located at j^O , and during each trip, only one blocking container is available at j^O . Furthermore, every round trip corresponds to a specific literal. The lifting capacities of those clause terminals in whose clause the considered literal appears are 1, while all other clause terminals have lifting capacities of zero when the train is halting. As a result, we get the following corollary:

Corollary 3.1. *The MCRP (as a decision problem) is NP-complete for $|L| \geq 1$.*

Instance Generation and Experimental Setup

To our knowledge (and since we are considering a new optimization problem), no existing benchmark instances are available that can be used for or easily adapted to our problem. Therefore, we artificially created instances to investigate the performance and behavior of our problem. We aimed for realistic characteristics of all instance parameters (costs, capacities, weights, etc.). These were obtained through public resources, the research community, or were directly provided by our project partners.

This chapter contains detailed descriptions of the instance generation algorithm, the identification of real-world values for the instance parameters that we used in our benchmark experiments, and our experimental setup.

The instance generation process is divided into the following steps:

1. Choosing a set of **terminals** to be considered,
2. the generation of the underlying **rail network**,
3. the selection of appropriate parameters for the terminals,
4. the generation of **trains**, including their round trip schedules and their associated parameters, and
5. the generation of **containers** with their respective parameters.

In Section 4.1 we describe the generation of our rail network. In Section 4.2 we discuss the remaining steps of our instance generation process in detail, including the identification of real-world values that are used as parameters, and the algorithm for generating all benchmark instances. Finally, we describe our experimental setup and the benchmarking environment in Section 4.3.

4.1 Rail-Network

The first step of our instance generation process is the determination of the architecture of the underlying rail network that our instances are based on. We have to choose appropriate terminal locations, a *size* ($\in (0, 1]$) for each terminal, and have to decide on the existence and length of train routes that connect pairs of terminals.

4.1.1 Terminal Locations, Sizes, Types and Unification

After consulting with experts and project partners, we identified ten terminal locations within Europe as the most relevant for our study. For estimating the size and other parameters of these ten terminal locations, we collected properties of all existing freight terminals within these cities using `www.intermodal_terminals.eu` [39]. We grouped all existing terminals of a single location together (e.g., Vienna has three intermodal or rail freight terminals) and collected associated information such as track length, the number of lifting devices, or lifting capacities. We then created an artificial terminal per location to represent the collected information, merged into a single terminal. These (artificial) terminals are the ones we use for our instances.

We assigned a relative size to each terminal based on the total track length of all available loading tracks. This feature appeared to be the best choice for estimating the terminals' sizes as it was available for all terminals.

We group these ten terminals into two types, which will be relevant in the next section: (sea) *port* terminals P and *city* (non-port, inland) terminals C , forming the set of terminals $J = P \cup C$.

Table 4.1 shows all used terminals, their corresponding country codes, if they are ports or cities, and their sizes. Table 1 in the appendix gives an overview of specific properties and the sizes derived from each of them for each considered (artificial) terminal. Table 2 contains the complete information of all individual terminal properties taken from [39].

4.1.2 Determining the Rail Network Using an ILP

Given our ten considered terminals with their respective sizes, we have to determine the underlying rail network. We calculate the minimum rail distance between each terminal pair w.r.t. the underlying physical real-world rail network that was provided by our project partners. This results in a complete graph that consists of ten vertices representing the terminals, and with distance information along all edges.

Although we could use all edges of the complete graph as for our underlying rail network, this would result in a large number of trains. Therefore, we have to determine a subset of all edges that will connect our terminals to form the final rail network. Note that for our instances, we only consider subsets of the ten terminals for our instances, depending on the choice of input parameters $|P|$ and $|C|$.

As hub-and-spoke networks are often in place, we want to model a similar architecture where we use the ports as hubs. Therefore, we want to have train lines that connect every

Table 4.1: Terminals of the rail network

Terminal	Abbrev.	Country Code	City (C) / Port (P)	Size
Hamburg	HMB	DE	P	1.00
Rotterdam	RTD	NL	P	0.78
Antwerp	AWP	BE	P	0.44
Bremerhaven	BHV	DE	P	0.14
Duisburg	DUI	DE	C	0.66
Munich	MUN	DE	C	0.30
Vienna	VIE	AT	C	0.23
Salzburg	SZB	AT	C	0.12
Berlin	BER	DE	C	0.11
Enns	ENS	AT	C	0.09

port-city pair (pc -pair). We further want to investigate trains that are operated in a way that they connect more than two cities. Thus, for each pc -pair, we consider the direct connection $\{p, c\}$ and want to allow detours over additional cities (e.g., $p - x - y - c$, where x, y are some other cities) as long as the resulting tour (w.r.t. its track length) does not exceed a defined threshold τ of the direct distance of the connection. At the same time, we try to minimize the length of all rail tracks in the network.

Formally, we can describe the problem as follows: Given a (complete) graph $G = (V, E)$, where the vertices are the union of port terminals P and city terminals C , formally $V = P \cup C$, and distances $d(e)$ for each edge $e \in E$, find a subset of edges $E' \subseteq E$ with minimal total distance s.t. for every pair (p, c) with $p \in P$, $c \in C$, there exists a path in E' . The accumulated distance of each path must not exceed $1 + \tau$ of the direct edge distance $d(\{p, c\})$. Here, $\tau \geq 0$ is the maximum factor that each pc -path is allowed to exceed the distance $d(\{p, c\})$ of the direct connection.

For solving this problem, we use an ILP with a multi-commodity flow formulation to model the path for each pc -pair. It requires the following variables:

- Binary variable x_e or x_{uv} is one if its corresponding edge $e = \{u, v\} \in E$ is part of our solution, i.e. $e \in E'$.
- Binary variable y_a^{pc} or y_{uv}^{pc} is one if arc $a = (u, v) \in A$ is part of the directed flow path from $p \in P$ to $c \in C$, where A is the directed arc set of our considered edges E (formally $A = \{(u, v), (v, u) \mid \{u, v\} \in E\}$).

The ILP looks as follows:

$$\min \sum_{e \in E} d(e) \cdot x_e \quad (4.1)$$

$$\text{s.t.} \quad - \sum_{(u,v) \in A} y_{uv}^{pc} + \sum_{(v,w) \in A} y_{vw}^{pc} = \begin{cases} 1 & \text{if } v = p \\ -1 & \text{if } v = c \\ 0 & \text{otw.} \end{cases} \quad \forall v \in V, \forall (p, c) \in P \times C \quad (4.2)$$

$$\sum_{a \in A} d(a) \cdot y_a^{pc} \leq d(\{p, c\}) \cdot (1 + \tau) \quad \forall (p, c) \in P \times C \quad (4.3)$$

$$y_{uv}^{pc} \leq x_{uv} \quad \forall (u, v) \in A \quad (4.4)$$

$$x_e \in \{0, 1\} \quad \forall e \in E \quad (4.5)$$

$$y_a^{pc} \in \{0, 1\} \quad \forall a \in A, \forall (p, c) \in P \times C \quad (4.6)$$

The objective (4.1) minimizes the overall distance of all selected edges in our solution. Constraints (4.2) are flow conservation constraints for every pc -pair. For each pc -pair, p sends out one unit of pc -flow, c consumes one unit of pc -flow, and for all other nodes, the amount of inflow has to be equal to the amount of outflow. Constraints (4.3) ensure that for every pc -pair, the total distance of the arcs on the path defined by the respective pc -flow does not exceed the maximum allowed distance $d(\{p, c\}) \cdot (1 + \tau)$. Constraints (4.4) ensure that flow can only exist on selected edges. Constraints (4.5) and (4.6) define the binary domains of the variables.

Figure 1 in the appendix shows the resulting rail networks for $\tau \in \{0.00, 0.05, 0.10, 0.15\}$. We can observe that the number of connections decreases for higher τ values as a low τ value restricts the total distance of each pc -path more. For example, at $\tau = 0.00$, there is a direct connection from HMB to VIE, but at $\tau = 0.05$, we can connect this pair via BER, forming the connection HMB-BER-VIE.

We use the result with $\tau = 0.10$ as the underlying rail network for our instances, which is illustrated in Fig. 4.1.

Note that due to the complexity of our problem, we did not actually generate instances including all ten terminals for our benchmark. The maximum number of used terminals is seven, see Section 4.3. The two resulting rail networks with $|J| = 7$ are shown in Fig. 4.2.

4.2 Instance Parameters

In this section, we list all parameters of our instance generation process. As already mentioned, these parameters are taken from existing literature or were provided by our project partners (except for some basic input parameters of our instance generation process like the number of considered containers).

Beside the architecture of the underlying rail network, see Section 4.1, the following parameters were used as an input for our instance generation process:

- $|I|$, the number of containers,

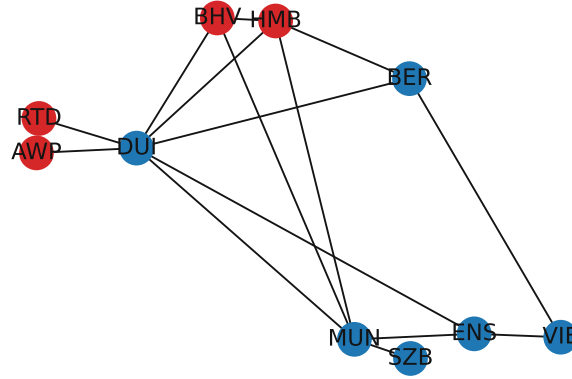


Figure 4.1: Resulting network for $\tau = 0.1$, ports are colored red, cities blue.

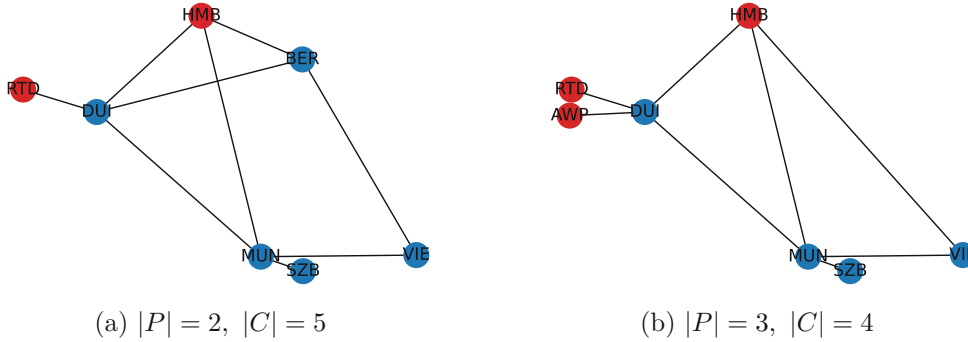


Figure 4.2: Maximum considered rail networks with seven terminals.

- $|J| = |P| + |C|$, the number of considered terminals of the rail network, where $|P|$ and $|C|$ are the number of ports and cities (see Section 4.1),
- the considered *time horizon* h , given in days,
- the length of container *buffer time windows* buff_i , i.e., the additional buffer time beside the expected (minimum) shipping time of a container, defining release time and deadline,
- the *revenue factor* ρ , a multiplier for each container revenue, and
- the *train scheduling method* for all trains, i.e., whether we schedule daily trains or just have a single recurring train per terminal (see Section 4.2.2).

Non-varying costs and capacities over time In Section 3.1, we defined all capacities and some costs in a way that they can vary over time, meaning that they are time-dependent on time point $t \in T$, consider, e.g., the storage capacity \hat{C}_j^t . This was done to allow our model to capture varying constraints of these resources that might occur

in real-world scenarios. Nevertheless, we use the same value for all time points in our instances, formally $c^t = c^{t+1}$, $\forall t \in T'$, $c \in \{\dot{C}_j, q_j, \ddot{C}_l, \ddot{W}_l\}$.

Considered time horizon For the length of the considered time horizon h , we choose multiples of seven days (see Section 4.3). For our instances, we use time units of $\mu = 15$ minutes, hence $t_{\max} = \text{number of days} \cdot 96$.¹

4.2.1 Terminal Parameters

Our terminals $j \in J = P \cup C$ have the following parameters, presented together with their respective values that they are based on, and corresponding sources:

Storage costs WienCont Container Terminal GmbH [64] offers the storage of empty containers for 3 € and charges 6.10 € for loaded containers per day. CMA CGM [15] list storage costs of many locations in Europe. In most cases, the first two (three, five or ten) days (including arrival day) are free of charge, followed by daily charges, usually in the range of 11 € (minimum 8 €, maximum 35 €), and often the price increases (e.g., doubles) after 5-21 days. Other parties charge 10.40 € per day, but the first day is free of charge. For simplicity, we use the same storage costs of about 10 € per day for all terminals $j \in J$, resulting in storage costs of $c_j^S = 0.104$ € per 15-minutes time unit².

Lifting costs According to Black et al. [10, p. 41f, 81], the costs for a single movement from rail to rail of a single container range from 14 € to 38 €, with an average of 27 €. This agrees with the information of Beresford [9, p. 240], who states varying costs of about 30 € for Europe, and 36 € for the UK. WienCont Container Terminal GmbH [64] charge 32.40 € for handling a single container, or 58 € for a two-handling operation. CMA CGM [15] list lifting costs of 25 € to 80 € for storing containers at various locations, but these might include multiple lifts in practice as the whole lifting process for storing is charged. For our instances, we use costs of $c_j^L = 27$ € for a single lifting operation at every terminal $j \in J$.

Storage and lifting capacities Schönemann [58, p. 77] states that the lifting time per container usually lies between three and four minutes, and is 3.375 minutes on average. This results in lifting capacities of about four containers per 15 minutes and per lifting device (crane, etc.) (=16 containers/h). Furthermore, www.intermodal-terminals.eu [39] lists various terminal/depot storage capacities, as well as the number of lifting devices, for a number of terminals (see Table 2 in the appendix).

However, we cannot directly use these values for our instances, as we currently only consider a fraction of all operating trains. In practice, there might be other

¹ $t_{\max} = \text{number of days} \cdot 24 \cdot \frac{60}{15}$
² $\frac{10 \cdot 15}{24 \cdot 60} \simeq 0.1042$, where 10 € is the price per day, 15 minutes is the size of a time unit, and $24 \cdot 60$ are the number of minutes per day.

trains (or even ships) that will also consume space and lifting capacities. Thus, we lack realistic storage (and lifting) capacity values like the average utilization of the considered resources, or the number of other trains present at a terminal.

For our instances, we assume that the largest terminal (Hamburg) is able to un- and reload all containers of a single train within one hour. Assuming 70 containers per train, we get a lifting capacity of 140 containers per hour, resulting in 35 container per time unit. We further assume that the number of lifts is directly proportional to the size of the terminal, resulting in lifting capacities of:

$$q_j^t = \lfloor \text{size}(j) \cdot 35 \rfloor$$

lifts per time unit for terminal j ($\forall t \in T$).

For the storage capacity, we assume that intermediate containers have to be stored to some extent at each depot, and for the largest terminal Hamburg, we assume that it can store the load of ten trains (700 containers). Again, we assume a direct correlation between the terminal's size and the space of its depot, resulting in a capacity of

$$\dot{C}_j^t = \text{size}(j) \cdot 700$$

containers for terminal j ($\forall t \in T$).

4.2.2 Train Generation

For generating train lines in between terminals, we follow the idea that was used during the construction of our rail network: We connect the terminals in such a way that all pc -pairs are connected by a train line. Trains stop and serve terminals on the way to replace redundant coverage of connections. For example, given train line A-B-C (where A is our port terminal), we do not generate an extra train line A-B, as it is already covered by the connection A-B-C. We always choose the port p as the train's **home terminal** when considering any pc -pair.

Transport capacity According to Allianz pro Schiene [3], transcontinental trains across Europe have at most 35 wagons and a maximum length of about 740 meters, which also matches the maximum length of trains in Austria. This leads to a maximum capacity of about 70-80 20' containers per block train, assuming two containers per wagon. In contrast, Schönknecht [57, p. 5] state a maximum capacity of 100 containers for a 700 m train.

UNECE [62, p. 2f] mention two types of typical block trains used in the Euro-Asian transport links: The first one being trains consisting of 39 wagons that can carry three 20' containers each (117 containers in total), and the second ones consisting of 57 wagons with a respective capacity of two 20' containers (114 containers in total).

Nevertheless, only eleven percent of trains reach a length of over 700 m (w.r.t. Deutsche Bahn), and over 60 percent measure less then 600 m, see [3].

In our benchmark instances, we use a transport capacity of $\ddot{C}_l^t = 70$ for all trains $l \in L$.

Weight capacity We assume that our trains are operated with a Siemens ES64U2 1116 Taurus locomotive and 35 2-TEU container wagons of type Lgjnss[33]. The locomotive is able to carry a total weight of about 1600 t (track dependent). Subtracting the wagon weight (13.5 t per 2-TEU wagon) yields a rounded total weight capacity of $\ddot{W}_l^t = 1127.5$ t for trains $l \in L$.

Transport costs Note that in this paragraph, we consider the operational costs of our freight forwarding company for shipping a container, and *not* the shipping costs that the customer is billed for (corresponding to the container's revenue, see later).

Black et al. [10] state that the shipping costs for a 20' container on their investigated routes lie between 0.32 € and 0.68 €. This results in an (unweighted) average of 0.45 € per kilometer [10, p. 40ff, 82]. The transport costs of 0.03 € per tonne-kilometer, provided by our project partners, agree with this³. Rounding the number to three decimal places, we get transport costs $\ddot{c}_l^t = 0.225$ € for a single time unit $t \in T$ and trains $l \in L$ ⁴.

Train speed The expected train speed is essential for calculating estimated shipping times of containers, and also for determining individual round trip times. According to European Court of Auditors [23], block trains travel with an average speed of 20 to 30 km/h. In contrast, Janic [40, p. 41] uses an average speed of 40 km/h, and an average anticipated delay of 0.5 h. We assume an average train speed of 30 km/h in all our instances.

Construction of Round Trips

Each round trip is constructed in a way that trains (of the same connection) leave their home terminal at the same time on their departure days (when they are not underway). To obtain this, we first construct the first round trip with the respective dwell times of each stop on the way. Then, we equally lengthen the dwell times of the first, the last, and an intermediate stop (stop C for round trip A-B-C-B-A) such that the (arrival and) departure of the first stop takes place at the same time of day then the (arrival and) departure of the last stop. This first round trip is then repeated multiple times until no further round trip can fit in the considered time horizon.

Note that we did not yet specify the exact starting times of the round trips, we only ensured that a train leaves its home terminal (and all other terminals) at the same time of day in the beginning of its round trips. The starting time of the first round trip (i.e., the arrival time at the first stop) is randomly chosen from such a time window that it can take place without the (departure) time at the last stop of its last round trip exceeding

³Assuming an average container weight of 14.3 t: $14.3 \cdot 0.03 \simeq 0.43 \text{ €} \simeq 0.45 \text{ €}$.

⁴Price per tonne-kilometer $0.03 \text{ €} \cdot \text{average train speed } 30 \text{ km/h} \cdot 0.25 (= \frac{15}{60}, 4 \text{ time units per hour})$

t_{\max} . All subsequent round trips are shifted accordingly to match the corresponding times of the first round trip.

For example, assume we have the following round trip after ensuring that it leaves its home terminal at the same time of day: The first stop has its arrival time on Monday 00:00, and its departure time on Monday 01:00. Its last stop has an arrival time on Wednesday 00:00, and a departure at 01:00. Recall that this stop is also the first stop of the second round trip. For simplicity, we consider a time horizon of a single week, i.e., until the consecutive Monday 00:00. This means that we will have three round trips, (so far) scheduled for Monday, Wednesday and Friday. As each round trip lasts two days, the last (=third) round trip has a last stop with arrival time on Sunday 00:00, and its departure time on Sunday 01:00. Note that exactly 23 hours remain until the end of our considered time horizon. This means that the initial starting time of the first round trip (=arrival time of the first stop of the first round trip) is arbitrarily chosen out of Monday 00:00 and Monday 23:00, and the dwell times of all stops of all round trip times are adjusted accordingly. For example, when Monday 10:00 is taken as the initial arrival time of the first round trip, the second round trip has its arrival time on Wednesday 10:00, and the third one on Friday 10:00. The departure time of the last stop of the last round trip is on Sunday 11:00.

Round trip times Shi and Chen [59, p. 71] describes the daily schedule of a shuttle train in between Göteborg and Karlstad (Sweden). The arrival at Göteborg is at 20:00, the departure at 01:50, the arrival at Karlstad is at 08:00, and the departure at 14:00.

Following this example, we chose the dwell times of our trains to be at most 6 hours, as we expect that the whole train can be un- and reloaded within this time. Additionally, we assume a correlation between terminal sizes and the dwell time of trains at that terminals, as potentially more container lifts are expected at larger terminals due to the increased demand. Therefore, we use the following formula to calculate the dwell times of train l at halting terminal j_p^{ld} (of stop π_p^{ld}):

$$B_p^{ld} - A_p^{ld} = \left\lceil \max\{4, \text{size}(j_p^{ld}) \cdot 24\} \right\rceil$$

time units. This corresponds to a maximum dwell time of 6 hours, and a minimum of 1 hour⁵, depending on the size of the halting terminal.

To determine the traveling time of trains in between stops π_p^{ld} , π_{p+1}^{ld} , we use the (track) distance in between the halting terminals and the average train speed from above, yielding⁶

$$A_{p+1}^{ld} - B_p^{ld} = \left\lceil \frac{\text{distance}(j_p^{ld}, j_{p+1}^{ld})}{30} \cdot 4 \right\rceil$$

time units.

⁵Recall time unit $\mu = 15$ minutes, hence 1 hour = $\frac{60}{15} = 4$ time units, and 6 hours = 24 time units.

⁶Average train speed: 30 km/h, number of time units per hour: $4 = \frac{60}{15}$.

Round trip costs Beresford [9, p. 240] states rail operating costs of £6.50 per train kilometer for a trailing weight of 1150 tonnes. This essentially supports the information provided by our project partners: Given the accumulated traveling distance of a round trip R_d^l of train $l \in L$, we chose the round trip costs \bar{c}_d^l to be 7 € per train kilometer, formally $\bar{c}_d^l = 7 \cdot \text{distance}(R_d^l)$.

Train Scheduling Method (Number of Trains per Train Line)

Given the train schedule for a round trip of a single train, we have specified its starting times, but we have not yet decided whether there are other trains traveling along the same train connection on different days of the week. We investigate two variants of *train scheduling methods* that are applied to all trains of a single instance:

1. We generate just one train per train connection, called “**single-train** variant”. (We call instances with this train scheduling method “*single-train* instances”.)
2. We generate as many trains as required for having one train leave its home terminal every day at the same time⁷, called “**daily-train** variant” (or “*daily-train* instances”).

For example, if the first train has its round trip starts on Monday 10:00 and is repeated on the fourth day (Thursday 10:00), two additional trains are generated. The second train has its first round trip on Tuesday 10:00, and the third one on Wednesday 10:00.

Note the possibility of these two variants being the same for a train connection if its round trips repeat after a single day. Nevertheless, this is not the case if round trips last more than one day.

4.2.3 Container Parameters

Finally, we place $|I|$ containers with the following parameters at our terminals:

Origin and destination terminal For the placement of containers, we assume a direct correlation to the size of terminals. Given the terminal sets P and C , we assume that we do not want to ship containers from port to port, as these are potentially covered by maritime transportation. We choose the origin terminal of container $i \in I$ from a weighted distribution \mathcal{W} based on the size of the terminals⁸, i.e., $o_i \sim \mathcal{W}(P \cup C)$. If the origin terminal is a port, we choose the destination terminal

⁷after the day when the first train leaves its home terminal for the first time with the respective scheduled times as mentioned before

⁸For weighted distribution $\mathcal{W}(\mathcal{J})$, the probability that each terminal $j \in \mathcal{J}$ is chosen as the origin terminal o_i is calculated by $P(o_i = j) = \frac{\text{size}(j)}{\sum_{j' \in \mathcal{J}} \text{size}(j')}$.

out of the cities, otherwise we (again) choose the destination terminal out of all terminals $J \setminus \{o_i\}$, formally:

$$d_i \sim \begin{cases} \mathcal{W}(C), & \text{if } o_i \in P \\ \mathcal{W}(C \cup P \setminus \{o_i\}), & \text{otw.} \end{cases}$$

Container time windows — release times and deadlines For choosing release time and deadline of a container $i \in I$, we first optimistically calculate the expected shipping time in time units w.r.t. the minimum distance $\text{distance}(o_i, d_i)$, and add $2 \cdot 6$ hours as an estimated un-/loading time (including buffer for lateness, see Section 5.4.4) of the train. This yields the expected shipping time E_i of container i , formally:⁹

$$E_i = \left\lceil \left(\frac{\text{distance}(o_i, d_i)}{30} + 12 \right) \cdot 4 \right\rceil$$

Based on E_i , we now calculate release time and deadline, respecting an additional flexibility *buffer time window* buff_i , given in the number of days, that is given as input to our instance generation process.

We choose the release time R_i of each container $i \in I$ randomly out of the considered time horizon s.t. the deadline still lies within the considered time horizon. Formally, where \mathcal{U} is the uniform distribution¹⁰:

$$R_i \sim \mathcal{U}(\{1\} \cup \{t \in T \mid t < t_{\max} - (E_i + \text{buff}_i \cdot 96)\})$$

The deadline D_i then is calculated by:

$$D_i = \min\{t_{\max}, R_i + E_i + \text{buff}_i \cdot 96\}$$

Container weight According to Janic [40], European Commission [22] states the average gross weight of a 20' containers is 14.3 metric tonnes [40, p. 40]. UNECE [62, p. 3] state that they rarely exceed an overall mass of 18 tonnes (2.5 tonnes tare plus 15.5 tonnes load). According to the freight forwarder company iContainers [37], an empty containers weights 2.3 tonnes and can be theoretically loaded with goods weighting at most 25.4 tonnes. For simplicity, we use a normal distribution with a mean of 14.3 and a standard deviation of 4 to randomly sample each container weight, while respecting permitted minimum and maximum loads:

$$w_i = \max\{2.3, \min\{27.7, \mathcal{N}(14.3, 4)\}\}$$

tonnes.

⁹The average train speed is 30 km/h, $2 \cdot 6 = 12$ h is the dwell time at the origin and destination terminal, and $\frac{60}{15} = 4$ is the number of time units per hour.

¹⁰ $24 \cdot \frac{60}{15} = 96$ is the number time units per day. The set $\{1\}$ is required because if the considered time horizon is smaller than the time windows of the container, the right hand side is empty, hence we pick 1 as the release time and set the deadline to t_{\max} .

Expected shipping distance and container revenue Note that the shipping price (= revenue) of a container has to be determined in advance to reliably charge the customer, although the actual shipping cost for the freight forwarder company is dependent on the actual shipping path. Therefore, the expected shipping price (including reward) is based on the expected traveling distance.

www.mtcontainer.de/container-service/transport/ [16] lists 1.25 € per container kilometer. Expecting an average container weight of 14.3 t, the use of 0.09 €¹¹ per tonne-kilometer seems appropriate for our project partners. To investigate different pricing strategies, we include additional *revenue factors* $\rho \in \{0.5, 1, 1.5, 2.5\}$. For the container revenues, we get the following formula:

$$r_i = \rho \cdot \text{distance}(o_i, d_i) \cdot w_i \cdot 0.09 \text{ €}$$

where $\text{distance}(o_i, d_i)$ is the minimum distance that container i has to travel in the underlying rail network from its origin to its destination terminal.

4.3 Experimental Setup

Generated instances For benchmarking, we generated 840 instances. We parameterized them by the cross-product of the following parameters: The number of containers $|I| \in \{1000, 2000, 3000, 4000, 5000\}$, the considered time horizon of $t_{\max} = 96 \cdot h$, where $h \in \{14, 21, 28, 35\}$ is the number of considered days¹², and container buffer time windows $\text{buff}_i \in \{3, 7, 14\}$,

The number of terminals, distinguished by the number of ports $|P|$ and the number of cities $|C|$, were chosen from $(|P|, |C|) \in \{(1, 2), (1, 3), (2, 3), (3, 3), (2, 4), (2, 5), (3, 4)\}$ (Table 3 in the appendix contains their respective rail networks), and we used both train scheduling methods (w.r.t. all trains of a single instance), i.e., the single-train and the daily-train variant (see Section 4.2.2). For all instances, we chose the neutral revenue factor of $\rho = 1$.

Used infrastructure and configuration All instances were executed on a single core of an Intel® Xeon® E5-2640 v4 with 2.40 GHz, and a time limit of 5 hours. We used CPLEX 20.1 as our ILP solver.

¹¹ $\frac{1.25\text{€}}{14.3\text{t}} \simeq 0.09 \text{ € per tonne-kilometer}$

¹² $96 = 24 \cdot 4$ is the number of time units per day ($\mu = 15$ minutes).

CHAPTER 5

Results

In this chapter, we investigate the results of our created and benchmarked instances from the previous section.

First, we describe the features (attributes/properties) we used in the investigations of our instances. Then, we consider the solution of a single exemplary instance in order to better understand the overall solution structure and characteristics. Next, the performance of our algorithm and various solution characteristics of all instances are analyzed together with the effects of different instance parameters on the respective solutions. Finally, we summarize these results and draw conclusions that are relevant for practical implementations of such systems.

5.1 Performance and Solution Characteristics

For investigating the performance of our instances, we consider the following features: The number of optimally solved instances, the average optimality gap (between our best solution and the dual bound, not our solution and the optimal solution), and the running times of the instances. Beside these performance characteristics, we focus our analysis on the following solution characteristics:

The **deliverable container rate** is important to identify whether all containers of an instance can theoretically be delivered. Sometimes, train schedules and narrow container time windows of instances can lead to containers that can never be transported with respect to the given constraints. In these cases, the underlying assumptions concerning train schedules (too long round trips, too few trains) and (too short) container time windows might be improvable.

The deliverable container rate is calculated by dividing the number of *deliverable* containers by the total number of containers, formally:

$$\text{deliverable container rate} = \frac{\text{number of deliverable containers}}{|I|}$$

A container is *deliverable* if it can theoretically be shipped w.r.t. its release time and deadline and our operated trains. This is the case if there is at least one path from its origin to its destination vertex in the underlying time-expanded network.

Note that usually all containers can be delivered (are *deliverable*), i.e., the deliverable container rate is 1. Nevertheless, short container time windows and given train schedules (especially when using the single-train variant as the train scheduling method, see Section 5.3.1) can lead to a few containers that cannot be shipped at all, resulting in a deliverable container rate strictly smaller than 1.

Further note that the deliverable container rate is actually the characteristic of an instance and not a solution.

The **(container) delivery rate** represents the rate of delivered containers in our instance. In general, we aim for higher delivery rate values as we want to have high container throughput through our rail network, even if we technically seek maximum revenue. A low value might indicate that the individual container revenues are too low, or that too few trains as scheduled to ship the considered containers.

The delivery rate is calculated by dividing the number of (actually) *delivered* containers by the number of *deliverable* containers, formally:

$$\text{delivery rate} = \frac{\sum_{i \in I} y_i}{\text{number of deliverable containers}}$$

Recall that the variable y_i is one if and only if a shipping path for container i has been found in the solution.

The **active round trip rate** indicates how many round trips in an instance are active, i.e., have their corresponding MIP-variable z_d^l set to one. If the active round trip rate is low, it indicates that many trains have non-active round trips, hence idle time. A reason for low values might be a low container density such that the round trips do not pay off. On the other hand, if the active round trip rate is almost one, it might make sense to investigate if scheduling more trains increases the throughput of the network, potentially resulting in higher container delivery rates and increased collected revenue.

The active round trip rate is calculated by dividing the number of active round trips by total number of round trips, formally:

$$\text{active round trip rate} = \frac{\sum_{l \in L} \sum_{d \in D^l} z_d^l}{\sum_{l \in L} |R^l|}$$

The **(average train) transport capacity utilization** represents the average load factor over all trains. It is the ratio between the average number of containers transported on a train and the total number of containers they can carry. i.e., their

transport capacity. If it is one, every single train carries the maximum amount of containers all the time (this is what we aim for), and if it is zero, all trains are travelling empty (the worst case). Therefore, this characteristic directly reflects the overall train utilization, and we aim for high values.

It is calculated as follows: For every single round trip of an instance, we calculate the transport capacity utilization whenever the train is moving, i.e., in between stops. Therefore, we consider all transport arcs $a \in \mathcal{A}(l)$ of train $l \in L$ where it is not halting, and which are part of an active round trip, formally:

$$\begin{aligned} \mathcal{A}(l) = & \{ (l^{t_1}, l^{t_2}) \in \bar{A}(l) \mid \neg \exists \pi_p^{ld} \in R_d^l : A_p^{ld} = t_1 \wedge B_p^{ld} = t_2 \} \\ & \cap \{ a \in \bar{A}_R(l, d) \mid d \in D^l : z_d^l = 1 \} \end{aligned}$$

Along each arc $a \in \mathcal{A}(l)$, we calculate its (transport) utilization:

$$\text{arc utilization}(a) = \frac{\sum_{i \in I(a)} x_a^i}{\bar{C}_a}$$

To get the utilization of a single train $l \in L$, we calculate the mean over all arcs $(l^{t_1}, l^{t_2}) = a \in \mathcal{A}$, weighted by their duration $\Delta_a = t_2 - t_1$, formally:

$$\text{single train utilization}(l) = \frac{\sum_{a \in \mathcal{A}(l)} \left(\Delta_a \cdot \frac{\sum_{i \in I(a)} x_a^i}{\bar{C}_a} \right)}{\sum_{a \in \mathcal{A}(l)} \Delta_a}$$

Finally, we take the mean value utilization value over all trains, formally:

$$\text{(overall) transport capacity utilization} = \frac{\sum_{l \in L} \frac{\sum_{a \in \mathcal{A}(l)} \left(\Delta_a \cdot \frac{\sum_{i \in I(a)} x_a^i}{\bar{C}_a} \right)}{\sum_{a \in \mathcal{A}(l)} \Delta_a}}{|L|}$$

The **weight capacity utilization** is the average utilization calculated similarly to the transport capacity utilization, but with respect to the total weight carried by the train relative to its weight capacity (in contrast to the transport capacity utilization, where the number of carried containers relative to the transport capacity is considered). The weight capacity along each arc $a \in \mathcal{A}(l)$ is calculated by:

$$\text{arc utilization}(a) = \frac{\sum_{i \in I(a)} x_a^i \cdot w_i}{\bar{W}_a}$$

Therefore, the weight capacity utilization for the whole instance is calculated as:

$$\text{weight capacity utilization} = \frac{\sum_{l \in L} \frac{\sum_{a \in \mathcal{A}(l)} \left(\Delta_a \cdot \frac{\sum_{i \in I(a)} x_a^i \cdot w_i}{W_a} \right)}{\sum_{a \in \mathcal{A}(l)} \Delta_a}}{|L|}$$

The weight capacity utilization strongly depends on the transport capacity utilization (and the individual container weights). This is due to the fact that container weights are chosen with the same distribution for all instances. As the transport capacity is usually more constraining than the weight capacity for our instances, we primarily investigate the transport capacity utilization and not the weight capacity utilization in the remainder of this chapter. In general, most observations regarding the transport capacity utilization can be directly applied to the train weight utilization as well.

The **(average) storage capacity utilization** represents the average load factor of all terminals over time and indicates whether the storage of terminals is heavily used.

It is calculated by summing up the number of containers stored in any terminal at each time point, and dividing it by its total storage capacity summed up over our considered time horizon. The values for each individual terminal are then averaged, resulting in a single value per instance, the *storage capacity utilization*. Formally, where $\mathbf{a}(t, j) = (j^{t_1}, j^{t_2}) \in \hat{A} : t \in [t_1, t_2]$ is the storage arc of terminal j at time point t :

$$\text{storage capacity utilization} = \frac{\sum_{j \in J} \frac{\sum_{t \in T} \sum_{i \in I(\mathbf{a}(t, j))} x_{\mathbf{a}(t, j)}^i}{\sum_{t \in T} \hat{C}_j^t}}{|J|}$$

Note that when arcs are longer than one time unit, the containers stored during that time at the corresponding terminal are counted multiple times in the sum, as we consider every single time unit. At the same time, we also sum up all storage capacities at all time points in the denominator of that sum.

Similarly, the **(average) lifting capacity utilization** represents the average number of lifting operations over all terminals and the considered time horizon.

It is calculated by computing the lifting utilization of each terminal at each time point. The number of performed lifting operations at a terminal is summed up, divided by the total lifting capacity over the considered time horizon. The values of each terminal are then again averaged, resulting in a single value per instance, formally:

$$\text{lifting capacity utilization} = \frac{\sum_{j \in J} \frac{\sum_{t \in T(j)} \sum_{a \in \hat{A}(j, t)} \sum_{i \in I(a)} x_a^i}{\sum_{t \in T} q_j^t}}{|J|}$$

Note that the lifting capacity utilization values for our solutions are usually quite small (<0.1) due to the fact that lifts only occur when trains halt at the terminals.

The **required train rate** is the last and most complex of our solution characteristics. From a practical point of view, we are interested in the advantages of operating trains in the way that we propose in this study, compared to the state of the art. Beside considering the transport capacity utilization, we can investigate whether we can reduce the number of required trains when transporting containers with our method.

Given a solution of an instances with its delivered containers, we ask the following: How many direct trains (of the same size) would be necessary for transporting the delivered containers in this solution?

For pairs of terminals $(j_1, j_2) \in J \times J$ (origin and destination), we have to minimize the number of trains required for transporting all containers $i \in I$ with $y_i = 1$ (i.e., those being delivered in our solution) that have $o_i = j_1$ (their origin terminal is j_1), and $d_i = j_2$ (their destination terminal is j_2). We further have to respect the time constraints induced by the time windows of these containers, and schedule trains based on that.

Note that this is a non-trivial optimization problem on it's own. Instead of solving it, we can easily compute the lower bound of trains that is necessary for transporting them, assuming that these direct trains have the same transport capacity of $C = 70$ as the trains in our instances.

$$\text{lower bound of direct trains} = \sum_{(j_1, j_2) \in J \times J} \left\lceil \frac{\sum_{i \in I: o_i=j_1 \wedge d_i=j_2} y_i}{C} \right\rceil$$

The inner sum counts the number of containers that have to be transported for each terminal-pair j_1, j_2 and were delivered in our solution. This number is then divided by the transport capacity. As trains that are just half fully loaded still count as a whole train, we round this number up and sum over all pairs of terminals, yielding the *lower bound of direct trains*. To make this number comparable to our instances, we have to consider that our round trips are operated both ways. For this calculation, an active round trip therefore counts as two trains.

We can now compute the *required train rate*. It is calculated by the number of trains that are used in the solution of our method (multi-stop round trips), i.e., the number of active round trips ($z_d^l = 1$) times two (for the two round trip legs), divided by the lower bound of direct trains, formally:

$$\text{required train rate} = \frac{2 \cdot \sum_{l \in L} \sum_{d \in D^l} z_d^l}{\text{lower bound of direct trains}}$$

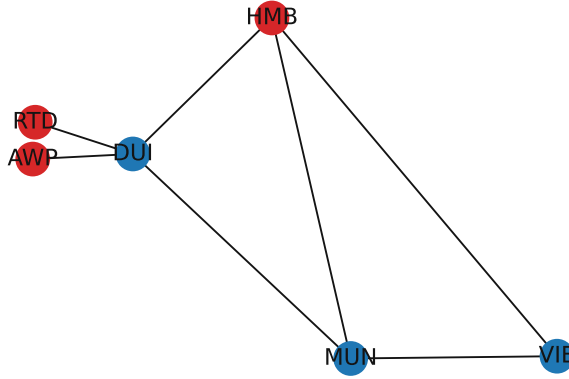


Figure 5.1: Rail network of our exemplary instance with three ports and three cities.

A required train rate value larger than one means that our solution requires more trains, when comparing it to lower bound of direct trains. Therefore, we aim for low required train rates, as these suggest that we can indeed perform better when using trains operated as proposed in our study, i.e., having multi-stop round trips.

5.2 Investigating an Exemplary Solution

To get an overall impression of the instances and their solution characteristics, we investigate the solution of a single exemplary instance in detail before our analyses of the entire pool of solutions.

For this purpose, we look at a “median-sized” instance. i.e., we consider the instance where all instance parameters are chosen from the (lesser) median of our benchmark configuration. Therefore, the instance of the exemplary solution under investigation has following parameters: Its rail network, shown in Fig. 5.1, has three ports and three cities, i.e., $(|P|, |C|) = (3, 3)$. The number of containers $|I|$ is 3000, the considered time horizon is three weeks, i.e., $h = 21$ (and $t_{\max} = 672 \cdot 3$), the container buffer time windows buff_i is 7 days, and the train scheduling method is the single-train variant, i.e., we have a single train per connection. Following the train scheduling method from Section 4.2.2, we get five trains (see Section 5.2.2 below).

This instance was solved to proven optimality within 9336 seconds¹. The *deliverable container rate* of this instance is one, i.e., all containers are (potentially) deliverable. In the optimal solution, 2169 out of the 3000 were indeed delivered, resulting in a *delivery rate* of 0.723.

When considering the overall number of containers that are present in the time-expanded network at each time point, we can observe that in the beginning and the end of the considered time horizon relatively fewer containers are present. This is illustrated in Fig. 5.2. The (available) container density is highest during the middle of our considered time horizon. This is due to the fact that in the beginning and the end, fewer containers

¹9335.163 s = 2 h 35 min 36.163 s

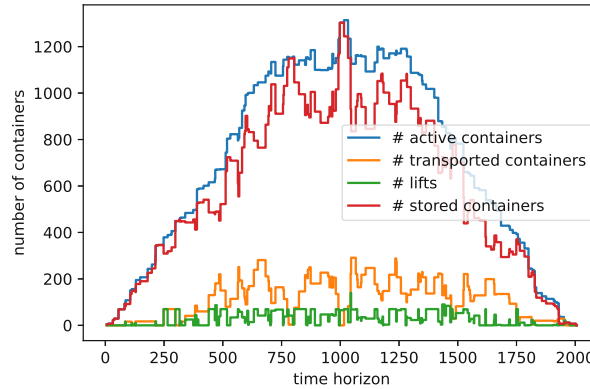


Figure 5.2: Number of containers being active, transported, lifted and stored at each point in time for our exemplary solution.

are available, as their release times and deadlines are chosen uniformly at random from the available time horizon (see Section 4.2.3). With the number of containers present in the network, the number of active containers, the number of actively transported containers, the number of lifting operations, and the number of stored containers similarly increase and decrease, as can be seen in Fig. 5.2. These observations hold for all solutions of our instances, not just this one.

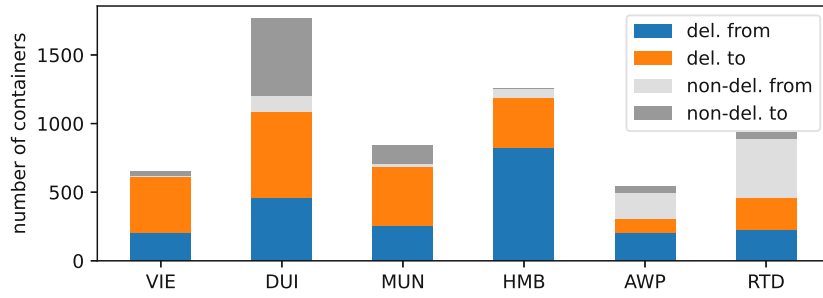
5.2.1 Investigating Terminals

At first, we investigate the six terminals of our instance and consider the number of containers that are deliverable or delivered from and to these terminals. Figure 5.3a shows the number of delivered and non-delivered containers from and to each of them.

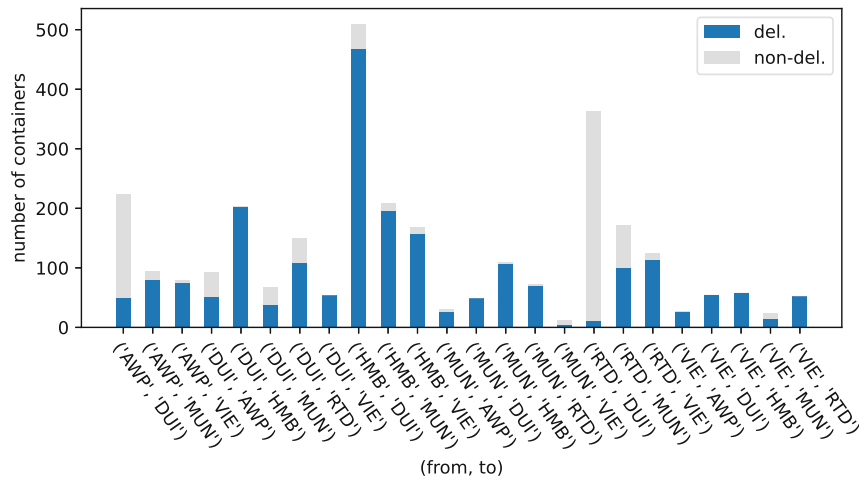
Because container origins and destinations are chosen as described in Section 4.2.3, it is not surprising that the number of deliverable containers from and to each terminal depends on its size, and whether it is a port or a city. Compared to all other terminals, DUI is the origin or destination terminal of the largest number of containers, followed by HMB. HMB is the largest terminal, but as it is a port, DUI is preferred as a destination terminal during the placement of containers, as it is a city terminal. Therefore, most containers (potentially) originate from HMB ($o_i = \text{HMB}$), followed by RTD, while DUI (as the largest city terminal) is the destination of the largest number of containers ($d_i = \text{DUI}$).

When we consider the actually delivered containers, HMB and DUI have the most delivered containers both from and to them. Almost all containers that should have been delivered from and to VIE could actually be delivered in our solution. In contrast, RTD has the most non-delivered containers originating from it, and DUI the most non-delivered containers that should have been delivered to it.

When looking at Fig. 5.3b, we gain further insight: The plot shows the number



(a) Number of (non-)delivered containers per terminal



(b) Number of (non-)delivered containers per terminal pair

Figure 5.3: Number of delivered and non-delivered containers per terminal (and terminal pair) of our exemplary solution.

of delivered containers (out of the deliverable containers) per terminal pair, i.e., the containers that are transported from j_1 to j_2 for terminal pair (j_1, j_2) . We can see that for most pairs almost all containers were delivered, with two noticeable exceptions: Almost no containers from either AWP or RTD could be delivered to DUI. This suggests that it might make sense to introduce additional train lines to cover the shipment of these containers.

When looking at the number of stored containers at each terminal at each point in time, shown in Fig. 5.4, we can again see that in the beginning and the end of our considered time horizon fewer containers are stored at the terminals. Another noticeable fact is that the y -values of the graph, representing the number of transported containers of a train, change only at terminal action time points, i.e., when a train is arriving at or departing from that terminal. At these time points, containers can be lifted from or to the terminal storage, or are initially placed or picked up, resulting in changes of values.

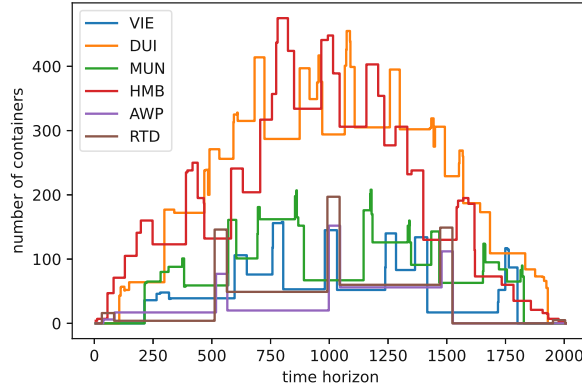


Figure 5.4: Number of stored containers at each terminal in our exemplary solution, plotted over time.

Table 5.1: Average storage and lifting capacity utilization values for all terminals of our exemplary instance.

	VIE	DUI	MUN	HMB	AWP	RTD	Average
Storage							
cap. util.	0.294	0.459	0.365	0.272	0.092	0.068	0.258
Lifting							
cap. util.	0.038	0.029	0.036	0.02	0.01	0.008	0.024

Recall that if we sum up the number of containers present at the terminal at each time point and divide it by the summed-up storage capacities over all time points, we get the storage capacity utilization for a single terminal. Table 5.1 shows the storage (and lifting) capacity utilization values of all individual terminals, as well as their averages over all terminals. In the case of DUI, its storage capacity utilization values is 0.459, meaning that on average about half of its storage capacity is used. The solution’s overall *storage capacity utilization* (i.e., the average utilization of all terminals) is 0.258. This means that on average, the terminals use about a quarter of their available storage.

Similarly, we can calculate the lifting capacity utilization of the individual terminals, as shown (again) in Table 5.1. Averaging the values of the individual terminals yields the solution’s overall *lifting capacity utilization* of 0.024. Note that compared to the storage capacity utilization values, the lifting capacity utilization values are small, as lifting operations only happen when a train halts, so most of the time, no lifting takes place.

Table 5.2: Number of (active) round trips of our exemplary solution.

	HMB -VIE	HMB -DUI	HMB -MUN	AWP -VIE	RTD -VIE	All
Number of active round trips	4	7	3	3	4	21
Total number of round trips	5	10	6	4	4	29
Ratio	0.8	0.7	0.5	0.75	1.0	0.724

5.2.2 Investigating Trains and Round Trips

Our instance has five trains with the following stops on their round trips:

- (1) HMB-VIE (visiting HMB-VIE-HMB),
- (2) HMB-DUI (visiting HMB-DUI-HMB),
- (3) HMB-MUN (visiting HMB-MUN-HMB),
- (4) AWP-VIE (visiting AWP-DUI-MUN-VIE-MUN-DUI-AWP), and
- (5) RTD-VIE (visiting RTD-DUI-MUN-VIE-MUN-DUI-RTD).

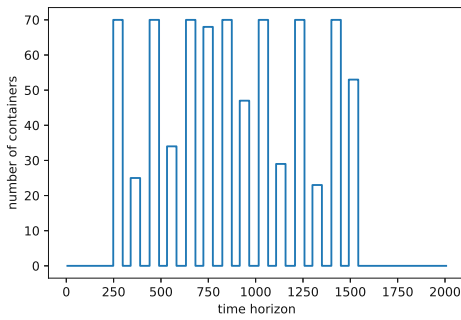
The round trips of train HMB-DUI have the shortest duration, yielding a total of ten round trips on that connection, while trains AWP-VIE and RTD-VIE only have four round trips each, as these individual round trips take longer. Train HMB-VIE has five, and HMB-MUN has six round trips, yielding 29 round trips altogether. Out of these 29 round trips, 21 are active, resulting in an *active round trip rate* of 0.724. Table 5.2 shows the number of active round trips of each terminal, and the resulting ratios when dividing the number of active round trips by the number of all potential round trips. We can see that all round trips of train RTD-VIE take place, whereas only half of the potential round trips of train HMB-MUN are active.

In order to understand the generation of the round trip times from Section 4.2.2 better, Table 5.3 shows the dwell times of each stop of the first round trip of train RTD-VIE, which repeats after five days.

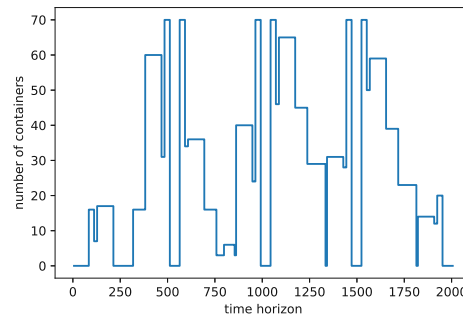
In order to calculate the utilization of trains, we firstly consider the *transport capacity utilization* (see Section 5.1). Recall that this solution characteristic is calculated by taking the average transport utilization of all sections (arcs) of all active round trips of the train where it is not halting, weighted by the travelling time of that section. Train RTD-VIE has the lowest transport capacity utilization with a value of 0.527, while train HMB-DUI has the highest one being 0.785. When taking the mean of all average transport utilization values w.r.t. all trains, we get an (overall) *transport capacity utilization* of 0.659 for this instance.

Table 5.3: Round trip times of train RTD-VIE of our exemplary instance.

Terminal	Arrival	arr. time unit	Departure	dep. time unit
RTD	Mon 08:00	32	Mon 21:00	84
DUI	Tue 04:00	112	Tue 08:00	128
MUN	Wed 05:30	214	Wed 07:30	222
VIE	Wed 21:30	278	Thu 07:30	318
MUN	Thu 21:30	374	Thu 23:30	382
DUI	Fri 21:00	468	Sat 01:00	484
RTD	Sat 08:00	512	Sat 21:00	564



(a) HMB-DUI



(b) RTD-VIE

Figure 5.5: Number of transported containers for two specific trains in our exemplary solution, plotted over time.

Figure 5.5 shows the number of transported containers for the two trains HMB-DUI and RTD-VIE over time. As the round trips of train HMB-DUI have just two stops (plus the return stop), all shipped containers are unloaded at every stop. Therefore, the number of containers drops to zero whenever the train is halting, as containers are lifted off the train at its arrival, and lifted onto the train at its departure. This leads to an even-looking, almost bar-plot-like line for this train. In contrast, the round trips of train RTD-VIE have more stops. When it halts, containers often remain on the train, as their target stop has not yet been reached. This leads to a more irregular looking plot, indicating that also at intermediate stops container are lifted on and off the train.

5.2.3 Calculating the Required Train Rate

When asking for the lower bound of direct trains, we consider the number of delivered container for pairs of terminals. For example, 468 containers were delivered from HMB to DUI. For transporting this number of containers, we need (at least) seven trains² with

² $468/70 \simeq 6.69$

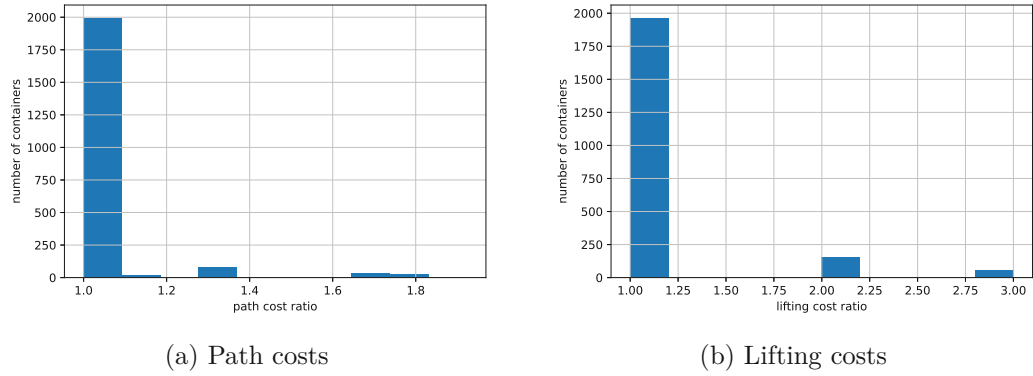


Figure 5.6: Relative path and lifting cost histograms for the delivered containers in our exemplary solution.

a capacity of 70 containers. When summing up the calculated values for all pairs of terminals, we get a *lower bound of direct trains* of 42 for this solution. As the number of active round trips is 21, this instance has a *required train rate* of exactly one³. This means that when comparing the number of active round trips (time two) with the lower bound of direct trains, we cannot see any gain (or loss) in efficiency for this instance with respect to this value. In other words, we need the same number of trains as when having directly scheduled trains (ignoring time window constraints).

5.2.4 Investigating Container Paths

When considering the paths of all 2169 delivered containers, we find that 1927 containers travel on their (individual) cost-optimal path. In contrast, 242 containers are shipped on paths that exceed the cost of their minimum shipping cost path by more than one percent. Out of these containers, 205 were lifted more often than would be necessary on their cost-optimal path. Figure 5.6 shows this. The values for each container in Fig. 5.6a are calculated by dividing the actual path costs of each container by their minimum path costs. The same holds for Fig. 5.6b, except that only lifting costs are considered, not the overall path costs of each container. We can observe that the relative lifting path costs in Fig. 5.6b are either 1, 2 or 3, i.e., all integer values. This is due to the fact that the lifting costs of all terminals are equal, and the number of required lifts is always an integer multiple of the minimum number of required lifts for this instance.

Note that this must not always be the case. In other instances, also fractional values are possible. The lifting costs are still equal for all terminals, but there might exist a container having a minimum number of, e.g., four lifts, but the actual number of lifts is six, resulting in a relative lifting cost value of 1.5.

We now consider the paths of two specific containers: container i_{2850} , which has the highest relative path cost, and container i_{1951} , which has the highest relative lifting cost (and among these containers, the highest relative path cost).

³ $(21 \cdot 2)/42 = 1$

Container i_{2850} should be shipped from DUI to MUN, but as the lifting capacity is exceeded when the train that transports it halts at MUN, it is shipped onward to VIE. There, it remains on the train and is shipped back to MUN, where it finally is unloaded. This results in 1.923 times shipping costs when compared to than would be required on its minimum cost path. Note that the customer of that container would not be charged more for this, as we are optimizing from the point of view of the freight forwarder. Instead, the price paid by the customer is the fixed revenue of the container.

The other container, i_{1951} , has the highest relative lifting costs. It should be shipped from MUN to AWP. Instead of being directly shipped by train AWP-VIE to its destination, it takes a detour via HMB as the transportation capacity of train AWP-VIE is already fully utilized. It is first shipped from MUN to HMB by the train HMB-MUN, then to DUI by the train HMB-DUI, and finally to AWP by train AWP-VIE. This detour results in thrice the number of required lifts, and overall higher shipping costs of 1.654 times that of its minimum-cost shipping path.

Note that our solution has been solved to proven optimality, hence the sub-optimal paths of the containers are only non-optimal w.r.t. to their own minimum shipping costs. Nevertheless, it overall pays off to send them on these individually suboptimal paths as we can, e.g., use fewer round trips to save costs.

5.3 Investigation of Results

After we have seen a single solution in detail, we now focus on the solution characteristics from Section 5.1 to analyse the whole set of benchmark solutions. In this section, we present a few general results of our benchmark, and then discuss the effects of instance parameters on the solution characteristics. Before that, we briefly investigate the deliverable container rate of our instances on its own.

5.3.1 Deliverable Container Rate

The deliverable container rate is one for most instances, but there do exist instances with values smaller than one. Figure 5.7 shows that a low deliverable container rate highly correlates with narrow container buffer time windows. The minimum proportion of deliverable containers is 0.75 for one instance with narrow container buffer time windows of $\text{buff}_i = 3$ days. When only considering instances with wider time windows ($\text{buff}_i \in \{7, 14\}$), we get a minimum deliverable container rate of 0.983, i.e., almost all containers can be delivered for all of these instances.

5.3.2 General Results and Performance w.r.t. Train Scheduling Methods

Out of the total of 840 benchmarked instance, 332 instances could be solved to optimality, and 751 instances yielded at least one feasible solution. A solutions counts as optimally solved when it reaches a gap smaller than 10^{-4} , which is the default relative MIP gap

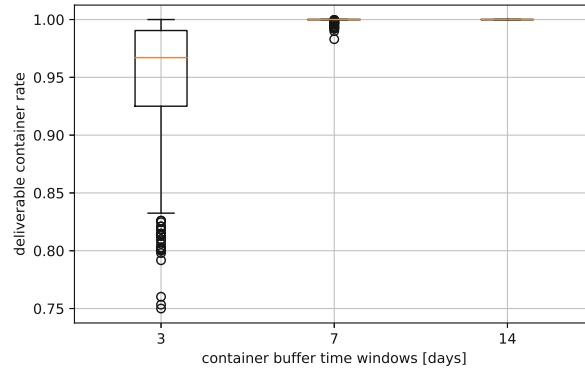


Figure 5.7: Boxplots of the deliverable container rate for all benchmarked instances, grouped by the size of the container buffer time windows $\text{buff}_i \in \{3, 7, 14\}$.

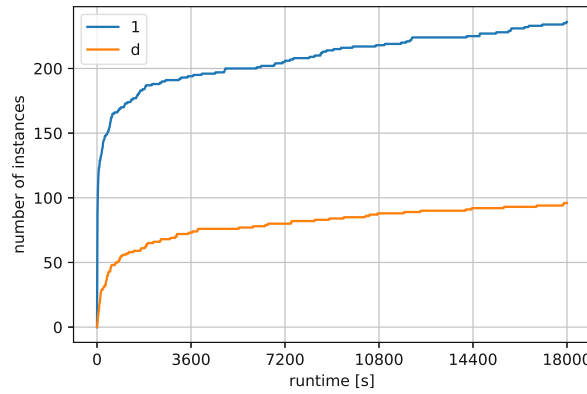


Figure 5.8: Performance plot of the runtimes of instances, grouped by their respective train scheduling method. “1” represents the single-train instances, while “d” represents the daily-train instances.

tolerance of CPLEX. The remaining 89 unsolved instances terminated due to exceeding the amount of available memory or could not find any feasible solution at all within the specified time limit.

When investigating the runtimes of our instances, we look at the performance plot of runtimes shown in Fig. 5.8. The x -axis represents the runtime, and the y -axis counts the number of instances that terminated within at most x seconds. The instances are grouped by their respective train scheduling method: “1” represents the single-train instances, and “d” is the daily-train instances. Recall that single-train instances have only a single train scheduled for each train connection. This train performs repeated round trips. In contrast, the daily-train instances have one train scheduled per day, resulting in potentially more than one train per connection (see Section 4.2.2).

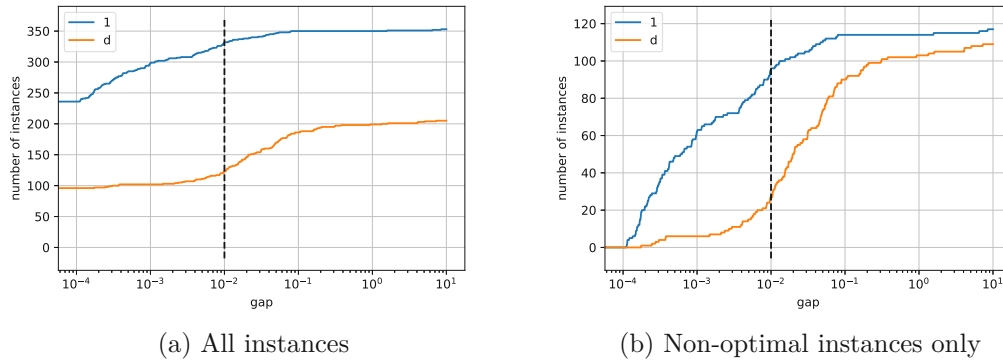


Figure 5.9: Performance plot of the gaps of our benchmark instances, grouped by train scheduling method. “1” represents the single-train instances, while “d” represents the daily-train instances. The dashed vertical line at $x = 0.01$ indicates the gap of instances that we consider in Section 5.3.3.

We can observe that the daily-train instances are much harder to solve than the single-train instances, as the availability of more trains leads to more routing options and larger models. Additionally, we can see that for both train scheduling methods, there are instances that terminate with proven optimality for almost all times within the specified time limit.

Out of the 420 single-train instances, 416 instances returned a solution while 4 instances did not yield a solution due to memory issues. Out of the 416 successful single-train instances, 180 reached the maximum time limit of 5 hours, while 236 instances were solved to proven optimality before that. Out of the 180 non-optimal solutions, 117 instances yielded a non-trivial solution, meaning that at least one container was delivered (in contrast to the trivial solution of setting all variables to zero), hence having a positive objective value.

In contrast, when considering the 420 daily-train instances, only 335 instances returned a feasible solution, and 85 instances terminated due to exceeding the amount of available memory or finding not even the trivial solution. Only 96 instances could be solved to proven optimality, and out of the 239 instances that reached the time limit, 111 instances yielded non-trivial solutions.

We now consider the gaps of instances with non-trivial solutions. Figure 5.9 shows the performance plot for the gaps of these solutions, again grouped by their train scheduling method. The x -axis represents the value of the gap, and the y -axis counts the number of solutions with a gap of at most x . Figure 5.9a contains solutions of all instances, while Fig. 5.9b only considers non-optimal instances.

We can see that the gaps of the single-train instances are smaller than the ones for the daily-train instances due to the increased hardness of the latter, as the line representing the single-train instances is steeper than the one for the daily-train instances. This is well-illustrated in Fig. 5.9b.

We can further observe that only a few instances have gap-values in between 0.1 and

10. This suggests that once a non-trivial solution is found, CPLEX can quickly narrow the gap significantly, but proving optimality is not easy.

5.3.3 Single-Train Instances – In-Depth Analysis of Solution Characteristics

When investigating the solution characteristics with respect to our instances parameters, we only consider instances that are close to or optimally solved. Otherwise, we could draw misleading conclusions from highly suboptimal solutions. We consider a gap of one percent as sufficiently meaningful for this purpose. In the following, we refer to instances with solutions having a gap smaller or equal to one percent as “instances with *sufficient gap*”. Note that also optimal solutions have a sufficiently small gap, hence are also contained in this set.

Single-train instances have more than twice the number of optimally solved instances in comparison to the daily-train instances. 332 single-train instances fulfill our aforementioned gap-criterion, while only 123 daily-train instances do so. Therefore, we only investigate the effect of instance parameters on the solution characteristics of single-train instances in detail in this (sub)section. Solutions of the daily-train instances are briefly investigated in that manner in the subsequent Section 5.3.4.

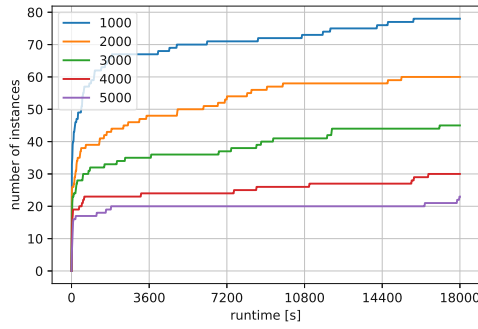
In order to examine the solutions, we group the single-train instances by different instance parameters and investigate their effects on the solution characteristics in the following.

Number of Containers

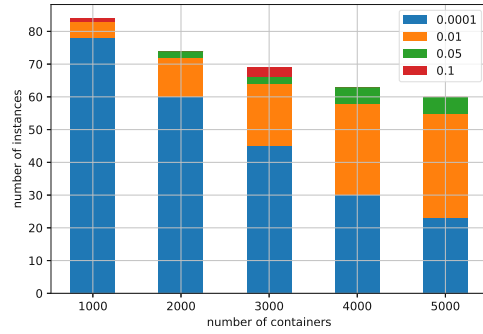
The first instance parameter under investigation is the number of containers $|I|$. When looking at the hardness of our instances, the results are rather unsurprising: As the number of containers increases, instances become harder to solve, as the number of decisions for scheduling and shipping increase and the size of the model grows.

Out of a total of 84 instances per investigated group of solutions with $|I| \in \{1000, 2000, 3000, 4000, 5000\}$, 78 instances with $|I| = 1000$ containers terminated with proven optimality. In contrast, only 23 instances with $|I| = 5000$ reach an optimal solution within the time limit of 5 hours. This is illustrated in the performance plot w.r.t. runtime shown in Fig. 5.10a. When comparing the line representing instances with $|I| = 1000$ to the one with $|I| = 5000$, the times at which instances terminate with proven optimality are more evenly spread across the allowed runtime range for instances with $|I| = 1000$. In contrast, a few easy instances with $|I| = 5000$ terminate within the first few minutes, and some of these instance just terminate right before reaching the time limit, but in between, there are almost no instances terminating.

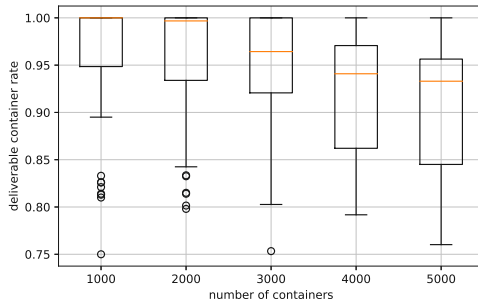
We continue our performance analysis by looking at the number of solutions with different gaps for each group of instances, shown in Fig. 5.10b. We can see that it makes sense to consider solutions having a gap smaller than one percent, as especially for those instances with a higher number of containers, there are much more solutions with gaps



(a) Runtime performance plot



(b) Number of instances with different gaps



(c) Deliverable container rate for optimal instances

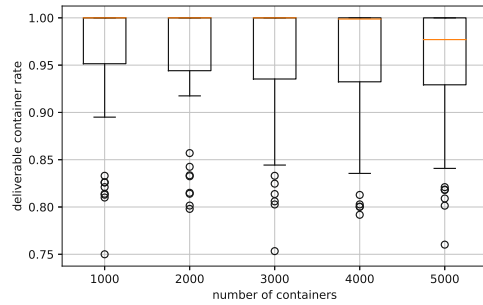
(d) Deliverable container rate for instances with gap $\leq 1\%$

Figure 5.10: Figures for investigating the effect of the number of containers on the runtime, gaps and deliverable container rate of the single-train instances. For the deliverable container rate, we investigate the difference between only considering optimal instances, and also including instances with gap $\leq 1\%$.

of at most one percent then the ones having optimal⁴ solutions. For example, there are 23 optimal solutions for instances with $|I| = 5000$, but 55 instances have a gap smaller than one percent, representing this group of solutions better.

The effects of just considering optimal solutions, or also including the ones with a sufficient gap, can be seen when considering the deliverable container rate. Figure 5.10c shows the deliverable container rate for the optimal instances, while Fig. 5.10d shows it for instances with sufficient gap. We can observe that instances with a high number of containers can be solved to proven optimality more often when having lower deliverable container rates. This makes sense as non-deliverable containers do not have to be considered during the search, making the instance easier. The same is true for instances with sufficient gap, but the effect is much less pronounced. Therefore, it makes sense to include instances with sufficient gap in our considerations, as we get a much better representation of all instances.

⁴Recall that CPLEX considers solutions with a gap $\leq 10^{-4}$ as optimal.

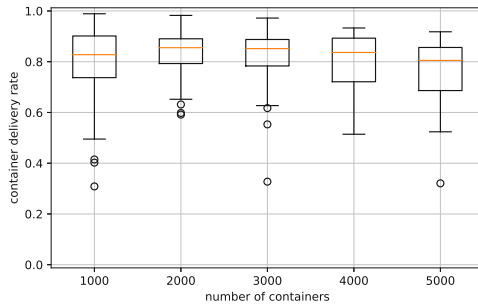
When considering the container delivery rates with increasing number of containers, we can again observe meaningful differences between just considering optimal instances, and ones with sufficient gap. Figure 5.11a shows the container delivery rate for optimal instances, and Fig. 5.11b shows it for instances with sufficient gap. When considering optimal solutions, it appears that the delivery rate only drops slightly when increasing the number of containers. When including instances with sufficient gaps in our consideration, the effect is much clearer. One of the reasons for the difference is clear, as the delivery rate is calculated by dividing the number of delivered containers by the number of *deliverable* containers. As already discussed above, instances with a large number of containers tend to be solved to optimality more easily for those instances with lower container delivery rates. Another explanation is that if more containers can be delivered in the solution (in contrast to begin *deliverable*), the gap is easier to close, as not that many alternative containers have to be considered, and all other containers have their own path. This might lead to higher container delivery rates for the optimally solved instances. Furthermore, the increase in delivery rate at $|I| = 2000$ is notable, as with a higher number of containers, more round trips pay off, leading to overall more delivered containers. Nevertheless, after $|I| = 3000$ it appears that for many instances, the transport capacities of many trains are reached and all round trips that are possible and pay off already take place, resulting in a drop in delivery rates for even higher number of containers.

We consider the active round trip rates next, illustrated in Fig. 5.11c for optimal instances, and in Fig. 5.11d for instances with sufficient gap. We can see that the two plots follow the same trend, but the active round trip rate is notably higher for instances with $|I| = 5000$ containers for instances with sufficient gap, indicating that instances with higher active round trips rates tend to be harder. At $|I| = 1000$, the active round trip is quite low, most likely due to the container density in the network being too low for many round trips to pay off and take place. With an increase in the number of containers, the active round trip rate increases. At $|I| = 4000$, all round trips already take place for more than a quarter of all considered instances, and at $|I| = 5000$, this is the case for half the instances. This indicates that for many instances with $|I| \geq 4000$, the maximum throughput of the network is already reached.

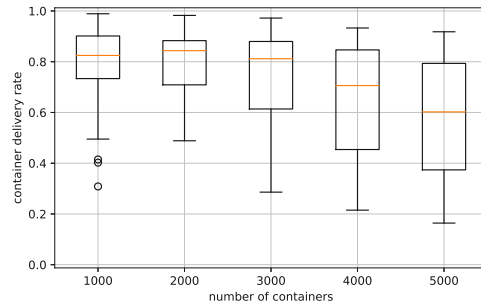
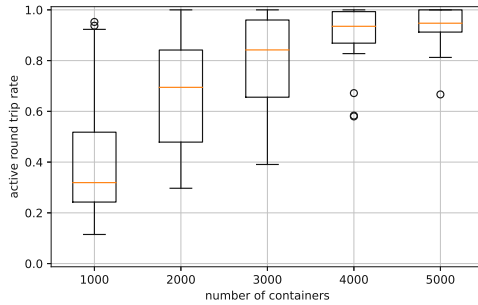
The required train rate, illustrated in Fig. 5.11e for optimal instances, and in Fig. 5.11f for instances with sufficient gap, appears to be very similar for instances of all sizes, i.e., number of containers, except for $|I| = 1000$ where it is significantly lower.

With fewer active round trips and a low container density, only a few containers per terminal might be picked up. In this case, the advantages of our approach become especially clear. As an example, imagine a single train that connects three terminals A-B-C, and transports a single container from B to C and some container from A to C. Our approach uses just one train, as the single container that is transported from B to C is also shipped by this one train. In contrast when only using trains without intermediate stops, a second train from B to C is required. Therefore, the values of the lower bound of direct trains are potentially much higher for these types of instances.

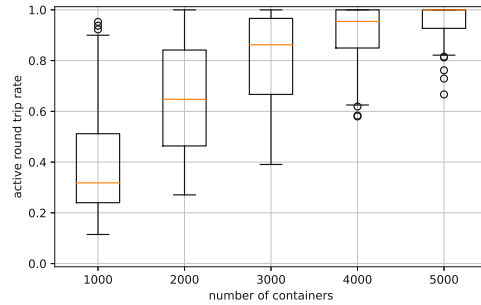
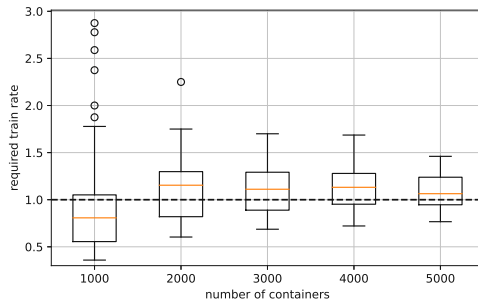
Furthermore, note that for most instances, it appears that the required train rate is above one. We will discuss this in more detail in Section 5.4.



(a) Container delivery rate for optimal instances

(b) Container delivery rate for instances with $\text{gap} \leq 1\%$ 

(c) Active round trip rate for optimal instances

(d) Active round trip rate for instances with $\text{gap} \leq 1\%$ 

(e) Required train rate for optimal instances

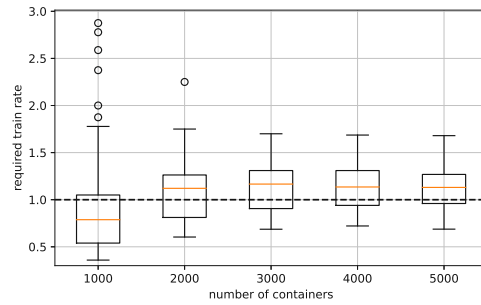
(f) Required train rate for instances with $\text{gap} \leq 1\%$

Figure 5.11: Boxplots of the container delivery rate, the active round trip rate, and the required train rate for the single-train instances, grouped by the number of containers. On the left, only optimal instances are considered, while on the right also instances with $\text{gap} \leq 1\%$ are included.

Unsurprisingly, we can observe that the transport, storage and lifting capacity utilization values are directly proportional to the number of containers, as shown in Fig. 5.12. The more containers are present, the more containers have to be transported, stored and lifted. There are no notable differences for the lifting capacity utilization between optimal instances and ones with sufficient gaps, as can be seen when comparing Fig. 5.12a and Fig. 5.12b.

Nevertheless, there are some noticeable differences when comparing transport and storage capacity utilizations, which are illustrated in Figs. 5.12c to 5.12f, respectively. The transport capacity utilization trends for optimal and sufficient gap instances are similar. Nevertheless, the transport capacity utilization of the latter tends to be a bit higher when compared to the former.

The same can be observed for the storage capacity utilization, but here, we can observe another interesting phenomenon: When considering instances with sufficient gap, the storage capacity utilization appears to drop for $|I| = 5000$ containers. This might be due to the fact that for instances with a large number of containers, only instances with fewer containers present in the network could be solved. This is also supported by the drop in both deliverable container rate and container delivery rate, as illustrated in Figs. 5.10c, 5.10d, 5.11a and 5.11b.

Optimal instances vs. instances with sufficient gap So far, we highlighted the differences arising from only considering optimal instances versus also including instances with sufficient gap into our considerations. We have seen that they follow the same trends with only small differences, especially for harder instances. The instances with sufficient gap appear to provide more reliable insights, as this set contains much more (hard) instances (as well as the optimal ones). Therefore, we focus our further analysis on only those instances with sufficient gap.

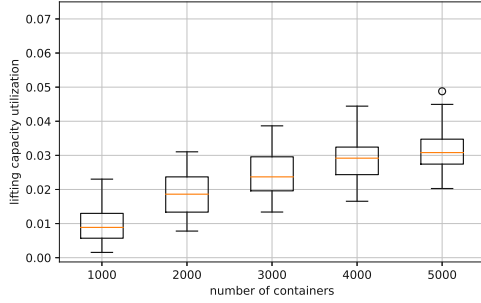
Considered Time Horizon

Next, we consider the effect of the length of the time horizon on our solutions. We group the instances by the number of considered days $h \in \{14, 21, 28, 35\}$, yielding 105 instances per group.

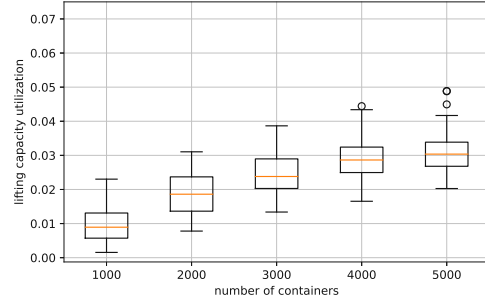
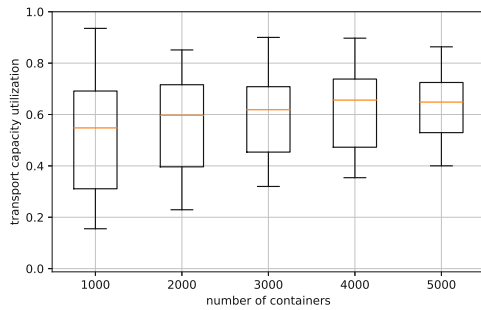
When looking at the runtime performance plot, Fig. 5.13a, it appears that a short time horizon makes the problem harder, as for a time horizon of 14 days, only 49 out of 105 real-world instances terminate with proven optimality, while for a time horizon of 35 days, this is the case for 67 instance.

Nevertheless, when also taking non-optimal instances with sufficient gap into account, we gain further insight. Figure 5.13b shows the number of instances with different gaps per group. Although the number of optimally solved solutions is lower for a shorter time horizon, the total number of solved instances having a sufficient gap is larger, i.e., 89 instances with $h = 14$ have a sufficient gap, while for $h = 35$ we only have 80 instances.

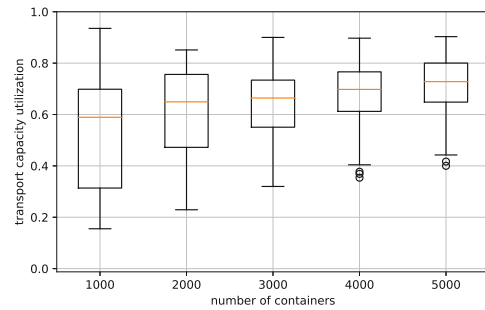
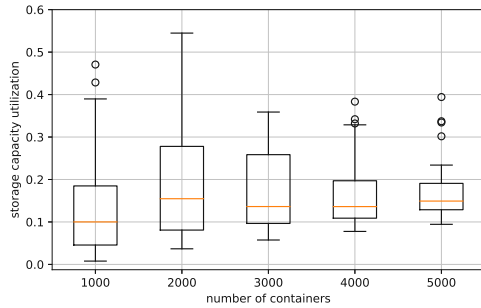
This means that finding good solutions for instances with a smaller time horizon is easier, but solving them to proven optimality is harder, compared to instances with a larger time horizon. This is plausible when considering that for a constant number of



(a) Lifting capacity utilization for optimal instances

(b) Lifting capacity utilization for instances with $\text{gap} \leq 1\%$ 

(c) Transport capacity utilization for optimal instances

(d) Transport capacity utilization for instances with $\text{gap} \leq 1\%$ 

(e) Storage capacity utilization for optimally solved instances

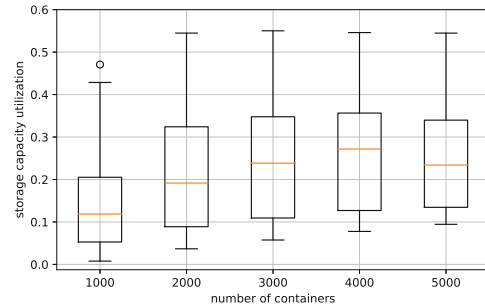
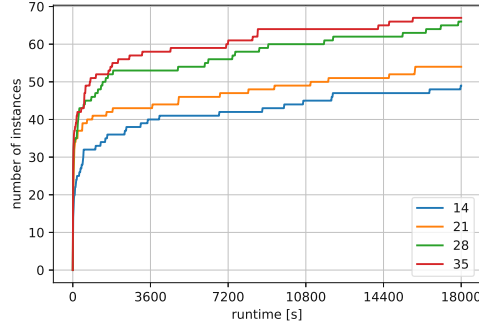
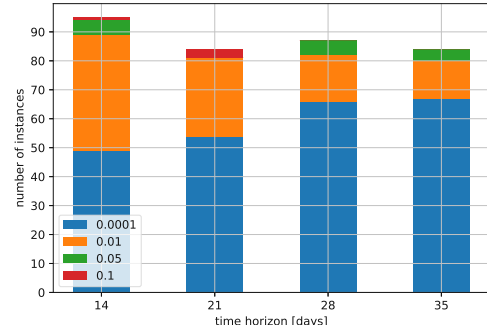
(f) Storage capacity utilization for instances with $\text{gap} \leq 1\%$

Figure 5.12: Boxplots of different capacity utilization values for the single-train instances, grouped by the number of containers. On the left, only optimal instances are considered, while on the right also instances with $\text{gap} \leq 1\%$ are included.

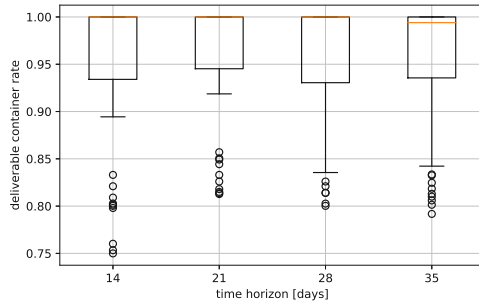
5. RESULTS



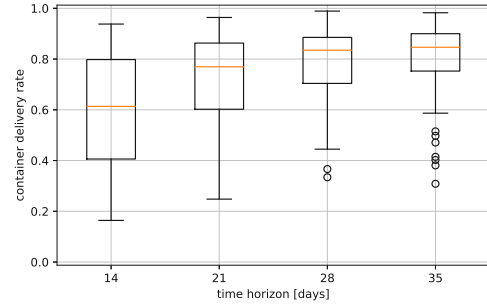
(a) Runtime performance plot



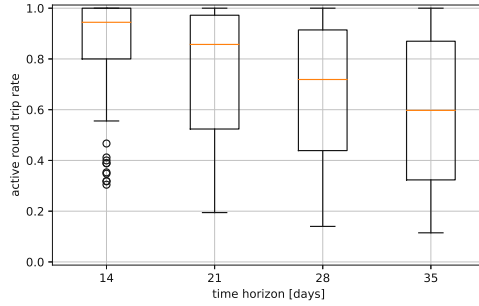
(b) Number of instances with different gaps



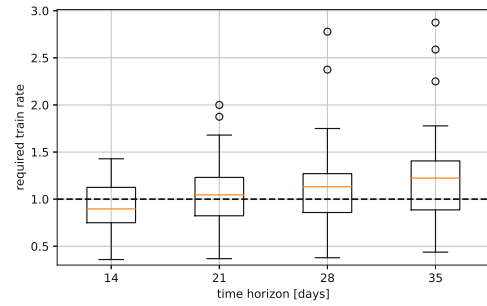
(c) Deliverable container rate for instances with $\text{gap} \leq 1\%$



(d) Container delivery rate for instances with $\text{gap} \leq 1\%$



(e) Active round trip rate for instances with $\text{gap} \leq 1\%$



(f) Required train rate for instances with $\text{gap} \leq 1\%$

Figure 5.13: Figures for investigating the effect of the length of the considered time horizon (in days) for the single-train instances.

containers, the container density $\frac{|I|}{t_{\max}}$ of instances with a short considered time horizon is much higher, hence there are much more choices to consider for, e.g., a single train (at a single stop), which makes closing the gap hard.

The decrease of the container density also explains an increase in container delivery rates with larger considered time horizons, shown in Fig. 5.13d for instances with sufficient gap, as well as the drop of the active round trip rate for larger time horizons, shown in Fig. 5.13e.

The deliverable container rates of instances (with sufficient gap) do not differ significantly for the different time horizons, as can be seen in Fig. 5.13c.

When looking at the required train rate, illustrated in Fig. 5.13f, we can observe an increase of the required train rate with increasing length of the considered time horizon. This makes sense as time constraints are not respected in the calculation of the lower bound of direct trains. Thus, the required train rate is higher for large time horizons. Furthermore, larger time horizons mean more round trips in the respective instances, leading to an accumulating effect of trains that are not fully utilized, resulting in lower required train rate values. For example, if a train is 90 percent utilized, and we consider a single round trip, the required train rate is one. Nevertheless, if we have ten round trips with the same utilization, the lower bound of required trains is nine, leading to a required train rate of 1.11.

The above claim is also supported when looking at the transport capacity utilization values, shown in Fig. 5.14a. We can observe that, due to lower container densities for larger considered time windows, the transport capacity utilization values drop. Less utilized trains also lead to higher required train rates.

When looking at the weight capacity utilization in Fig. 5.14b, we can verify the claim that it strongly correlates with the transport capacity utilization. In general, the weight capacity utilization is just a bit lower than the transport capacity utilization, as the latter is usually more constraining.

Furthermore, both storage and lifting capacity utilization follow the trend of the transport capacity utilization, as can be seen in Figs. 5.14c and 5.14d, respectively: With larger considered time horizons, the utilization values drop.

A closer look – equal container density over time We have seen that the number of containers is essential for the hardness of our instances. With more containers to consider, the instances become harder to solve. Specifically, the container density $\frac{|I|}{t_{\max}}$ is important for the hardness.

Therefore, we briefly investigate instances with the same container density only. Instances satisfying one of the following tuples have an equal container density over time:

$$(|I|, t_{\max}) \in \{ (2000, 14), (3000, 21), (4000, 28), (5000, 35) \}$$

We get 21 instances per group, consisting of the three instances per container buffer time windows $\text{buff}_i \in \{3, 7, 14\}$, and the seven different configurations of ports and cities.

Now, we can clearly observe that a larger time horizon indeed makes the instances harder, as shown in Fig. 5.15a.

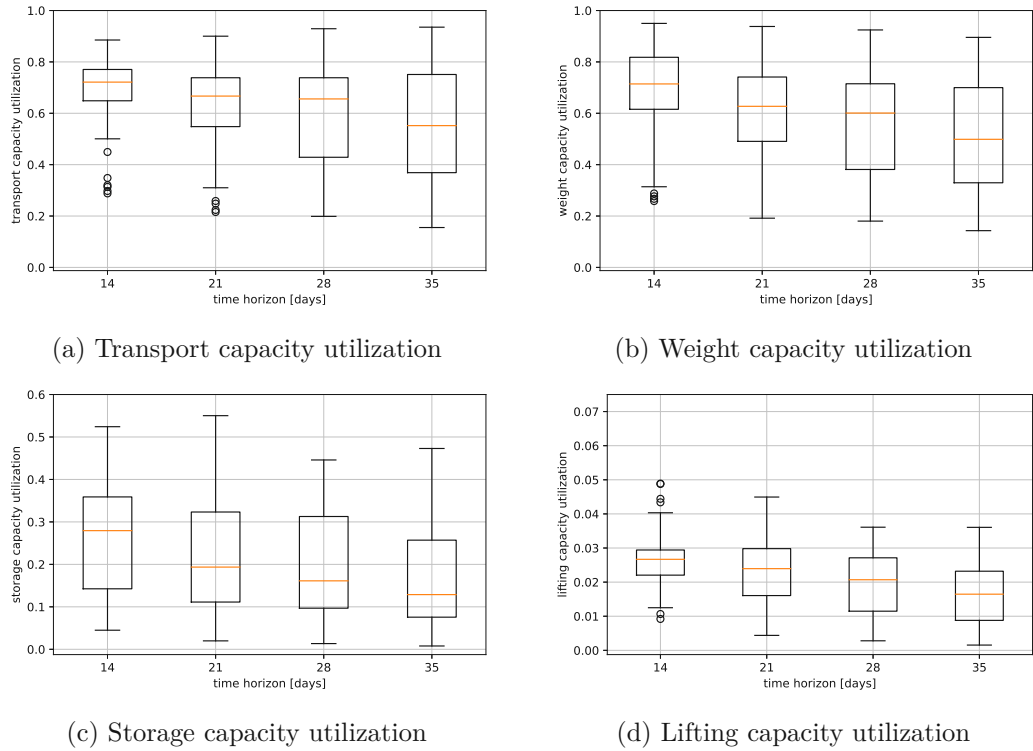


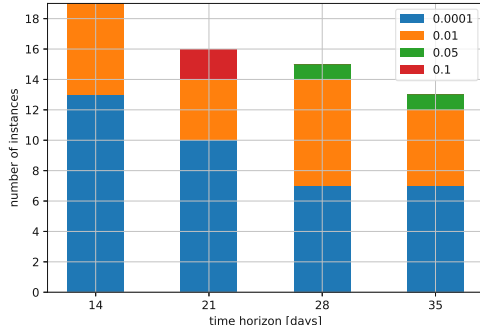
Figure 5.14: Boxplots of various capacity utilization values for single-train instances with $\text{gap} \leq 1\%$, grouped by the length of the considered time horizon in days.

When comparing the deliverable container rates of all of these instances, shown in Fig. 5.15b, to the rates of instances with sufficient gap, shown in Fig. 5.15d, we can see that for larger time horizons, only instances with fewer deliverable containers can be solved. This can be deduced, as Fig. 5.15b shows roughly similar deliverable container rates for all considered instances, whereas lower median values for larger time horizons can be seen in Fig. 5.15d.

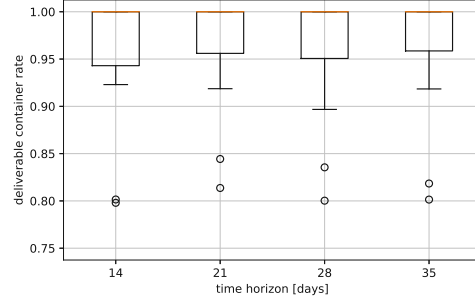
Although the container delivery rate, shown in Fig. 5.15c, roughly follows the previously observed pattern, the active round trip rate, shown in Fig. 5.15e, shows different results: With the consideration of equal container density, the active round trip rate now slightly increases with larger time horizons. This effect can be explained by the non-availability of containers in the beginning and the end of our instances, highlighted in Section 5.2. With increasing time horizons, the beginning and the end are outweighed by the rest of the (now larger) time horizons. We can further notice a drop of the container delivery rates for instances with the largest time horizons.

The observations for the required train rate remain the same: With increasing length of the considered time horizon, the required train rates grow. This can be seen in Fig. 5.15f for instances with sufficient gap.

The transport capacity utilization, shown in Fig. 5.16a, differs slightly for the different



(a) Number of instances with different gaps



(b) Deliverable container rates for all instances

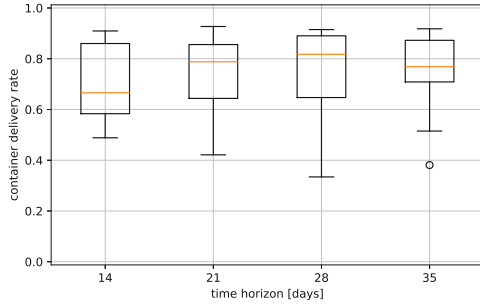
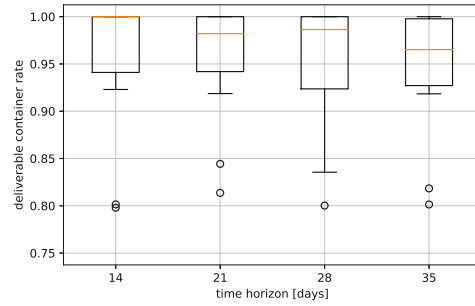
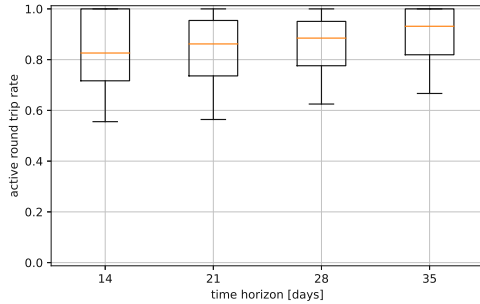
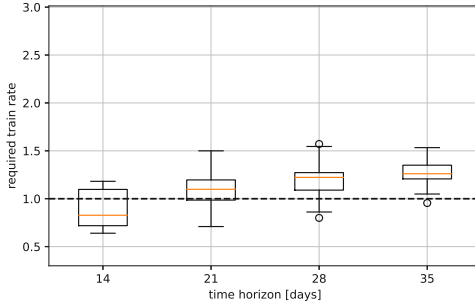
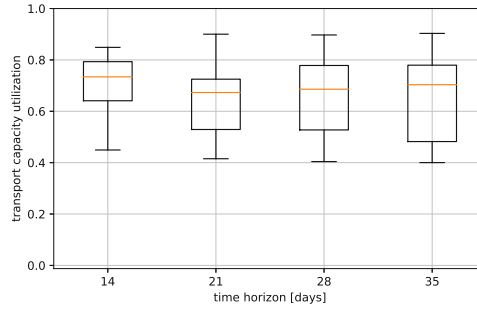
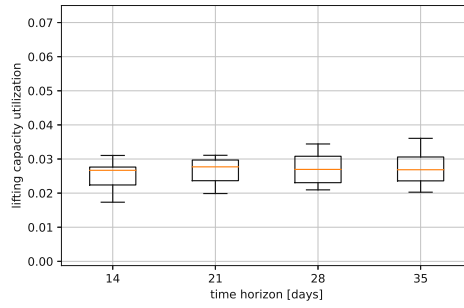
(c) Container delivery rates for instances with gap $\leq 1\%$ (d) Deliverable container rates for instances with gap $\leq 1\%$ (e) Active round trip rates for instances with gap $\leq 1\%$ (f) Required train rates for instances with gap $\leq 1\%$

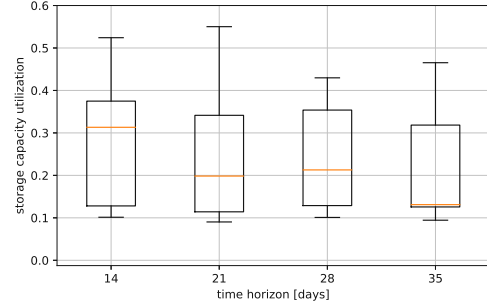
Figure 5.15: Figures for analyzing single-train instances with equal container density. The instances fulfill $(|I|, t_{\max}) \in \{(2000, 14), (3000, 21), (4000, 28), (5000, 35)\}$.



(a) Transport capacity utilization



(b) Lifting capacity utilization



(c) Storage capacity utilization

Figure 5.16: Figures for analyzing the capacity utilization values for single-train instances with the same container density. The instances fulfill $(|I|, t_{\max}) \in \{(2000, 14), (3000, 21), (4000, 28), (5000, 35)\}$ and have a gap $\leq 1\%$.

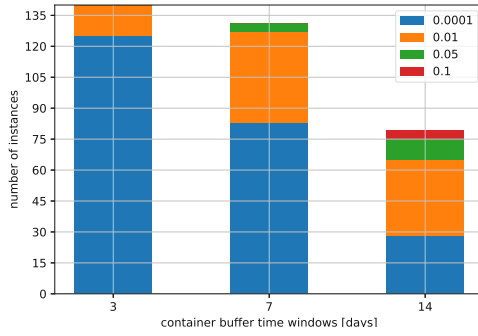
groups. For $h = 14$, the values tend to be higher, dropping at $h = 21$, and then increase again with larger time horizons. This agrees with our previous argument, as a higher container density in the middle of the considered time horizon enables the transportation of more containers in the round trips that take place.

The lifting capacity utilization values are about the same for all groups of instances, as can be seen in Fig. 5.16b.

Interestingly, the storage capacity utilization values appear to drop with larger time horizons. This is illustrated in Fig. 5.16c. A possible explanation for this is that instances with larger time horizons also have more containers in this considered setting, which are harder to solve. Thus, fewer instances with a large number of delivered containers could be solved for larger time horizons, leading to lower storage capacity utilization values.

Container Buffer Time Windows

When considering the effect of size of the container buffer time windows $\text{buff}_i \in \{3, 7, 14\}$ days, we have 140 instances per group. The number of instances per group that could be solved with a given maximum gap are shown in Fig. 5.17a. Out of 140 instances with a



(a) Number of instances with different gaps

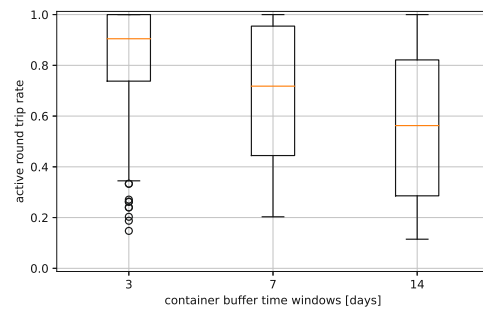
(b) Deliverable container rate for instances with gap $\leq 1\%$ (c) Container delivery rate for instances with gap $\leq 1\%$ (d) Active round trip rate for instances with gap $\leq 1\%$

Figure 5.17: Figures for investigating the effect of the container buffer time windows $\text{buff}_i \in \{3, 7, 14\}$ (in days) on the single-train instances.

buffer time window $\text{buff}_i = 3$ days, 125 instances terminate with proven optimality, while for $\text{buff}_i = 7$ days, we still have 83 optimally solved instances, and for instances with $\text{buff}_i = 14$ days, only 28 instances terminated with proven optimality. This suggests that having small time windows leads to easier instances. This is for two reasons:

As mentioned in Section 5.3.1, the deliverable container rate is lower when the buffer time windows are smaller, as many containers cannot be delivered at all. Having fewer containers to consider makes instances easier, as already mentioned. ?? shows the deliverable container rate for instances with sufficient gap, grouped by the different buffer time windows.

In addition, longer buffer time windows lead to more routing options for the containers, hence to harder instances. This effect can be seen clearly when considering the low number of solved instances for $\text{buff}_i = 14$, suggesting that the choice of appropriate container buffer time windows is essential for the hardness of the instances.

When looking at the container delivery rates, shown in Fig. 5.17c we can see an unexpected drop for $\text{buff}_i = 14$. This is surprising, as we would expect higher container delivery rates for larger buffer time windows, as these allow a greater flexibility in

scheduling. Again, we have two explanations.

The first explanation is that larger buffer time window lead to harder instances, and only instances with lower delivery rates could be solved with sufficient gap.

The second explanation is that for large buffer time windows, active trains (and round trips) can almost be fully utilized, leaving other round trips with too few containers to take place. An example is that instead of having three trains that are 75 percent utilized, the larger buffer time windows allow us to schedule the containers in a way to only use two fully utilized trains. Nevertheless, the remaining 25 percent of containers of the last train remain non-delivered, as the last train does not pay off with a load of only 25 percent. This is supported by the drop in active round trip rates, shown in Fig. 5.17d, and the increasing transport capacity utilization for larger container buffer time windows, shown in Fig. 5.18b.

When considering the required train rate, illustrated in Fig. 5.18a, we can see that for instances with $\text{buff}_i = 3$, the values are often above one, while for $\text{buff}_i = 7$, the median is below one, and for $\text{buff}_i = 14$, it is about one. Lower transport capacity utilizations lead to higher required train rates and vice versa. When considering instances with $\text{buff}_i = 14$, it appears that the required train rate increases compared to instances with $\text{buff}_i = 7$. Nevertheless, this is due to the fact that more instances with fewer terminals could be solved in general, which tend to have higher required train rates (see Section 5.4).

With lower container delivery rates for $\text{buff}_i = 14$, the lifting capacity utilization values also drop, as can be seen in Fig. 5.18d.

In contrast, the storage capacity utilization drastically increases for longer container buffer time windows, as show in Fig. 5.18c. This is plausible, as with larger buffer time windows, (delivered) containers are guaranteed to remain longer in our network and have to be stored at the terminals.

Number of Terminals (Ports and Cities) (& Number of Trains)

In this section, we consider the number of terminals. Nevertheless, the observations of this section also hold for our instances when considering the number of trains, as the number of trains $|L|$ strongly correlates with the number of ports $|P|$ and cities $|C|$: The number of trains follows the formula $|L| = |P| + |C| - 1$, except for $(|P|, |C|) = (3, 3)$ where $|L| = 5$, and $(|P|, |C|) = (3, 4)$ where $|L| = 7$. Table 3 in the appendix contains the rail networks of all numbers of terminals, and their associated train lines⁵. In the following, the plots are therefore ordered by ascending number of terminals and trains.

Similar to previous observations, the container density per terminal and train decreases as the number of terminals increases, making the problem easier (especially w.r.t. proven optimality). At the same time, more routing options are potentially available for each container, making the problem harder. This is reflected in the performance plot w.r.t. runtime, see Fig. 5.19a, and in Fig. 5.19b showing the number of instances with different gaps per $(|P|, |C|)$ group, each with a total of 60 instances. When looking at the

⁵Note that for the single-train instances under consideration, the number of trains lines also corresponds to the number of trains.

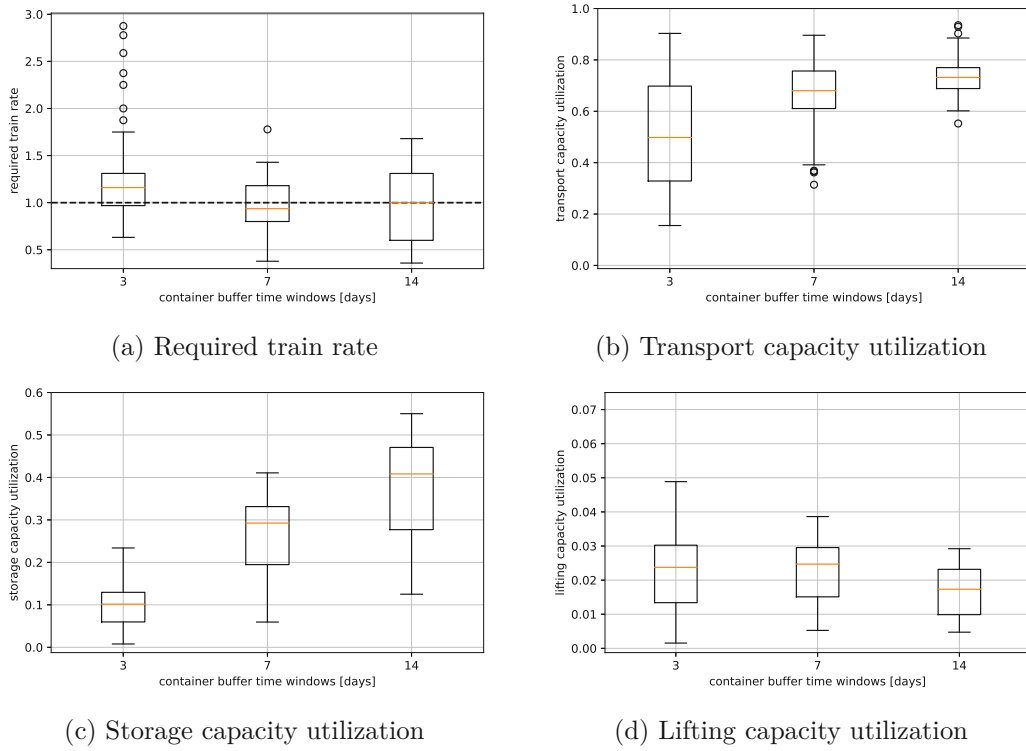


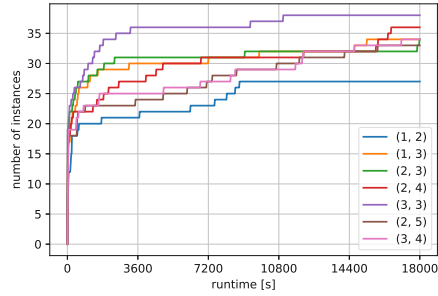
Figure 5.18: Required train rate and capacity utilization values for single-train instances with $\text{gap} \leq 1\%$, grouped by their container buffer time windows $\text{buff}_i \in \{3, 7, 14\}$ (in days).

performance plot, instances with $(|P|, |C|) = (1, 2)$ yield the lowest number of optimal solutions with only 27 out of 60 instances, while $(3, 3)$ -instances have the most optimally solved solutions with 38 out of 60, and $(3, 4)$ -instances yield 34. Nevertheless, when also considering solutions with sufficient gap, we see the opposite: 59 of the $(1, 2)$ -instances have a gap smaller than one percent, while the instances with $(p, c) = (3, 4)$ now have the fewest out of all instance groups with only 38 solutions.

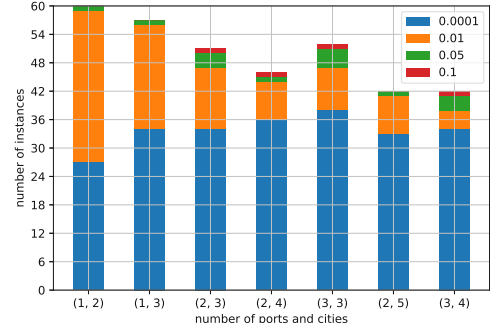
Knowing that the deliverable container rate plays a significant role in the number of instances that can be solved, we have a look at Fig. 5.19d that shows the deliverable container rate for instances with container buffer time windows of $\text{buff}_i = 3$. We know from Section 5.3.1 that these instances are the ones with deliverable rates that are significantly lower than one. Once again, when comparing Fig. 5.19d with Fig. 5.19b, we can see that instances tend to be solved more easily when the deliverable container rate is low.

It is interesting to note that instances with $(|P|, |C|) = (2, 3)$ and $(3, 3)$ yield much lower deliverable container rates. Figure 5.19c shows the rail network for $(|P|, |C|) = (3, 4)$. When considering the sizes of the terminals, we know that the last two ports to add to our network (when increasing its size w.r.t. the number of terminals) are RTD and

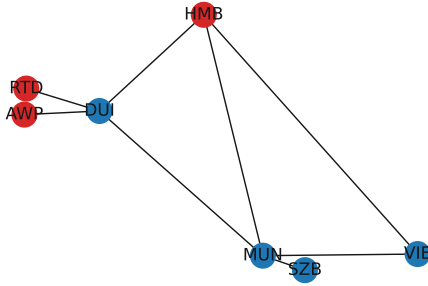
5. RESULTS



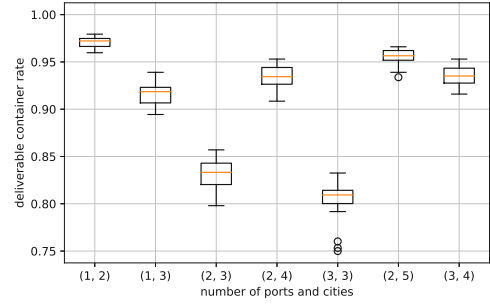
(a) Runtime performance plot



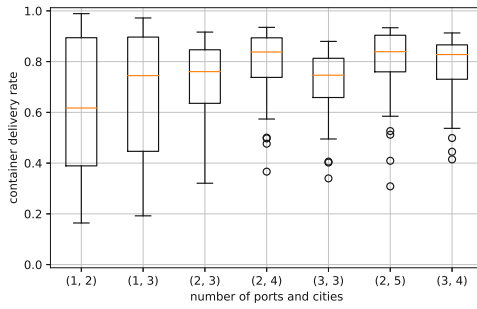
(b) Number of instances with different gaps



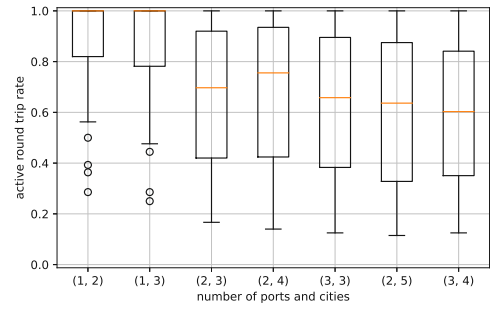
(c) Rail network with $(|P|, |C|) = (3, 4)$



(d) Deliverable container rate for *all* instances with $\text{buff}_i = 3$.



(e) Container delivery rate for instances with $\text{gap} \leq 1\%$



(f) Active round trip rate for instances with $\text{gap} \leq 1\%$

Figure 5.19: Figures for investigating the effect of the number of terminals on the single-train instances. Each group is denoted by $(|P|, |C|)$, where $|P|$ is the number of ports and $|C|$ is the number of cities.

AWP, and the last city is SZB. An explanation for the lower deliverable rates for (2,3) and (3,3) is that with the addition of one (or two) ports, the long-lasting round trip RTD-DUI-MUN-VIE (AWP-DUI-MUN-VIE) is added, and with narrow time windows many containers originating from VIE cannot be transported in time. When adding the additional city SZB, the number of container at VIE is reduced, increasing the deliverable container rate again.

When considering the container delivery rate and the active round trip rate, shown in Fig. 5.19e and Fig. 5.19f respectively, we can see that a low number of terminals (resulting in a higher density of containers per terminal and train), leads to higher active round trip rates. At the same time, the median of the container delivery rate is lower, but deviations are much higher than for instances with more terminals. This indicates that for instances with just one port, the other instance parameters strongly influence the container delivery rate. The same conclusions can be drawn when considering the high deviations of the active round trip values for instances with more than one port.

Lower container densities might also be the reason for dropping transport capacity utilization values when increasing the number of terminals. These values can be seen in Fig. 5.20b. The same can be said about the storage and lifting capacity utilization, illustrated in Fig. 5.20c and Fig. 5.20d, but the effect is less pronounced.

The required train rate, shown in Fig. 5.20a, appears to be above one for most instances with one port. This is plausible when considering the fact that instances with only one port have trains that are scheduled in a way that each city is directly connected by a single train line to the port. For containers that are transported from city to city, two trains are required instead of having a single direct connection. Thus, the required train rate values are higher for these instances. In contrast, it appears that with a higher number of ports (and terminals in general), the required train rate drops, with medians of all groups being below one.

5.3.4 Daily-Train Instances

In the previous section, we mostly considered instances generated with the single-train scheduling method. These instances have just one scheduled train per train connection (potentially performing recurring round trips). In this section, we briefly consider daily-train instances, where for every train connection, a train leaves the home terminal every day at the same time, leading to an increased number of trains. While single-train instances have at most 7 trains, daily-train instances have up to 27 trains.

When comparing the deliverable container rates of the daily-train instances to the single-train instances, it is rather unsurprising that the deliverable container rates are higher for the daily-train instances. Even instances with narrow container buffer time windows have more containers that can be delivered, as more trains are scheduled in the daily-train instances. Nevertheless, there are still instances where less than ninety percent of containers are deliverable. Figure 5.21a shows the deliverable rates for all single-train instances, and Fig. 5.21b for all daily-train instances.

As discussed in Section 5.3.3, 332 instances of the single-train have a sufficient gap, while only 123 of the daily-train instances do so. In order to make the results comparable,

5. RESULTS

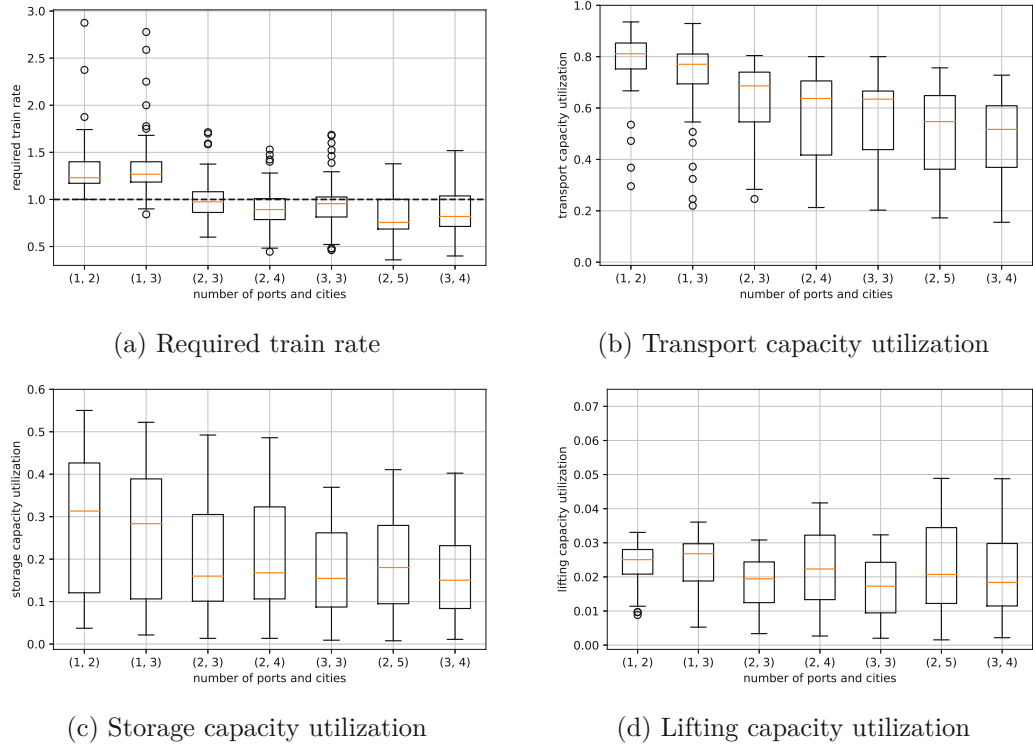


Figure 5.20: Boxplots for the required train rate and capacity utilization values for single-train instances with $\text{gap} \leq 1\%$, grouped by their respective number of terminals. Each group is denoted by $(|P|, |C|)$, where $|P|$ is the number of ports and $|C|$ is the number of cities.

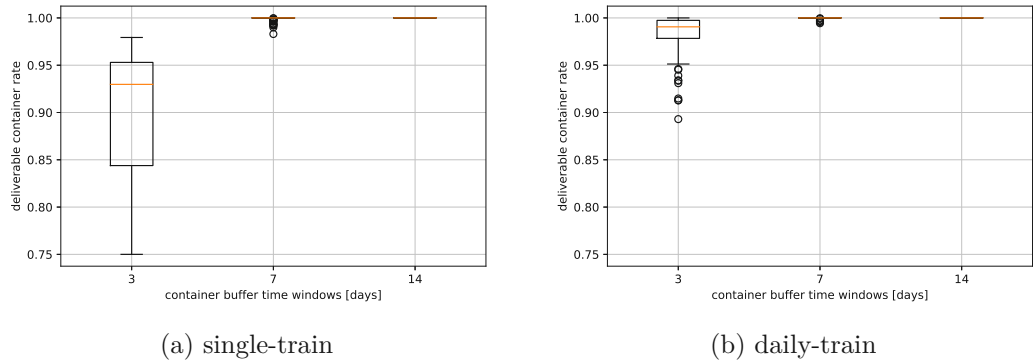


Figure 5.21: Boxplots of the deliverable container rates for all single-train instances on the left, and for all daily-train instances on the right. The instances are grouped by length of the container buffer time windows $\text{buff}_i \in \{3, 7, 14\}$.

we filter out the single-train instances that do not correspond to the 123 daily-train instances with respect to their respective instance parameters. The remaining 123 single-train instances have the same parameters as the considered daily-train instances (except for the train scheduling method and, therefore, the number of trains)⁶.

Figure 5.22 contains boxplots for the values of the container delivery rates, the active round trip rates, and required train rates of the daily-train instances on the right side, and the corresponding single-train instances on the left.

The container delivery rate is higher for the daily-train instances, and the active round trip rate is much lower. In the daily-train instances, much more trains are available, but still only the required round trips take place, leading to lower active round trip values. Nevertheless, the higher container delivery rates of the daily-train instances indicate that the total number of active round trips is higher, as more containers have been shipped (and the transport capacity utilization values are not significantly higher).

A notable difference to the previously observed trends is that the daily-train instances do *not* have dropping delivery rate values with a higher number of containers. In contrast to the single-train instances, where the maximum throughput of the network appears to be reached for many $|I| = 3000$ already, the daily-train instances have sufficiently many trains (and round trips) to cover the increasing demand.

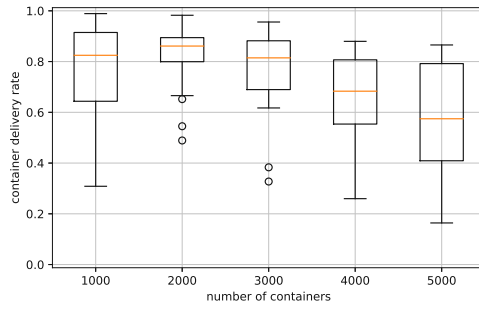
When considering the required train rates, the daily-train instances appear to have slightly higher values. When taking the transport capacity utilization into account, shown in Fig. 5.23b and Fig. 5.23a for daily-train instances and the corresponding single-train instances respectively, we can see that the daily-train instances have lower values. With more transported containers and active round trips (indicated by the increased container delivery rates) and less train capacity utilization, the required train rate values naturally increase.

Next, we further investigate the capacity utilization values of the daily-train instances in comparison to the single-train instances. The corresponding plots are shown in Fig. 5.23. Again, plots of the daily-train instances (with sufficient gap) are on the right side, while the corresponding single-train capacity utilization values are on the left. We also consider the values of the single-train instances (previously) shown in Fig. 5.12. (These plots contain not only values of the corresponding, but values of all single-train instances with sufficient gap.)

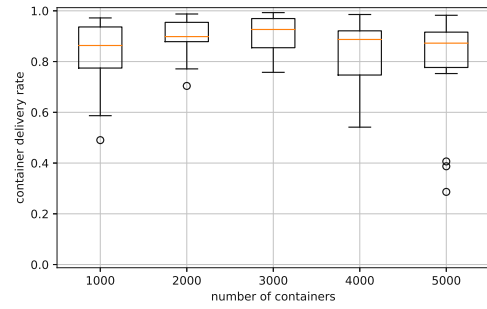
The transport capacity utilization values of the daily-train instances are shown in Fig. 5.23b, and the ones for the corresponding single-train instances in Fig. 5.23a (and Fig. 5.12c and Fig. 5.12d). When comparing the daily-train instances to the corresponding single-train instances, it appears that for a small number of containers, the daily-train has slightly higher transport utilization values than the corresponding single-train instances, while for a large number of containers, it has slightly lower transport utilization values. Nevertheless, when looking at the plots containing all single-train instances, we have a similar distribution of values. An explanation for the observed difference to the corresponding single-train instances is that for small number of containers, the container

⁶Note that the same number of containers was generated for each instance pair, but the parameters of the individual containers differ.

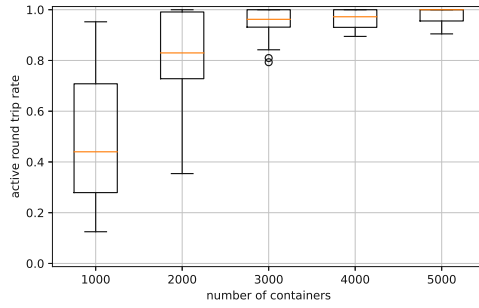
5. RESULTS



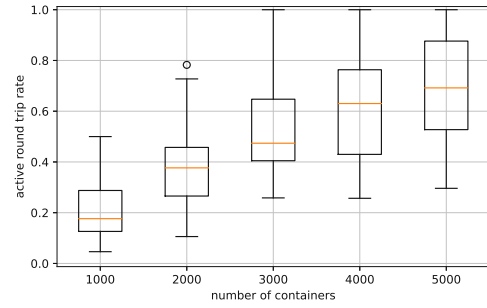
(a) Container delivery rate for the corresponding single-train instances



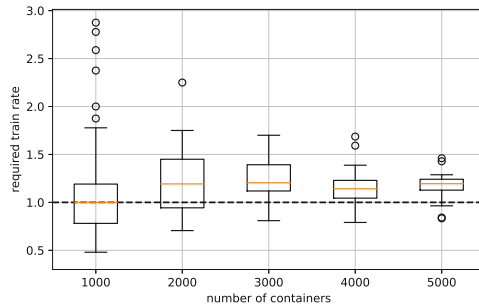
(b) Container delivery rate for the daily-train instances with $\text{gap} \leq 1\%$



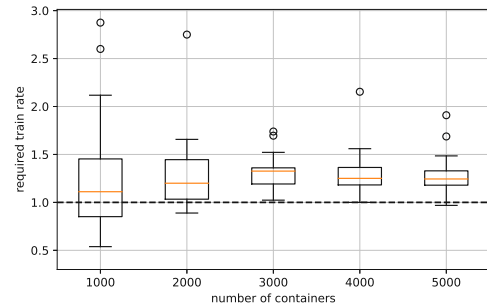
(c) Active round trip rate for corresponding single-train instances



(d) Active round trip rate for the daily-train instances with $\text{gap} \leq 1\%$

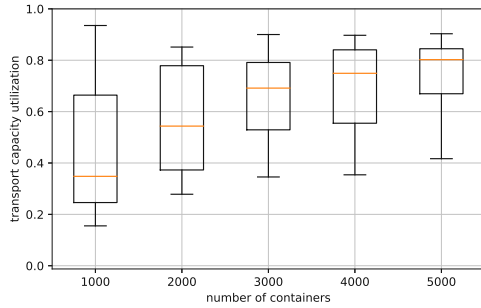


(e) Required train rate for corresponding single-train instances

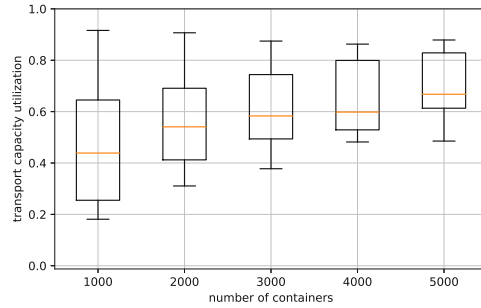


(f) Required train rate for the daily-train instances with $\text{gap} \leq 1\%$

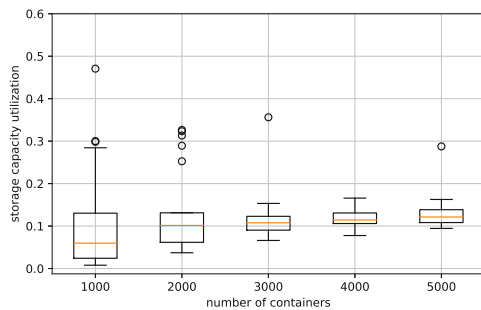
Figure 5.22: Delivery rate, active round trip rate, and required train rate values for the daily-train instances with $\text{gap} \leq 1\%$ are shown on the right side. Plots for the *corresponding* single-train instances are shown on the left. This set of single-train instances differs only in the train scheduling method, but otherwise has the same instance parameters as the respective daily-train instances, yielding the same number (123) of considered instances.



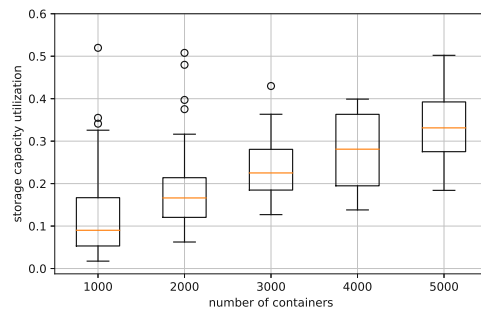
(a) Transport capacity utilization for the corresponding single-train instances



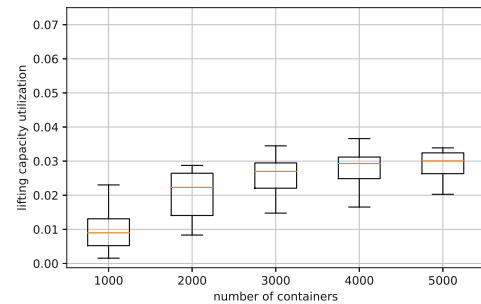
(b) Transport capacity utilization for the daily-train instances with $\text{gap} \leq 1\%$



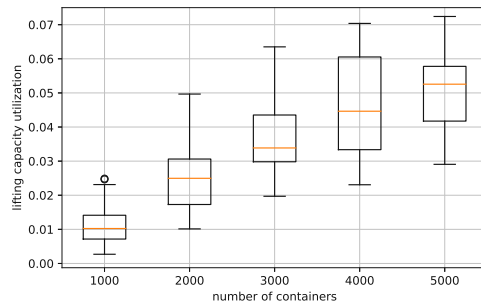
(c) Storage capacity utilization for the corresponding single-train instances



(d) Storage capacity utilization for daily-train instances with $\text{gap} \leq 1\%$



(e) Lifting capacity utilization for the corresponding single-train instances



(f) Lifting capacity utilization for the daily-train instances with $\text{gap} \leq 1\%$

Figure 5.23: Boxplots for comparing the different capacity utilization values for the daily-train instances with $\text{gap} \leq 1\%$, plotted on the right, with their corresponding single-train instances plotted on the left.

density in the single-train instances is so low that the active round trips cannot be utilized well. In contrast, the daily-train instances can pick the best choice of all active round trips for transporting more containers. When considering instances with larger number of containers (resulting in higher container densities), almost all round trips of the single-train instances are active to transport as many containers as possible, leading to higher train utilization values. For the daily-train instances, much more round trip can be active. Therefore, the container density is not high enough to increase the transport capacity utilization, as it is the case for the single-train instances.

When considering the storage and lifting capacity utilization values, shown in Fig. 5.23d and Fig. 5.23f, we compare them to Fig. 5.23c and Fig. 5.23e respectively, (and also to Fig. 5.12e, Fig. 5.12f, and to Fig. 5.12a and Fig. 5.12b respectively). We can see that especially for instances with a large number of containers, the daily-train instances tend to have higher utilization values. This makes sense as the container delivery rate is also much higher for these instances, hence more containers are transported, lifted and stored. Note that this is not directly reflected in the transport capacity utilization values, as potentially more trains (or round trips) are used, but the individual trains (and round trips) are not utilized more on average.

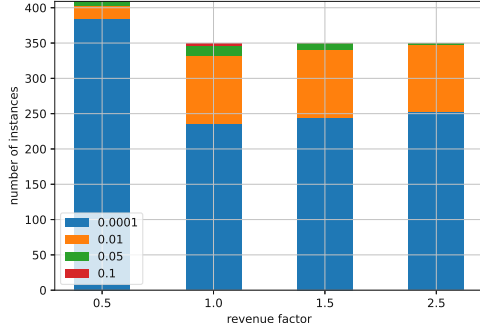
5.3.5 Revenue Factor

In the previous sections, we considered instances with a revenue factor of $\rho = 1$, i.e., whose costs and revenue parameters were derived from real-world data, as described in Section 4.2.3. To investigate the effects of varying profit margins on the resulting solutions, we will now compare them to additionally performed benchmark runs of our single-train instances with different revenue factors $\rho \in \{0.5, 1.5, 2.5\}$.

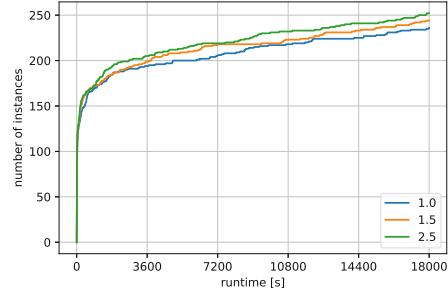
Figure 5.24a shows the number of (single-train) instances with respect to their gap, grouped by their respective earning rates $\rho \in \{0.5, 1, 1.5, 2.5\}$. Every instance group has a total of 420 instances. Figure 5.24b and Fig. 5.24c are the performance plots for runtime and gaps of instances with $\rho \in \{1, 1.5, 2.5\}$. The remaining three plots, Fig. 5.24d, Fig. 5.24e and Fig. 5.24f show boxplots of the container delivery rate, the active round trip rate, and the required train rate respectively. For these and the following plots, we again only consider instances with sufficient gap. Figure 5.25 contains boxplots for the capacity utilization values.

In Fig. 5.24a we can clearly see that the instances with a revenue factor of $\rho = 0.5$ are much easier to solve than the other groups. The reason for this gets clear when looking at the container delivery rate: It is zero for almost all instances, simultaneously resulting in low active round trip rates, as the container revenues are too low to make the round trips pay off and take place. For the few instances where indeed some containers were delivered, the transport capacity utilization is high, as it only then pays off to ship containers with low revenue at all.

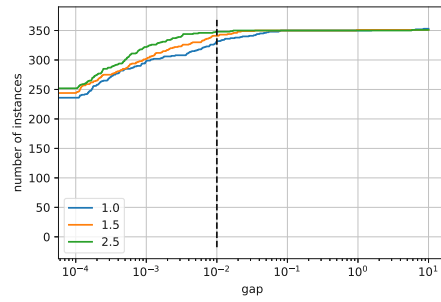
When looking at the higher revenue factors $\rho = 1.5$ and $\rho = 2.5$, we can observe that the instances also appear to get slightly easier with increasing revenue factor. This can be seen in the performance plots Fig. 5.24b and Fig. 5.24c. 332 instances yielded sufficient



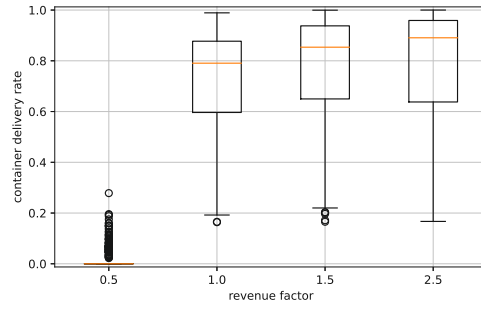
(a) Number of instances with different gaps



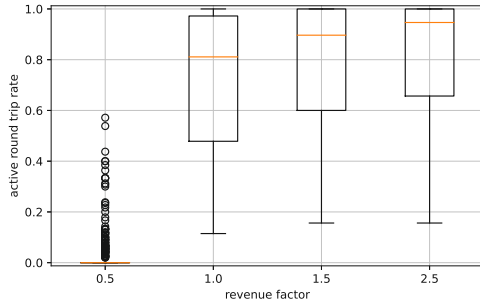
(b) Performance plot of runtimes



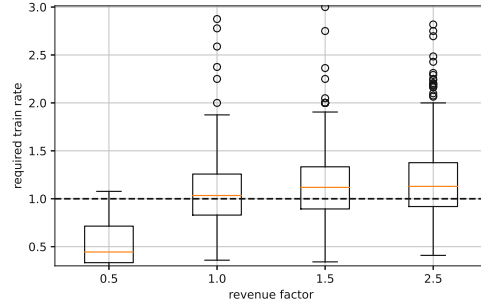
(c) Performance plot of gaps



(d) Container delivery rate for instances with gap ≤ 1%

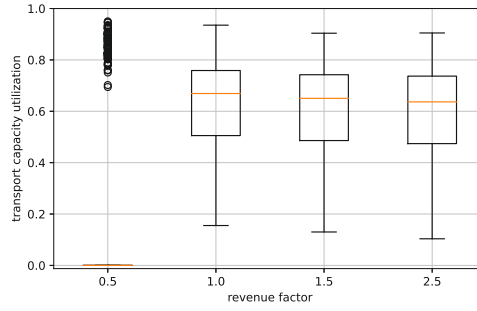
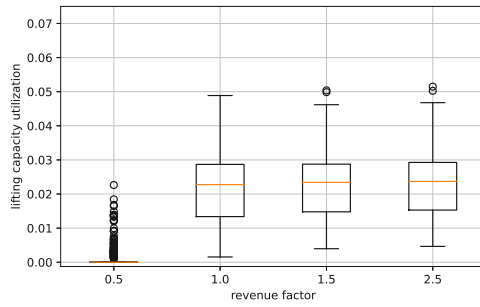
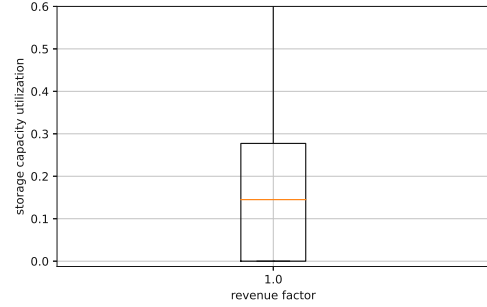


(e) Active round trip rate for instances with gap ≤ 1%



(f) Required train rate for instances with gap ≤ 1%

Figure 5.24: Figures for investigating single-train instances with earning rates $\rho \in \{0.5, 1, 1.5, 2.5\}$.

(a) Transport capacity utilization for instances with gap $\leq 1\%$ (b) Lifting capacity utilization for instances with gap $\leq 1\%$ (c) Storage capacity utilization for instances with gap $\leq 1\%$ Figure 5.25: Boxplots of the capacity utilization values for investigating single-train instances with earning rates $\rho \in \{0.5, 1, 1.5, 2.5\}$.

gaps (\leq one percent) when considering the $\rho = 1$ instance group, while for $\rho = 1.5$ we have 341 instances, and for $\rho = 2.5$ a total of 348 instances have sufficient gaps.

Unsurprisingly, the active round trips rate values increase with higher revenue factors, as more and more round trips pay off with higher container revenues. At the same time, even fewer containers are transported on each train, as indicated by the drop in transport capacity utilization, shown in Fig. 5.25a. Simultaneously, the container delivery rates and the required train rate both increase with higher revenue factors, (as can be seen in Fig. 5.24d and Fig. 5.24f), as well as the lifting and storage capacity utilization values (shown in Fig. 5.25b and Fig. 5.25c respectively).

5.4 Summary of Results and Practical Insight

In the previous sections, we have analyzed our instances and their solutions with respect to the solutions characteristics from Section 5.1 quite thoroughly. In this section, we summarize the key results from above and consider practical aspects of our problem and our results.

5.4.1 Factors of Hardness

We have seen that the number of containers, and especially the container density over time, significantly influence the hardness of our instances. Nevertheless, being able to consider large numbers of containers is important for practice, as in real-world scenarios, we might have to consider many more containers than in the instances we currently investigated.

The hardness of the problem also grows with an increased number of trains and considered terminals, and with the length of the considered time horizon (when the container density remains the same).

An important difference between the single-train and the daily-train instances is that the latter are harder, but yield higher container delivery rates for high numbers of containers as more trains (and round trips) are available.

Furthermore, we have seen that the length of the container buffer time windows is essential for the hardness. Customers prefer small time windows, as this gives them more certainty when planning their operations. Nevertheless, we have seen that the number of deliverable containers drops significantly with narrow buffer time windows. The same holds true for the transport capacity utilization values. Therefore, out of our investigated values $\text{buff}_i \in \{3, 7, 14\}$, picking an intermediate value of seven days as the buffer time windows appears to be best in practice.

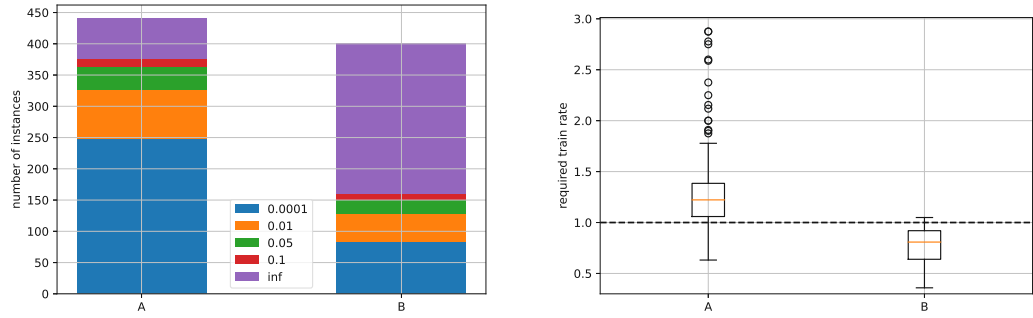
5.4.2 (Incomparable) Train Utilization

When comparing the transport capacity utilization of our instances with real-world data, Black et al. [10] mention a typical load factor (=average fill of a vehicle) of 75% on average [10, p. 16].

Nevertheless, it is difficult to compare our values to actual real-world data, as our instance are artificially generated. We have seen that the transport capacity utilization highly depends on the available number of containers, and more specifically the container density over time, per terminal and train. If the container density is sufficiently high, and the container buffer time windows are sufficiently large, the trains can be utilized well. If this is not the case, they are utilized poorly. In practice, many more containers (and higher container densities) than we have considered in this study will have to be accommodated, leading to potentially higher transport capacity utilization values than we have observed for our instances.

Another notable fact is that the revenues of the individual containers and the individual round trip costs influence the train utilization significantly. We have seen in Section 5.3.5 that with higher container revenues, the transport capacity utilization drops. The opposite is the case when increasing the round trip costs: With higher round trip costs, more container revenue has to be collected for a single round trip to be profitable, hence more containers have to be transported on each round trip. We expect that tuning these parameters would therefore lead to higher transport capacity utilization values.

In summary, we have seen that our instances tend to have lower transport capacity utilization values than 75 percent. Nevertheless, a fair comparison to this value can only



(a) Number of instances with different gaps – “inf” addresses instances with larger gaps or no solution.

(b) Required train rate for instance with gap $\leq 1\%$

Figure 5.26: Group “A” contains instances that either have just one port or container buffer time windows with length $\text{buff}_i = 3$ days, while group “B” contains the remaining instances.

be obtained by using real-world data to reflect an authentic container density distribution and a realistic number of containers. Furthermore, adaptations of the container revenues and round trip costs might force higher transport capacity utilization values.

5.4.3 Required Train Rate

Besides the transport capacity utilization, the required train rate is highly relevant for practical insights.

We have seen that the number of considered terminals plays a key role influencing this solution characteristic. If we consider just one port and trains with only two stops (like direct trains) on their round trips, as in our instances with $|P| = 1, |C| \in \{2, 3\}$, the resulting required train rates are high, usually above one. Considering more terminals and trains with more than two stops on their round trips leads to the lower required train rates that we aim for.

The required train rate also profits from larger container buffer time windows, as small buffer time windows lead to low transport capacity utilization values and high required train rates.

In the previous sections, it appeared that the required train rate is mostly above one, indicating that our approach does not work well in practice. Nevertheless, this is due to the overrepresentation of instances with small buffer time windows (three days) and just one port, as these are easier to solve (with sufficient gap). These are precisely the instances where it appears that our method does not perform well for the mentioned reasons.

Figure 5.26a shows the number of instances with different gaps for instances with one port or container buffer time windows of $\text{buff}_i = 3$ days, labeled group “A”, and group “B” are the remaining investigated instances.

When looking at the required train rates of these groups of instances with sufficient gap, illustrated in Fig. 5.26b, we can see that group “B” has required train rate values mostly below one. This indicates that for instances with these types of parameters, our method would outperform a system that uses direct trains.

5.4.4 Low Priority of Freight Trains

Another important factor to consider is that in practice, freight trains have lower priority than passenger trains w.r.t. their scheduled times and the use of the required tracks. The reason for the low priority of freight trains is that passenger trains have to rely on strict schedules to ensure customer satisfaction and their continued use. In contrast, the actual arrival and departure dates of freight trains sometimes deviate from their originally planned ones by multiple days. It is common that freight trains start their journey earlier than planned if the required tracks are available and all containers are already loaded. Conversely, freight trains are often delayed when passenger trains block the required tracks.

When trains are scheduled as we propose in this study, we can only compensate this kind of uncertainty in freight train schedules to some extent. We can increase the dwell times by a buffer compensation time to absorb some of the incurred delays. Nevertheless, this is only possible to some degree, as otherwise the travelling times of the individual round trips would take too long. As containers might be transported by multiple trains, the lateness of a single train might also influence the lateness of subsequent trains. Nevertheless, having high container and train densities might counteract this problem. In this case, containers might just use the next freight train and still arrive on time, while the previous train can carry another container instead.

These factors are important to keep in mind when considering practical implementations of our proposed system.

5.4.5 Investigation of Bottlenecks

Another area of interest for practical implementations is the identification of bottlenecks within the system. Identifying the most occupied resources can potentially lead to cheap improvements of the system’s throughput when the corresponding parameters are accordingly adapted.

One way to identify bottlenecks is to disable individual sets of capacity constraints within our model from Section 3.3.3, specifically the storage capacity constraints (3.3), lifting capacity constraints (3.6), transport capacity constraints (3.4), or weight capacity constraints (3.5).

We performed additional experiments on our previously studied instances to investigate their bottlenecks. Figure 5.27a contains the delivery rates of single-train instances with sufficient gap without any disabled constraints⁷. Figure 5.27b shows the delivery rates of the same set of instances run with disabled lifting constraints, and Fig. 5.27c contains the

⁷identical to Fig. 5.11b from Section 5.3.3

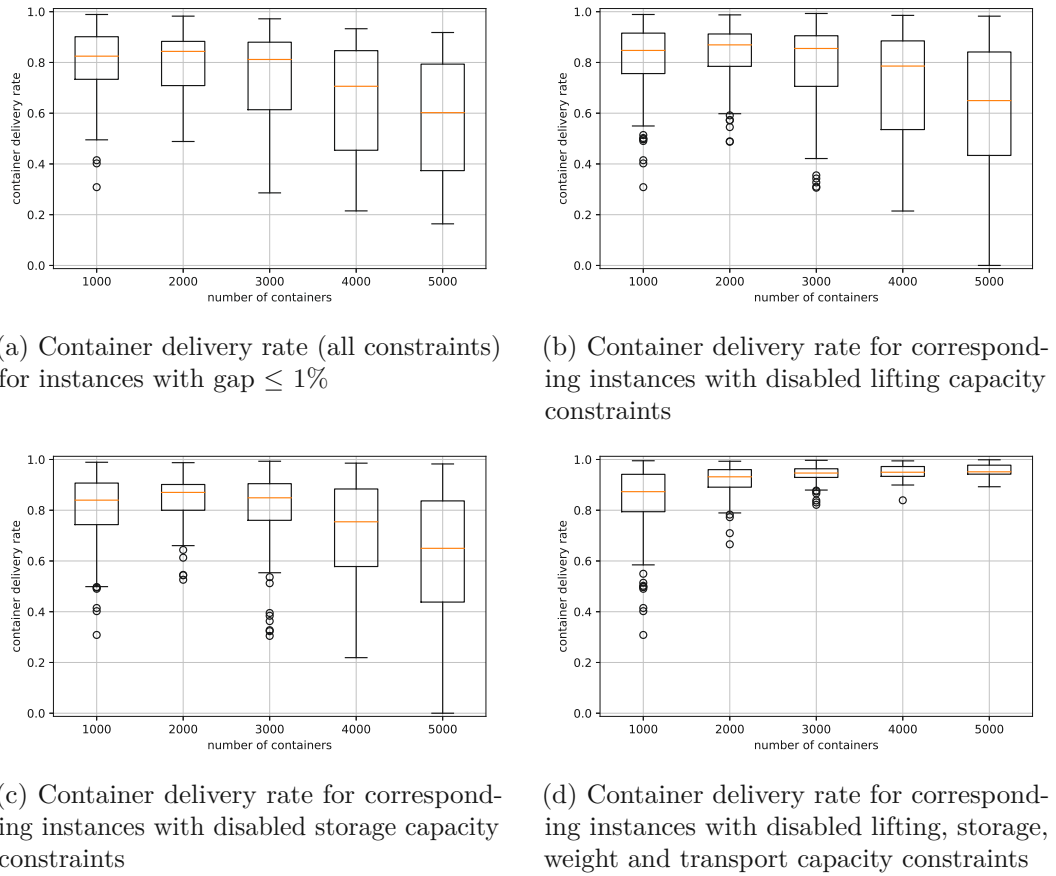


Figure 5.27: Container delivery rates for the single-train instances with $\text{gap} \leq 1\%$, and the same instances with a set of disabled constraints.

instances run with disabled storage constraints. Disabling the storage or lifting capacity constraints did not result in large increases of delivery rates, although a slight increase is notable, meaning that a few more containers could be delivered overall. This indicates that the primary bottlenecks of our instances are the transport (and weight) capacity constraints, i.e., the number of scheduled trains (cf. daily-train instances).

This can be verified by considering Fig. 5.27d. Here, all constraints have been disabled, leading to a drastic increase in the container delivery rates. When disabling all constraints, the only reason for containers not to be delivered is that the container density is too low for round trips to pay off.

Similar investigations can be done on individual instances to identify their bottlenecks. For example, when looking at our exemplary instance from Section 5.2 with disabled storage constraints, it would make sense to increase the storage capacity of terminals MUN and VIE. This can be seen in Fig. 5.28, showing the utilization of storage capacity over time. The dashed lines represents the terminal's current maximum storage capacity, which is exceeded for MUN and VIE, but not for AWP.

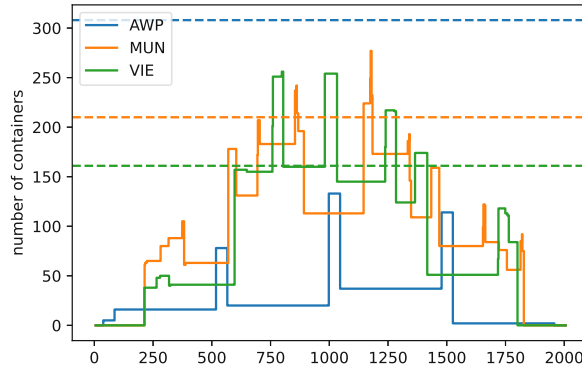


Figure 5.28: Number of stored containers for the three terminals AWP, MUN and VIE of our exemplary instance, run with disabled storage capacity constraints, plotted over time. The dashed lines are the terminal's nominal storage capacities.

Increasing the storage capacities lead to an increase of the container delivery rate from 0.723 to 0.738. At the same time, the storage capacity utilization increases from 0.258 to 0.297. The transport capacity utilization increases from 0.659 to 0.698, and the active round trip rate even drops from 0.659 to 0.724. This means that we can omit an entire round trip (and save its associated costs), while still increasing the number of delivered containers and the overall collected revenue.

CHAPTER 6

Conclusion

In this study, we have investigated the MCRP, a new problem that arises in the context of collaborative freight logistics. We optimize the shipment of containers and the use of trains from the point of view of a freight forwarder within a rail network. Instead of using direct trains, which is the current practice, trains are operated with potentially recurring round trips, multiple stops, and fixed schedules. This allows us to optimize container paths from a global point of view. We can try to avoid underutilized runs (round trips) of trains by shifting containers to other trains. This saves costs for the freight forwarder, which are then able to provide better prices for the customers as well.

After an extensive literature review in Chapter 2, we formally described the problem and clarified its context in Chapter 3. We used a time-expanded network to represent the problem and proposed an ILP model to solve it. Furthermore, we proved the complexity of our problem to be NP-complete w.r.t. its decision variant.

We created artificial instances for benchmarking the problem, described in detail in Chapter 4. These instances are based on a rail network that connects a subset of ten currently relevant cities for rail freight transport. To obtain the network, we assumed a hub-and-spoke architecture and solve a small ILP to minimize the length of the connecting tracks. We identified suitable realistic instance parameters through further literature review and feedback from our project partners, and provided sources for their respective values.

We discussed the results of applying our algorithm to these instances in Chapter 5. There, we analysed the impact of different input parameters on the hardness of our problem, as well as their influence on the characteristics of our solutions. To provide insights into the composition of individual solutions, we also investigated the results of a single exemplary instance in detail.

Finally, we summarized the following key results of our analysis and discussed managerial insight in Section 5.4: Beside the size of the instances, an appropriate choice of intermediately-sized container buffer time windows appears so be important for both the hardness of our problem and customer satisfaction. High container densities are

necessary to obtain high train utilizations. And using trains as operated in the MCRP appears to be promising for a variety of larger instances. Nevertheless, without further investigations and the use of real-world data, it still remains an open question whether the use of freight trains with recurring round trips, multiple stops, and fixed schedules would be beneficial in practice.

6.1 Future Work

In order to answer the aforementioned question, further research is required and some practical aspects will have to be addressed.

For example, we did not specify what happens to trains during *inactive* round trips. The alternative use of these trains has to be further investigated.

Another practical issue is the low priority of freight trains within rail networks. It potentially causes lateness that propagates to subsequent trains in case they have to wait on delayed ones. Stochastic or robust modeling would allow for the consideration of this issue.

Additionally, further research is required to compare the state of the art of directly operated freight trains with our investigated method. The required train rate that we used in this study is just one of the possibilities. In the best case, we would be able to compare our method to actual real-world data, but then it is still unclear how the corresponding instances of our problem would look like. Specifically, the construction of the individual train lines and their respective round trip schedules have to be further investigated for real-world application.

We have seen that in our instances, fewer containers are present at the start and end of the considered time horizon. It could make sense to consider more evenly distributed containers over time. Furthermore, in future analyses regarding the performance and other solution characteristics, more insights could be gained by investigating instances grouped by equal container density w.r.t. terminals, and over time. Also, the consideration of varying capacities over time could be of interest to account for resources used by other parties. Nevertheless, future investigations should preferably be based on real-world data as much as possible.

Our investigations have shown that high container densities lead to better train utilizations. Mindful of this and the ever-increasing demand for freight transportation, we believe that larger instances than the ones we considered in this study should be studied in future works.

We have seen in Section 5.4 that when considering the required train rate, trains operated as in the MCRP appear to potentially outperform the current practice of using directly operated trains, especially for larger types of our instances. Nevertheless, the difficulty of solving our problem grows noticeably with increasing size. This makes the application of (meta-)heuristics for this problem especially interesting. Greedy heuristics

appear to be particularly attractive due to the fact that in many of the optimal solutions we found, most containers are shipped on their shortest paths w.r.t. their shipping costs.

In this study, we have investigated train schedules with fixed scheduled round trips. In future work it might be interesting to simultaneously optimize round trip times and the container paths.

Furthermore, the use of a more sophisticated pricing model for the storage costs might be of interest. While the instances we have used assumed constant costs per terminal, current practice often uses pricing models where the first day is free of charge, and then the costs gradually increase, see Section 4.2.1.

Allowing different container sizes, as well as multiple origin and destinations terminals for each container (with varying additional cost to account for first and last mile delivery), would provide additional flexibility to the platform's operator and customers. Finally, we might want to consider alternative multi- and intermodal routes for each rejected container.

Appendix

Table 1: The summed up properties of all terminals within each considered city and their corresponding derived terminal sizes. Columns 2-7 contain the accumulated property information of all terminals, e.g., column “Tracks” contains the total number of loading tracks of all terminals of that city. The columns labeled “Size” are the respective sizes using one of these property columns. For our study, we use relative sizes based on the column “Size (Track Length)”.

City	Tracks	Track Length [m]	Gantry Cranes	Reach Stackers	Total Lifts per Hour	Interim Storage [TEU]	Size (Tracks)	Size (Track Length)	Size (Lifting Devices)	Size (Lifting per Hour)
Wien	11	7602	4	14	460	5260	0.23	0.23	0.53	0.57
Salzburg	8	3800	2	15	285	3600	0.17	0.12	0.50	0.35
Enns	4	3000	1	4	90	5000	0.09	0.09	0.15	0.11
Berlin	10	4420	4	5	195	2300	0.21	0.11	0.26	0.24
Duisburg	38	21520	13	21	735	22700	0.81	0.66	1.00	0.91
München	14	9800	8	2	270	1000	0.30	0.30	0.29	0.33
Bremerhaven	6	4590	4	0	120	0	0.13	0.14	0.12	0.15
Hamburg	47	32800	26	2	810	4400	1.00	1.00	0.82	1.00
Antwerpen	23	14550	10	17	565	8200	0.49	0.44	0.79	0.70
Rotterdam	39	25700	16	11	600	3381	0.83	0.78	0.79	0.74

Table 2: Information based on [39] of all relevant terminals.

City	Terminal Name	Tracks	Total Track Length [m]	Gantry Cranes	Max Weight [t]	Lifting Operations per Hour	Reach- stackers	Max Weight [t]	Lifting Operations per Hour	Total Lifting Operations per Hour	Interim Storage [TEU]	Depot Capacity [TEU]
Wien	Wien Cont	7	4802	2	45	25	6	45	25	200	2000	5000
Wien	Wien						8	12	25	200		
Wien	Wien Süd CCT	4	2800	2	41	30				60	3260	
Salzburg	Salzburg Frachtenbahnhof - ROLA	2	800							0		
Salzburg	Salzburg CTS	6	3000	2	41	30	15	10	15	285	3600	
Enns	Enns Hafen CCT	4	3000	1	50	30	4	45	15	90	5000	
Berlin	Multimodal Terminal Berlin	2	1220				2	45	15	30	500	
Berlin	Berlin Westhafen	2	700	2	45	30	1		15	75	1200	
Berlin	Großbeeren	4	2100	1	41	30	1	35	15	45	600	600
Berlin	Metracons Containerterminal Berlin	2	400	1		30	1		15	45		
Duisburg	Duisburg KV-Hub Rhein-Ruhr	4	2840	2	40	30				60		
Duisburg	DeCe Te Duisburg	2	1400				5	42	15	75		
Duisburg	Duisburg						5	8	15	75		
Duisburg	Duisburg RRT Home Terminal	4	1240	2	50	30	4	42	15	120	8000	
Duisburg	Duisburg						1	17	15	15		
Duisburg	Duisburg Trinodal Terminal (D3T)	4	1400	1		30				30	1800	
Duisburg	Duisburg Kombiterminal (DKT)	6	2820	2		30				60	1800	
Duisburg	Duisburg DIT	6	4500	2	50	30	1		15	75	7500	
Duisburg	Duisburg RRT Gateway West	4	1400	2	50	30	3	42	15	105	3600	
Duisburg	Duisburg logport III (Hohenbudberg)	8	5920	2		30	2		30	120		
München	München-Riem	14	9800	8	41	30	2	41	15	270	1000	
Bremerhaven	Bremerhaven - Eurogate C/TB	6	4590	4		30				120		
Hamburg	Hamburg - Eurokombi Waltershof	11	7840	8	41	30	2	41	15	270	3000	
Hamburg	Hamburg - Tollerort (CTT)	5	3600	3		30				90		
Hamburg	Hamburg - Burchardkai (CTB)	10	7400	4	40	30				120		
Hamburg	Hamburg - Altenwerder (CTA)	9	6300	4	43	30				120		
Hamburg	Hamburg - Billwerder	12	7660	7	41	30				210	1400	
Antwerpen	Antwerpen Combinant	5	3100	2	44	25				50	1200	
Antwerpen	Antwerpen Cirkeldyk	4	2600	2	40	30	4	35	15	120		
Antwerpen	Antwerpen						1	30	15	15		
Antwerpen	Antwerpen Main-Hub	8	5600	3	40	30	4		15	150		
Antwerpen	HTA Hupac Terminal Antwerp	5	3100	3	40	30	3	35	15	135	7000	
Antwerpen	Antwerpen ATO	1	150				3	45	15	45		
Antwerpen	Antwerpen						2	10	25	50		
Rotterdam	Rotterdam Euromax	6	4800	3		15				45		
Rotterdam	Rotterdam APM Terminals Maasvlakte II	4	3000	2		30				60		
Rotterdam	Rotterdam ECT Delta Terminal	4	2800	2	40	30	2	40	15	90		
Rotterdam	Rotterdam RWG	6	4500	2		30				60		
Rotterdam	Rotterdam Botlek	3	1200	1		30				30		
Rotterdam	Rotterdam Combi Terminal Twente	8	3400	2		30	3	45	15	105	1800	

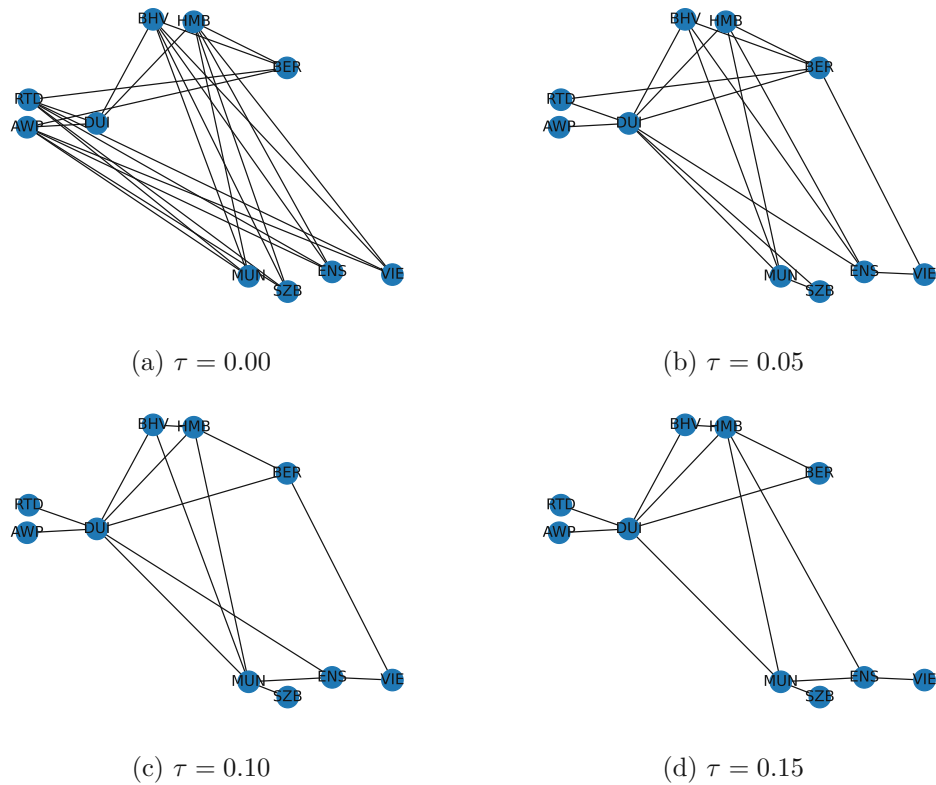


Figure 1: Results of the ILP from Section 4.1.2 generating our real-world rail network for $\tau \in \{0, 0.05, 0.10, 0.15\}$. For our study, we use $\tau = 0.10$

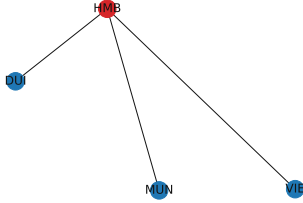
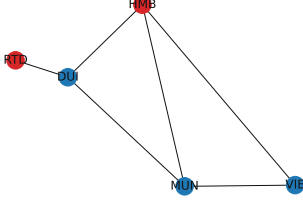
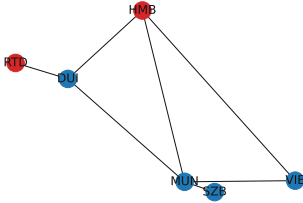
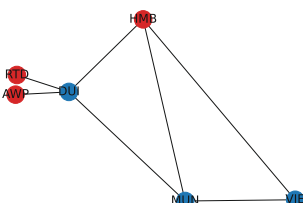
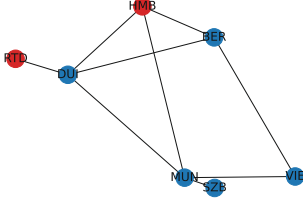
Table 3: Rail networks and train lines of the associated instances with different numbers of ports $|P|$ and cities $|C|$. Here, column “ $|L|$ ” indicates the number of train *lines* (and not the number of individual trains¹).

(P , C)	$ L $	Rail Network	Train Lines
(1,2)	2		<ul style="list-style-type: none"> • HMB-DUI • HMB-MUN

Continued on the next page

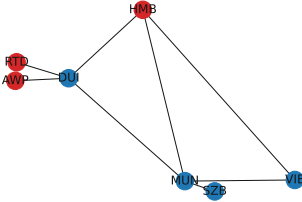
¹Nevertheless, the number of trains and train lines coincides for the single-train instances.

Table 3 – continued from previous page

(P , C)	$ L $	Rail Network	Train Lines
(1,3)	3		<ul style="list-style-type: none"> • HMB-DUI • HMB-MUN • HMB-VIE
(2,3)	4		<ul style="list-style-type: none"> • HMB-DUI • HMB-MUN • HMB-VIE • RTD-DUI-MUN-VIE
(2,4)	5		<ul style="list-style-type: none"> • HMB-DUI • HMB-MUN-SZB • HMB-VIE • RTD-DUI-MUN-VIE • RTD-DUI-MUN-SZB
(3,3)	5		<ul style="list-style-type: none"> • HMB-DUI • HMB-MUN • HMB-VIE • RTD-DUI-MUN-VIE • AWP-DUI-MUN-VIE
(2,5)	6		<ul style="list-style-type: none"> • HMB-DUI • HMB-MUN-SZB • HMB-BER-VIE • RTD-DUI-MUN-VIE • RTD-DUI-MUN-SZB • RTD-DUI-BER

Continued on the next page

Table 3 – continued from previous page

(P , C)	$ L $	Rail Network	Train Lines
(3,4)	7		<ul style="list-style-type: none">• HMB-DUI• HMB-MUN-SZB• HMB-VIE• RTD-DUI-MUN-VIE• RTD-DUI-MUN-SZB• AWP-DUI-MUN-VIE• AWP-DUI-MUN-SZB

Bibliography

- [1] RK Ahuja, TL Magnanti, and JB Orlin. *Network Flows*, volume 10. Prentice hall, 1993.
- [2] Allianz pro Schiene. Die bahn ist das umweltfreundlichste motorisierte verkehrsmittel. <https://www.allianz-pro-schiene.de/themen/umwelt/>. Accessed: 2021-10-06.
- [3] Allianz pro Schiene. Überblick: Wie der güterzug länger werden kann. <https://www.allianz-pro-schiene.de/themen/aktuell/740-meter-gueterzug/>, 2016. Accessed: 2021-10-06.
- [4] Claudia Archetti and Lorenzo Peirano. Air intermodal freight transportation: The freight forwarder service problem. *Omega*, 94:102040, 2020. ISSN 0305-0483. doi: <https://doi.org/10.1016/j.omega.2019.02.009>. URL <https://www.sciencedirect.com/science/article/pii/S0305048318304316>.
- [5] Paul Gustav Heinrich Bachmann. *Die analytische Zahlentheorie. Dargestellt von Paul Bachmann*. Leipzig B.G. Teubner, 1894.
- [6] Eric Ballot, Benoit Montreuil, and Russell Meller. *The physical internet*. La Documentation Française, 2014.
- [7] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1): 87–90, 1958.
- [8] Anne Benoit, Yves Robert, and Frédéric Vivien. *A guide to algorithm design: paradigms, methods, and complexity analysis*. CRC Press, 2013.
- [9] Anthony K. C. Beresford. Modelling freight transport costs: A case study of the uk-greece corridor. *International Journal of Logistics Research and Applications*, 2 (3):229–246, 1999. doi: 10.1080/13675569908901583. URL <https://doi.org/10.1080/13675569908901583>.
- [10] Ian Black, Roger Seaton, Andrea Ricci, and Riccardo Enei. Final report: actions to promote intermodal transport. *RECORDIT Final Report. Cranfield University, Institute of Studies for the Integration of System, Cranfield, Rome*, 2003.

- [11] Nils Boysen, Malte Fliedner, Florian Jaehn, and Erwin Pesch. Shunting yard operations: Theoretical aspects and applications. *European Journal of Operational Research*, 220(1):1–14, 2012. ISSN 0377-2217. doi: <https://doi.org/10.1016/j.ejor.2012.01.043>. URL <https://www.sciencedirect.com/science/article/pii/S0377221712000811>.
- [12] Tsung-Sheng Chang. Best routes selection in international intermodal networks. *Computers & Operations Research*, 35(9):2877–2891, 2008. ISSN 0305-0548. doi: <https://doi.org/10.1016/j.cor.2006.12.025>. URL <https://www.sciencedirect.com/science/article/pii/S0305054806003273>. Part Special Issue: Bio-inspired Methods in Combinatorial Optimization.
- [13] Ping Chen, Yunsong Guo, Andrew Lim, and Brian Rodrigues. Multiple crossdocks with inventory and time windows. *Comput. Oper. Res.*, 33:43–63, 2006.
- [14] Mateusz Cicheński, Florian Jaehn, Grzegorz Pawlak, Erwin Pesch, Gaurav Singh, and Jacek Blazewicz. An integrated model for the transshipment yard scheduling problem. *Journal of Scheduling*, 20, 02 2017. doi: 10.1007/s10951-016-0470-4.
- [15] CMA CGM. Tariff inland terminals import storage costs 2020. <https://www.cma-cgm.com/static/DE/attachments/LC%20-%20Inland%20Terminal%20Storage%20Charges%20Import%202020%20V1.3b.pdf>, 2021. Accessed: 2021-11-02.
- [16] MT Container. Container-transport – optionen im Überblick. <https://www.mtcontainer.de/container-service/transport/>. Accessed: 2022-04-27.
- [17] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery. ISBN 9781450374644. doi: 10.1145/800157.805047. URL <https://doi.org/10.1145/800157.805047>.
- [18] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
- [19] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*, chapter Lower Bounds Based on the Exponential-Time Hypothesis, pages 467–521. Springer International Publishing, Cham, 2015. ISBN 978-3-319-21275-3. URL https://doi.org/10.1007/978-3-319-21275-3_14.
- [20] George Dantzig. *Linear Programming and Extensions*. Princeton University Press, 2016. ISBN 9781400884179. doi: 10.1515/9781400884179. URL <https://doi.org/10.1515/9781400884179>.

- [21] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
- [22] European Commission. Real cost reduction of door-to-door intermodal transport – recordit, 2001. Directorate General DG VII, RTD 5th Framework Programme, Brussels, Belgium.
- [23] European Court of Auditors. Rail freight transport in the eu: still not on the right track. Technical Report 08, European Court of Auditors, 12, rue Alcide De Gasperi, 1615 Luxembourg, 2016. Special Report.
- [24] Eurostat. Key figures on europe – 2021 edition. <https://ec.europa.eu/eurostat/documents/3217494/13394938/KS-EI-21-001-EN-N.pdf/ad9053c2-debd-68c0-2167-f2646efeaec1>, 2021. Accessed: 2021-11-05.
- [25] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6): 345, 1962.
- [26] Jr Ford, L. R. and D. R Fulkerson. Solving the transportation problem. *Management science*, 3(1):24–32, 1956. ISSN 0025-1909.
- [27] Lester Randolph Ford and Delbert Ray Fulkerson. Flows in networks. In *Flows in Networks*. Princeton university press, 2015.
- [28] Lance Fortnow. The status of the p versus np problem. *Commun. ACM*, 52(9): 78–86, sep 2009. ISSN 0001-0782. doi: 10.1145/1562164.1562186. URL <https://doi.org/10.1145/1562164.1562186>.
- [29] Fraunhofer Austria Research GmbH. Physical project. <https://physical-project.at/>. Accessed: 2021-10-04.
- [30] Armin Fügenschuh, Henning Homfeld, and Hanno Schülldorf. Single-car routing in rail freight transport. *Transp. Sci.*, 49:130–148, 2015.
- [31] William Gasarch. The $p=?np$ poll. *SIGACT News*, 33:34–47, 01 2002.
- [32] Igor Grebennik, Remy Dupas, Oleksandr Lytvynenko, and Inna Urniaieva. Scheduling freight trains in rail-rail transshipment yards with train arrangements. *International Journal of Intelligent Systems and Applications*, 9:12–19, 10 2017. doi: 10.5815/ijisa.2017.10.02.
- [33] Rail Cargo Group. Lgjnss. <https://www.railcargo.com/de/dam/jcr:e3895fb3-8adf-4c02-aa6a-7c27538a769c/lgjnss-fs.pdf>. Accessed: 2022-04-27.
- [34] G. Guastaroba, M. G. Speranza, and D. Vigo. Intermediate facilities in freight transportation planning: A survey. *Transportation Science*, 50(3):763–789, August 2016. ISSN 1526-5447. doi: 10.1287/trsc.2015.0631. URL <https://doi.org/10.1287/trsc.2015.0631>.

- [35] Frank L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Physics*, 20(1-4):224–230, 1941. doi: <https://doi.org/10.1002/sapm1941201224>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm1941201224>.
- [36] IBM. Ibm ilog cplex optimizer. <https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer>. Accessed: 2022-11-30.
- [37] iContainers. 20-foot container – dimensions, measurements and weight. <https://www.icontainers.com/help/20-foot-container/>. Accessed: 2021-10-29.
- [38] R. Impagliazzo and R. Paturi. Complexity of k-sat. In *Proceedings. Fourteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference) (Cat.No.99CB36317)*, pages 237–240, May 1999. doi: 10.1109/CCC.1999.766282.
- [39] AGORA intermodal terminals.eu. Intermodal terminals in europe. <https://www.intermodal-terminals.eu/database>. Accessed: 2022-03-09.
- [40] Milan Janic. Modelling the full costs of an intermodal and road freight transport network. *Transportation Research Part D: Transport and Environment*, 12(1):33–44, 2007. ISSN 1361-9209. doi: <https://doi.org/10.1016/j.trd.2006.10.004>. URL <https://www.sciencedirect.com/science/article/pii/S1361920906000794>.
- [41] Krishna C. Jha, Ravindra K. Ahuja, and Güvenç Şahin. New approaches for solving the block-to-train assignment problem. *Networks*, 51(1):48–62, 2008. doi: <https://doi.org/10.1002/net.20195>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20195>.
- [42] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing, STOC '84*, page 302–311, New York, NY, USA, 1984. Association for Computing Machinery. ISBN 0897911334. doi: 10.1145/800057.808695. URL <https://doi.org/10.1145/800057.808695>.
- [43] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.
- [44] Leonid Khachiyan. Polynomial algorithms in linear programming. *Ussr Computational Mathematics and Mathematical Physics*, 20:53–72, 1980.
- [45] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. ISSN 00029939, 10886826. URL <http://www.jstor.org/stable/2033241>.

- [46] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. ISSN 00129682, 14680262. URL <http://www.jstor.org/stable/1910129>.
- [47] Edmund Landau. *Handbuch der Lehre von der Verteilung der Primzahlen*. Leipzig B.G. Teubner, 1909.
- [48] Leonid A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3), 1973.
- [49] Andrew Lim, Zhaowei Miao, Brian Rodrigues, and Zhou Xu. Transshipment through crossdocks with inventory and time windows. *Naval Research Logistics Quarterly*, 52(8):724–733, 12 2005. ISSN 0894-069X. doi: 10.1002/nav.20113.
- [50] Reza Mohammad Hasany and Yousef Shafahi. Two-stage stochastic programming for the railroad blocking problem with uncertain demand and supply resources. *Computers & Industrial Engineering*, 106:275–286, 2017. ISSN 0360-8352. doi: <https://doi.org/10.1016/j.cie.2017.02.014>. URL <https://www.sciencedirect.com/science/article/pii/S036083521730075X>.
- [51] S. Moll, U. Weidmann, and W. Stölzle. *Productivity Improvements for Freight Railways Through Collaborative Transport Planning*. Institute for Transport Planning and Systems. ETH, 2012. URL <https://books.google.at/books?id=DPmEwgEACAAJ>.
- [52] Edward F Moore. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*, pages 285–292, 1959.
- [53] Alex Orden. The transshipment problem. *Management science journal of the Institute for Operations Research and the Management Sciences*, 2(3):276–285, 1956. ISSN 0025-1909.
- [54] Matthias Prandtstetter, Georg Brandstätter, and Lukas F. Krasel. Optimising container transports in collaborative roundtrip scenarios. In *Proceedings of the 8th International Physical Internet Conference*, pages 123–128, 2021.
- [55] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957. doi: 10.1002/j.1538-7305.1957.tb01515.x.
- [56] Jean-Paul Rodrigue and Theo Notteboom. Comparative north american and european gateway logistics: the regionalism of freight distribution. *Journal of Transport Geography*, 18(4):497–507, 2010. ISSN 0966-6923. doi: <https://doi.org/10.1016/j.jtrangeo.2010.03.006>. URL <https://www.sciencedirect.com/science/article/pii/S0966692310000384>. Special Issue on Comparative North American and European gateway logistics.
- [57] Axel Schönknecht. *Maritime Containerlogistik: Leistungsvergleich von Containerschiffen in intermodalen Transportketten*. Springer-Verlag, 2009.

- [58] René Schönemann. *Scheduling rail freight node operations through a slot allocation approach*. Doctoral thesis, Technische Universität Berlin, Berlin, 2016. URL <http://dx.doi.org/10.14279/depositonce-5452>.
- [59] Huirong Shi and Yang Chen. Capacity analysis of rail container freight shuttle train which factors are important for running this kind of train? *Master Thesis No 2003 5*, 2004.
- [60] James A Storer. On the complexity of chess. *Journal of computer and system sciences*, 27(1):77–100, 1983.
- [61] J. A Tomlin. Minimum-cost multicommodity network flows. *Operations research*, 14(1):45–51, 1966. ISSN 0030-364X.
- [62] UNECE. Recommendations on organizing demonstration runs of container block trains on euro-asian transport links. <https://digitallibrary.un.org/record/584890>, 2006. Accessed: 2021-10-27.
- [63] VCÖ. Güterverkehr auf klimakurs bringen. *Mobilität mit Zukunft*, 2020.
- [64] WienCont Container Terminal GmbH. Tariffs 2018. <https://www.wienholding.at/tools/uploads/STANDARD-TARIF-2018kompakt-engl.pdf>, 2018. Accessed: 2021-10-29, Address: Freudenuer Hafenstrasse 8-10, 1020 Wien, a company of Port of Vienna.
- [65] Laurence A Wolsey. Integer programming john wiley & sons. *New York, NY*, 4, 1998.
- [66] Jie Xiao and Boliang Lin. Comprehensive optimization of the one-block and two-block train formation plan. *Journal of Rail Transport Planning & Management*, 6(3):218–236, 2016. ISSN 2210-9706. doi: <https://doi.org/10.1016/j.jrtpm.2016.09.002>. URL <https://www.sciencedirect.com/science/article/pii/S221097061630049X>.
- [67] Endong Zhu, Teodor Gabriel Crainic, and Michel Gendreau. Scheduled service network design for freight rail transportation. *Oper. Res.*, 62:383–400, 2014.