IT UNIVERSITY OF COPENHAGEN

Section of Data Science
Machine Learning Research Group

PHD THESIS

Learning and Combinatorial Optimization for Efficient Container Vessel Stowage Planning

Author:
Jaike van Twiller

Supervisor:
Rune Møller Jensen

Co-Supervisor:
Djordje Grbic

Submitted on
April 7, 2025

Imprint

Project:	PhD Thesis
Title:	Learning and Combinatorial Optimization for Efficient Container Vessel Stowage Planning
Author:	Jaike van Twiller
Abstract Translation:	Rune Møller Jensen
Date:	April 7, 2025
Copyright:	IT University of Copenhagen

Supervisor:
Rune Møller Jensen
IT University of Copenhagen
Email: rmj@itu.dk

Co-Supervisor:
Djordje Grbic
IT University of Copenhagen
Email: djgr@itu.dk

Acknowledgements

This thesis marks the end of a long journey, and there are many people without whom I wouldn't have made it here.

First and foremost, I would like to express my deepest gratitude to my supervisor, Rune Møller Jensen, for his invaluable guidance and encouragement throughout this research journey. His expertise in stowage planning and combinatorial optimization, along with his constructive and insightful feedback, has been instrumental in shaping this thesis.

I am also sincerely grateful to my co-supervisor, Djordje Grbic, who guided me through the world of machine learning and was consistently generous with his time — always open to questions, thoughtful discussions and taking the time for feedback.

My heartfelt thanks go to the members of my PhD committee—Andrzej Wąsowski, Trine Krogh Boomsma, and Emma Frejinger—for your time and dedication, critical insights, and thoughtful feedback throughout this process.

I would also like to acknowledge my mentors during my research stay at HEC Montréal, Erick Delage and Yossiri Adulyasak. Your hospitality expertise in data-driven decision-making and all the insightful discussions made the visit a valuable and enriching part of my PhD experience.

Additionally, a big thanks to all collaborators on various articles for your sharp ideas and enjoyable teamwork. Especially to Aga, with whom I shared the majority of this PhD journey—thank you for all the good talks and mutual support.

Special thanks to my colleagues in the Data Science section and at ITU for the stimulating discussions, warm welcome, and all the enjoyable moments we shared.

To my fellow PhD Club committee members—Laura, Kasper, Kai, Meisam, and Camilla—thank you for fostering such a supportive and motivating environment. Our efforts in building a sense of community among PhD students has left a lasting impact on my experience.

Furthermore, I am truly grateful to my friends for your unwavering support, humor, and help in escaping the world of academia when needed. In particular, Frans and Michel, for your feedback, insights, and perspective—your presence has been a constant source of strength.

Last but certainly not least, I am deeply thankful to my family and my significant other, Anna, for your unconditional love, patience, and belief in me. Your support has been the foundation upon which I've built this journey.

Abstract

Container shipping plays an essential role in the global transportation of internationally traded goods, making it a crucial component of the world economy. Due to the large volume of cargo moved by each vessel, container shipping is also one of the most environmentally friendly modes of transport, resulting in significantly lower emissions per tonne of cargo per kilometer compared to alternatives.

A key operational challenge in container shipping is deciding how to efficiently place containers onto vessels, a task known as stowage planning. This task is critical yet challenging, involving many factors and constraints that interact in combinationally difficult ways. Due to its complexity, stowage planning is usually split into two phases: (1) the master planning problem (MPP), which broadly determines how containers are grouped and placed onboard, and (2) the slot planning problem (SPP), which assigns individual containers to specific slots.

This PhD research explores how advanced techniques from combinatorial optimization (CO) and machine learning (ML) can accelerate decision-making for stowage planning, especially at the master planning level. The goal is to develop efficient and practical solutions that bridge the gap between theoretical models and real-world industry needs, leading to faster and better planning decisions that result in reliable and efficient supply chains with implications for global trade and environmental sustainability.

The research comprises four main contributions: (1) a comprehensive literature review, (2) novel problem formulations of the MPP, (3) scalable CO and ML-based solutions methods, (4) theoretical analysis on computational complexity and mathematical soundness.

First, a literature review classifies the single-port and multi-port container stowage planning problem (CSPP), highlighting key issues such as oversimplified problem formulations and limited industrial validation. A research agenda is proposed to address challenges, such as the need for scalable algorithms to solve realistic problem definitions on benchmark instances, with particular attention to the MPP.

Second, building on these insights, novel problem formulations are provided in the form of a 0-1 integer program (IP) model that searches in the space of valid paired block stowage and a Markov decision process and its extension that both decompose the decision process into sequential steps. Furthermore, this thesis includes paired block stowage patterns and demand uncertainty in the MPP, which are features to consider in the MPP.

Third, the findings indicate that the 0-1 IP model outperforms a traditional mixed-integer programming (MIP) model in terms of optimality and runtime. Regardless, larger problem instances require more than 10 minutes to solve, which is considered

intractable given the dynamic nature of stowage planning. In contrast, the MDPs addressed by deep reinforcement learning (DRL) can construct solutions for the MPP within this timeframe. However, the MDPs do not offer guarantees on optimality and feasibility, which need to be learned through extensive training. Especially on feasibility, it is shown that differentiable projection layers are needed to ensure feasibility, while alternatives as reward scaling and feasibility regularization can work but are hard to balance with other objectives. In the case of specific non-convex constraints, action masking in combination with feasibility projection can be applied.

Fourth, this thesis shows that searching in the space of valid block stowage pattern is an NP-hard task but also demonstrates how a differentiable projection layer based on violation gradients can minimize the violations of convex inequality constraints.

This research advances both theory and practice in stowage planning by introducing scalable optimization techniques. It highlights the value of improved problem formulations and learning-based heuristics for real-world planning problems. These contributions show how decision-support systems can be enhanced, paving the way for more resilient and efficient container shipping.

Resumé

Containershipping spiller en væsentlig rolle i den globale transport af handelsvarer og er dermed en afgørende faktor i verdensøkonomien. På grund af den store mængde gods, der transporteres af hvert skib, er containershipping også en af de mest miljøvenlige transportformer med betydeligt lavere emissioner pr. ton gods pr. kilometer end alternativer.

En vigtig driftsudfordring inden for containershipping er at beslutte, hvordan containere effektivt placeres på skibe. Dette kaldes stuvningsplanlægning. Stuvningsplanlægning er et kritisk og vanskeligt optimeringsproblem. Det involverer mange faktorer og begrænsninger, der interagerer kombinatorisk. På grund af dets kompleksitet er stuvningsplanlægning normalt opdelt i to faser: (1) the master planning problem (MPP), som i vid udstrækning bestemmer, hvordan containere grupperes og placeres ombord, og (2) the slot planning problem (SPP), som tildeler individuelle containere til bestemte positioner på skibet.

Denne ph.d.-afhandling undersøger, hvordan avancerede metoder inden for kombinatorisk optimering (CO) og machine learning (ML) kan anvendes til stuvningsplanlægning, især i forhold til master planning. Målet er at udvikle effektive og praktiske løsninger, der bygger bro mellem teoretiske modeller og industriens behov for hermed at opnå hurtigere og bedre planlægningsbeslutninger, der resulterer i mere effektive og bæredygtige forsyningskæder i den globale handel.

Forskningen har fire hovedbidrag: (1) en omfattende litteraturgennemgang, (2) nye problemformuleringer af MPP, (3) skalerbare CO- og ML-baserede løsningsmetoder, (4) teoretisk analyse af beregningsmæssig kompleksitet og matematisk korrekthed.

Med hensyn til (1) klassificerer en litteraturgennemgang stuvningsproblemet i forhold til stuvning i en enkelt og flere havne (CSPP) og fremhæver problemstillinger såsom overforenklede problemformuleringer og begrænset industriel validering. Der foreslås en forskningsagenda for at tackle udfordringer såsom behovet for skalerbare algoritmer til at løse realistiske MPP instanser.

På denne baggrund bidrager (2) med nye problemformuleringer i form af et 0-1 heltalsprogram (IP), der søger i et rum af gyldige paired block stuvninger, og en Markov-beslutningsproces (MDP) og dens udvidelse, der begge nedbryder beslutningsprocessen i sekventielle trin. Desuden inkluderes paired block stuvningsmønstre og efterspørgselsusikkerhed i MPP'en.

Med hensyn til (3) indikerer resultaterne, at 0-1 IP-modellen overgår en traditionel MIP-model (Mixed-Integer Programming) med hensyn til optimalitet og kørselstid. Uanset hvad kræver større probleminstanser mere end 10 minutter at løse, hvilket er lang tid for mange praktiske anvendelser. I modsætning hertil kan MDP'erne løses a deep reinforcement learning (DRL) inden for denne tidsramme. DRL metoderne

garanterer dog ikke optimalitet og korrekthed. Især med hensyn til korrekthed er det nødvendigt at anvende differentierbare projektionslag, mens alternativer som belønningsskalering og korrekthedsregulering kan fungere, men er svære at balancere i forhold til andre mål. I tilfælde af specifikke ikke-konvekse begrænsninger kan der anvendes maskering i kombination med projektion.

Med hensyn til (4) viser denne afhandling, at søgning i et rum med gyldig block stuvning er NP-hårdt. Et differentierbart projektionslag baseret på overtrædelsesgradienter kan dog minimere overtrædelserne af konvekse ulighedsbegrænsninger.

Denne afhandling fremmer både teori og praksis inden for stuvningsplanlægning ved at introducere skalerbare optimeringsteknikker. Den fremhæver værdien af forbedrede problemformuleringer og læringsbaserede heuristikker for planlægningsproblemer i den virkelige verden. Disse bidrag viser, hvordan beslutningsstøttesystemer kan forbedres og bane vejen for en mere modstandsdygtig og effektiv containertransport.

Samenvatting

Containervervoer speelt een essentiële rol in het wereldwijde transport van internationaal verhandelde goederen en vormt daarmee een cruciale schakel in de wereldeconomie. Door de grote hoeveelheden vracht die per schip vervoerd worden, is het ook een van de meest milieuvriendelijke transportmethoden, met aanzienlijk lagere emissies per ton vracht per kilometer dan alternatieve vervoerswijzen.

Een belangrijke operationele uitdaging in het containervervoer is het efficiënt plaatsen van containers aan boord van schepen, een taak die bekend staat als stuwageplanning. Deze taak is cruciaal maar uitdagend vanwege de vele factoren en restricties die op combinatorisch complexe wijze met elkaar interageren. Door deze complexiteit wordt stuwageplanning meestal opgesplitst in twee fasen: (1) het *meester planning probleem* (MPP), dat bepaalt hoe containers in grote lijnen worden gegroepeerd en geplaatst, en (2) het *stuwlocatie planning probleem* (SPP), dat individuele containers aan specifieke posities toewijst.

Dit promotieonderzoek onderzoekt hoe geavanceerde technieken uit combinatorische optimalisatie (CO) en machine learning (ML) het beslissingsproces voor stuwageplanning kunnen versnellen, met name op het niveau van masterplanning. Het doel is om efficiënte en praktische oplossingen te ontwikkelen die de kloof tussen theoretische modellen en werkelijke industriële behoeften overbruggen. Dit leidt tot snellere en betere planningsbeslissingen, met als resultaat betrouwbare en efficiënte toeleveringsketens met positieve gevolgen voor wereldhandel en duurzaamheid.

Het onderzoek bestaat uit vier hoofdonderdelen: (1) een uitgebreide literatuurstudie, (2) nieuwe probleemformuleringen van het MPP, (3) schaalbare oplossingsmethoden gebaseerd op CO en ML, en (4) theoretische analyses omtrent computationele complexiteit en wiskundige onderbouwing.

Ten eerste classificeert de literatuurstudie het enkel-haven en multi-haven container stuwage planning probleem (CSPP), en benoemt belangrijke knelpunten zoals te simplistische probleemformuleringen en beperkte industriële validatie. Een onderzoeksagenda wordt voorgesteld om uitdagingen aan te pakken, waaronder de behoefte aan schaalbare algoritmes die realistische probleemdefinities op referentie-instanties kunnen oplossen, met bijzondere aandacht voor het MPP.

Ten tweede worden, op basis van deze inzichten, nieuwe probleemformuleringen gepresenteerd in de vorm van een 0-1 geheeltallig programmeringsmodel (IP) dat zoekt in de ruimte van geldige stuwage in gekoppelde blokken, en een Markov beslissingsproces (MDP) en diens uitbreiding die het beslissingsproces opsplitst in sequentiële stappen. Verder bevat dit proefschrift stuwage patronen in gekoppelde blokken en vraagonzekerheid als elementen van het MPP.

Ten derde tonen de resultaten aan dat het 0-1 IP-model beter presteert dan een traditioneel gemengd geheelgetallig programmingsmodel (MIP) op het vlak van optimaliteit en rekentijd. Toch blijken grotere probleeminstanties meer dan 10 minuten nodig te hebben om op te lossen, wat als niet toelaatbaar wordt beschouwd gezien de dynamiek van stuwageplanning. Daarentegen kunnen de MDP's, opgelost via deep reinforcement learning (DRL), binnen deze tijdslimiet oplossingen genereren voor het MPP. Echter, MDP's bieden geen garanties op optimaliteit en haalbaarheid, wat via intensieve training moet worden geleerd. Met name voor haalbaarheid blijkt dat differentiabele projectielagen nodig zijn om deze te waarborgen, terwijl alternatieven zoals het schalen van belongen en het regulariseren van haalbaarheid ook kunnen werken, maar moeilijk in balans te brengen zijn met andere doelstellingen. In het geval van specifieke niet-convexe restricties kan actie-maskering in combinatie met haalbaarheidsprojectie worden toegepast.

Ten vierde toont dit proefschrift aan dat zoeken in de rruimte van geldige stuwagepatronen in gekoppelde blokken een NP-moeilijke taak is, maar ook hoe een differentiabele projectielaag gebaseerd op schedingen van gradiënten kan helpen bij het minimaliseren van schendingen van convexe haalbaarheid regios.

Dit onderzoek levert zowel theoretische als praktische bijdragen aan stuwageplanning door schaalbare optimalisatietechnieken te introduceren. Het onderstreept het belang van verbeterde probleemformuleringen en leergestuurde heuristieken voor realistische planningsproblemen. Deze bijdragen tonen aan hoe beslissingsondersteunende systemen kunnen worden versterkt, en effenen het pad naar een veerkrachtigere en efficiëntere containervaart.

Contents

List of Figures

List of Tables

List of Abbreviations

AC	Actor critic
AGV	Automated guided vehicle
AI	Artificial intelligence
AM	Attention model
AM-P	Projected attention model
B&B	Branch-and-bound
BM	Bending moment
BS	Block stowage
BW	Ballast water
CI	Crane intensity
CNN	Convolutional neural network
CO	Combinatorial optimization
CP	Constraint programming
CPU	Central Processing Unit
CSPP	Container stowage planning problem
CV	Coefficient of variation
DC	Dry containers
DG	Dangerous goods
DP	Dynamic programming
DRL	Deep reinforcement learning
DWT	Dead weight tonnage
FF	Feedforward layer
FFN	Feedforward network
FR	Feasibility regularization
ft	Feet
FT	Fuel tanks
GA	Genetic algorithm
GAE	Generalized advantage estimation
GPU	Graphics Processing Unit
GM	Metacentric height
HC	Highcubes
HO	Hatch overstowage
IMDG	International maritime dangerous goods
IMO	International Maritime Organization
in	Inches
IP	Integer programming
JSSP	Job shop scheduling problem
LCG	Longitudinal center of gravity
LNS	Large neighborhood search
LOS	Line of sight
LWT	Lightship weight
MAE	Mean absolute error

MBPP	Master bay planning problem
MC	Monte Carlo
MDP	Markov decision process
MHA	Multi-head attention
MIP	Mixed integer programming
ML	Machine learning
MLP	Multi-layer perceptron
MPP	Master planning problem
MSE	Mean squared error
NN	Neural network
NP	Nondeterministic polynomial-time
OOG	Out-of-gauge
OR	Operations research
P	Polynomial-time
PBS	Paired block stowage
PC	Policy clipping
PG	Policy gradients
POD	Port-of-discharge
POL	Port-of-load
PPO	Proximal policy optimization
RL	Reinforcement learning
RNN	Recurrent neural network
ROB	Remaining onboard
SA	Self-attention
SAC	Soft actor critic
SF	Sheer force
SGD	Stochastic gradient descent
SL	Supervised learning
SMIP	Stochastic mixed integer programming
SPP	Slot planning problem
SSL	Semi-supervised learning
TCG	Transversal center of gravity
TD	Temporal difference
TEU	Twenty-foot equivalent units
TSP	Travelling salesperson problem
UL	Unsupervised learning
VCG	Vertical center of gravity
VP	Violation projection
VRP	Vehicle routing problem
WB	Ballast water tanks
WS	Weighted scaling

Chapter 1

Introduction

This chapter sets the stage for the thesis by introducing the motivation of the research. Subsequently, it provides a thesis statement with the research objectives, summarizes the contributions of this dissertation, and lists relevant publications and research dissemination. Finally, it outlines the thesis structure to guide the reader through the upcoming chapters.

1.1 Research Motivation

Over the past 70 years, container shipping has transformed from a small-scale, labor-intensive operation into a highly efficient system that underpins global trade and modern consumerism [104]. The breakthrough came in 1956 when Malcolm McLean refitted a tanker vessel to carry 58 containers on a voyage from Newark, New Jersey, to Houston, Texas. Beyond refitting the vessel, McLean invented the modern standardized container and developed the automation processes for handling cargo in port operations. Both innovations drastically reduced cargo handling costs, sparking the *container revolution* and reshaping the global logistics landscape [133].

Ever since, global container volumes have continuously increased, reflecting the growing reliance on this mode of transportation for international trade [209], as shown in Figure 1.1. In recent years, container shipping was responsible for transporting approximately 45% of annual goods, valued at $8.1 trillion [215], cementing its status as a cornerstone of the global supply chain. Despite its critical role, many individuals are unaware of the range of essential goods carried by container vessels, including but not limited to medicine, medical equipment, food supplies, electronics, clothing, industrial machinery, automotive parts, and key raw materials [104]. Without the support of container shipping, our world would be unrecognizable, as it serves as the backbone of modern commerce and daily life.

Despite being a critical service in our modern world, container shipping has a significant environmental impact, with annual CO2 emissions reaching 204 million tonnes [138, 201]. To put this into perspective, the average person in Denmark emits 6.9 tonnes of CO2 annually, making the total emissions from container shipping equivalent to those of 29.5 million people, exceeding the combined population of Scandinavia estimated at 28 million. However, container vessels are considered an environmentally friendly mode of transport due to their relatively low emissions per tonne of cargo transported. This efficiency is largely attributed to the scale of the vessels, which allows for significant emission savings. Consequently, improved utilization of vessel capacity enables the transportation of greater cargo volumes on

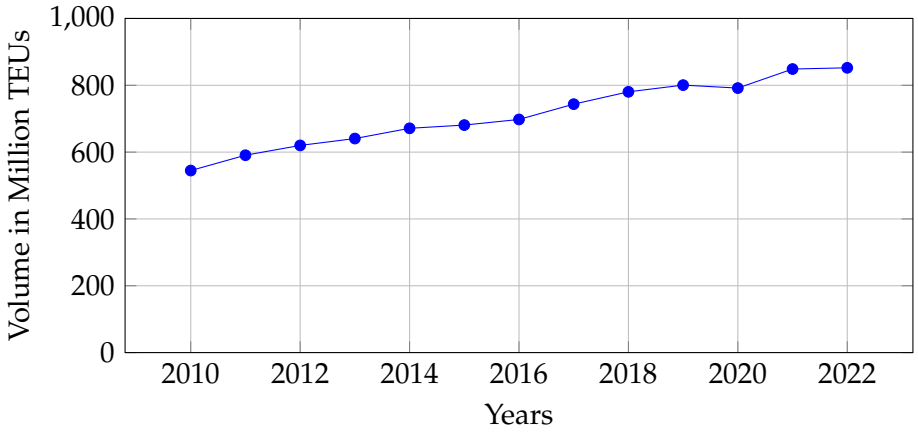


FIGURE 1.1: Trend of global container volumes [216]

existing voyages, further reducing environmental impact. Recognizing this potential, the European Commission has identified container shipping as a key player in the global green transition [67].

Container shipping involves multiple interconnected planning problems, such as pre-marshalling [87], berth allocation [146], quay crane scheduling [83], and container vessel stowage planning [160]. Each planning problem addresses a specific aspect of the container shipping supply chain, sharing similar yet distinct characteristics and complexities. Moreover, the solutions to these problems are interconnected, with the output of one often serving as an input or influence for another. The details of this supply chain will be discussed in Chapter 2. Furthermore, these planning problems are traditionally modeled as combinatorial optimization (CO) problems, characterized by an objective function and a set of hard constraints. While CO methods are well-suited for handling constraints, they often face challenges in scaling efficiently when dealing with large problem sizes or highly complex constraint structures.

Due to its myriad combinatorial aspects and large scale, the container stowage planning problem (CSPP) is arguably one of the most complex interconnected planning problems. The main objective is to assign container cargo to available vessel capacity while maximizing cargo revenue and minimizing operational costs, subject to many combinatorial aspects such as capacity limits, seaworthiness rules, port terminal availability, stowage rules, planning strategies and demand uncertainty [104]. In this mix of combinatorial aspects, we have at least one NP-hard aspect [18, 211], while other aspects are yet to be analyzed. The existing literature exhibits significant variability in problem formulations, assumptions, and solution methods, compounded by the scarcity of publicly available data [7, 160, 130]. This suggests additional work is needed to advance the field of stowage optimization.

The CSPP solution provides decision support to human planners overseeing the planning of container vessel voyages. This planning process can extend over several weeks, particularly for long-haul routes such as the 30- to 45-day journey from Asia to Europe. As vessels approach ports, planners receive additional container bookings. However, the absence of a no-show penalty for container bookings introduces uncertainty regarding the arrival of booked containers. This uncertainty gradually diminishes and is fully resolved by a predefined cut-off time at the port,

beyond which late cargo is rejected. Consequently, the human planner must dynamically update the stowage plan as new information becomes available. To support this process, decision-support systems must operate with relatively short runtimes to accommodate frequent updates and ensure efficient planning.

Current decision-support systems are limited by their inability to generate realistic solutions to the CSPP. Consequently, stowage planning heavily depends on human expertise to produce adequate plans, often leading to suboptimal outcomes in terms of profitability and sustainability. This reliance highlights a significant research gap as human-led planning struggles to account for the intricate complexities of the CSPP. Given the critical role of efficient stowage planning in global supply chains, there is an urgent need for advanced decision-support systems capable of addressing these challenges and enhancing efficient decision-making.

In recent years, machine learning (ML) has been increasingly applied to CO problems, giving rise to the subfield of ML4CO [27]. ML4CO enhances traditional CO methods by either integrating with them to improve efficiency or replacing them entirely in certain applications. Key strategies include learning-based heuristics for faster approximations, guided search to prioritize promising solutions, and end-to-end learning for direct problem-solving. While ML4CO has demonstrated promising results in improving decision-making for CO problems, it still faces challenges such as generalization across problem instances, data requirements, and computational efficiency. Despite these hurdles, ML4CO holds significant potential for advancing decision-support systems in stowage planning, offering more adaptive and scalable solutions for this complex, large-scale optimization problem.

1.2 Thesis Statement

The primary objective of this PhD thesis is to support decision-making in stowage planning by exploring scalable solution methods in ML and CO in order to generate solutions to (subproblems of) the CSPP efficiently.

To achieve the primary objective, the thesis has the following sub-objectives:

1. Investigate the existing literature on stowage planning to explore current solution approaches and identify research gaps.

2. Develop novel problem formulations for (sub)problems of the CSPP to leverage mathematical modeling techniques from ML and CO to enhance efficiency.

3. Compare ML- and CO-based solution methods on synthetic and real-world data to evaluate optimality, efficiency, and feasibility.

4. Examine the theoretical properties of the proposed formulations and solution methods to understand computational complexity and guarantee mathematical soundness.

1.3 Thesis Contributions

This thesis presents several contributions to stowage optimization and ML4CO. The key contributions are summarized below.

- In the article [221], we conducted a comprehensive literature review on container vessel stowage planning. In particular:
  - We introduce a classification scheme to analyze single-port and multi-port CSPPs. We also examine the hierarchical decomposition of CSPPs into master and slot planning problems.
  - We highlight the limited number of publications in this field and identify key challenges in assessing the industrial applicability of existing solution methods due to oversimplified problem formulations.
  - We propose a research agenda that advocates for more representative problem definitions and the development of new benchmark instances where necessary.
- In the article [220], we propose a novel 0-1 integer program (IP) for the master planning problem (MPP) that searches within the space of valid paired block stowage patterns, named template planning. Specifically:
  - The model introduces a novel integration of paired block stowage while ensuring sufficient vessel capacity and incorporating crane makespan, trim, and bending moment constraints.
  - We show that searching for valid paired block stowage patterns is an NP-hard component.
  - Our results demonstrate that template planning outperforms traditional allocation-based approaches in terms of optimality and runtime efficiency while maintaining a sufficiently accurate representation of master planning constraints and objectives.
- In the article [219], we explore the application of deep reinforcement learning (DRL) and present a proof of concept demonstrating its ability to optimize a deterministic, small-scale yet non-trivial MPP. More precisely:
  - This work is the first attempt to formulate the MPP as a Markov Decision Process (MDP), incorporating reward-scaling to enforce constraints.
  - We show that a DRL method can train a model to efficiently find reasonable solutions, serving as preliminary evidence of the potential value of DRL in stowage planning.
- In the article [217], we develop a DRL framework with feasibility projection to solve the MPP under demand uncertainty with dynamic operational constraints, in which:
  - We develop a novel MDP for MPP under demand uncertainty, incorporating problem-specific constraints. To address data scarcity, we release the environment as an open-source implementation[1].

[1] https://github.com/OptimalPursuit/navigating_uncertainty_in_mpp

  - We incorporate differentiable projection layers, including weighted scaling, policy clipping, and violation projection, to enforce inequality constraint satisfaction in DRL frameworks.

  - Our experiments demonstrate that our policy efficiently generates adaptive and feasible solutions under demand uncertainty, significantly outperforming well-known DRL methods and a multi-stage stochastic MIP model.

  - Our decision-support policy transcends deterministic models, enabling dynamic and uncertainty-informed planning in a critical part of the global supply chain.

- In the article [218], we extend the DRL framework proposed in [217] to incorporate paired block stowage patterns and solve problem instances involving realistic vessel sizes and practical planning horizons. Particularly:

  - We extend the original MDP with blocks to include paired block stowage patterns: an industrially relevant planning strategy often overlooked in existing stowage planning research [221]. The extended implementation is released in the existing open-source repository[1].

  - We integrate an action-masking mechanism to enforce non-convex paired block stowage constraints in combination with projection layers to minimize convex feasibility violations in the DRL framework.

  - Experiments show that AI2STOW learns adaptive and feasible policies, outperforming baselines from stochastic programming and DRL in both objective quality and computational efficiency. The evaluation also includes a comparison of different projection layer configurations.

  - AI2STOW can generalize well to larger problem instances, offering decision support for a realistic-sized vessel and operational planning horizons. These findings underscore the potential of DRL-based approaches in developing scalable algorithms for stowage planning.

1.4 List of Publications and Dissemination

The contributions of this thesis have resulted in the following publications, each of which is included as a chapter:

- Jaike van Twiller, Agnieszka Sivertsen, Dario Pacino, and Rune Møller Jensen. 2024. *Literature survey on the container stowage planning problem.* European Journal of Operational Research [221].

- Jaike van Twiller, Agnieszka Sivertsen, Rune Møller Jensen, and Kent H Andersen. 2024. *An Efficient Integer Programming Model for Solving the Master Planning Problem of Container Vessel Stowage.* International Conference on Computational Logistics [220].

- Jaike van Twiller, Djordje Grbic, and Rune Møller Jensen. 2023. *Towards a Deep Reinforcement Learning Model of Master Bay Stowage Planning.* International

Conference on Computational Logistics [219].

- Jaike van Twiller, Yossiri Adulyasak, Erick Delage, Djordje Grbic, and Rune Møller Jensen. 2025. *Navigating Demand Uncertainty in Container Shipping: Deep Reinforcement Learning for Enabling Adaptive and Feasible Master Stowage Planning.* Under Review [217].

- Jaike van Twiller, Djordje Grbic, and Rune Møller Jensen. 2025. *AI2STOW: End-to-End Deep Reinforcement Learning to Construct Master Stowage Plans under Demand Uncertainty.* Under Review [218].

Additionally, a contribution is made to the following publication, which is not included in this thesis:

- Mathias Offerlin Herup, Gustav Christian Wichmann Thiesgaard, Jaike van Twiller, and Rune Møller Jensen. 2022. *A Linear Time Algorithm for Optimal Quay Crane Scheduling*. International Conference on Computational Logistics [83].

Furthermore, the research conducted throughout this thesis has been disseminated at various academic conferences and workshops. The research was presented in plenary sessions at:

- 25th DNV Nordic Maritime Universities Workshop. 2025. *Deep Reinforcement Learning for Revenue Management under Uncertainty in Master Stowage Planning on Container Vessels.* Lyngby, Denmark.

- International Conference on Computational Logistics. 2024. *An Efficient Integer Programming Model for Solving the Master Planning Problem of Container Vessel Stowage.* Monterrey, Mexico.

- European Conference on Operational Research. 2024. *An End-to-End Deep Reinforcement Learning Model for Master Stowage Planning on Container Vessels.* Lyngby, Denmark.

- Odysseus Workshop. 2024. *Deep Reinforcement Learning for Master Bay Stowage Planning.* Carmona, Spain.

- Journées d'Optimzation. 2024. *An End-to-End Deep Reinforcement Learning Model for Master Stowage Planning on Container Vessels*. Montréal, Canada.

- International Conference on Computational Logistics. 2023. *Towards a Deep Reinforcement Learning Model of Master Bay Stowage Planning.* Berlin, Germany.

- International Conference on Computational Logistics. 2022. *Deep Reinforcement Learning for Master Bay Planning on Container Vessels.* Barcelona, Spain.

Another part of the research was presented in poster sessions at:

- D3A 2.0 Conference. 2024. *An End-to-End Deep Reinforcement Learning Model for Master Stowage Planning on Container Vessels.* Nyborg, Denmark.

- Nordic AI Meet. 2023. *Towards a Deep Reinforcement Learning Model of Master Bay Stowage Planning.* Copenhagen, Denmark.

1.5 Thesis Outline

Each chapter in this dissertation is self-contained with respect to terminology, definitions, and mathematical notation. Therefore, while overlap between chapters may exist, their specific meanings or usage could differ based on context.

The remainder of the dissertation is organized as follows:

- Chapter 2 provides background on the container shipping industry.
- Chapter 3 introduces preliminary knowledge on CO, ML and RL, essential for understanding the remainder of the thesis.
- Chapter 4 reviews existing research on container vessel stowage planning, identifying research gaps and proposing a unified research agenda.
- Chapter 5 proposes a 0-1 IP model for solving the MPP by searching in the space of valid paired block stowage patterns
- Chapter 6 explores the use of DRL to solve a small-scale MPP based on reward scaling of objectives and constraints.
- Chapter 7 develops an MDP with explicit action constraints for the MPP under demand uncertainty, which is addressed by a DRL framework with projection layers for action feasibility.
- Chapter 8 extends the DRL framework of Chapter 7 to include paired block stowage and solve the MPP under demand uncertainty for a realistic-sized vessel and operational planning horizons.
- Chapter 9 summarizes the key findings, discusses their implications, provides some ethical considerations, and outlines future research directions in stowage optimization.

Chapter 2

Background

This chapter provides essential background information on container shipping and container vessel stowage planning, including:

- Container dimensions and types (Section 2.1).
- The structure and layout of container vessels (Section 2.2).
- The business of liner shipping (Section 2.3).
- The supply chain of a typical container shipment (Section 2.4).

To avoid redundancy, this will not cover the CSPP and its subproblems in detail, which will be discussed in Chapter 4. However, some overlap with the background sections of the articles in Chapters 4 through 7 is inevitable. For a comprehensive overview of container vessel stowage planning, the reader is referred to [104].

2.1 Containers

All containers will adhere to *ISO standards*, ensuring compatibility across multiple modes of transport and facilitating efficient (un)loading [97]. Container dimensions are measured in imperial units—feet (ft) and inches (in). The regular lengths are 20 ft, 40 ft, and 45 ft, while typical heights are 8 ft 6 in for *standard containers* (DC), and 9 ft 6 in for *highcube containers* (HC). The width is standardized at 8 ft. Common shorthand notations are 20DC, 20HC[1], 40DC, 40HC, 45DC[1], and 45HC. Figure 2.1 shows the dimensions of a 40DC container. Most containers conform to ISO-standardized dimensions and require no special handling. Their structure consists of *corner castings* connected by vertical corner posts and horizontal beams, forming a skeletal frame. The enclosure is made of steel panels for structural integrity and protection, while the floor is typically marine-grade plywood or steel reinforced by structural beams. Depending on cargo needs, containers may feature fully enclosed panels for protection or partial enclosures (e.g., open-top, ventilated) for specialized transport.

However, several containers require special handling, often referred to as *specials*. One of the most prevalent specials is the *refrigerated container (reefer)*, which must be

[1] These container dimensions are rarely seen in practice.

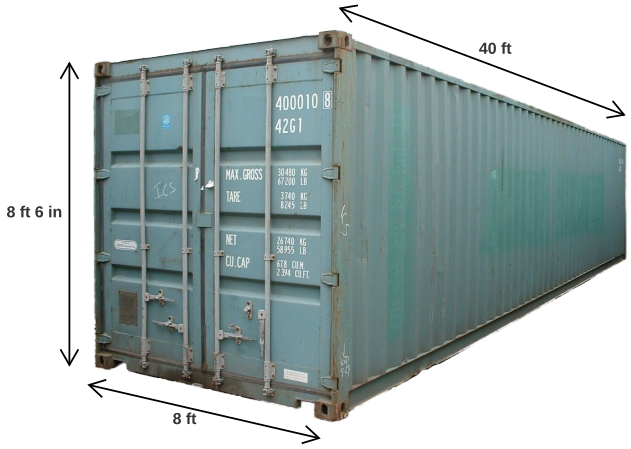


FIGURE 2.1: Dimensions of 40DC container

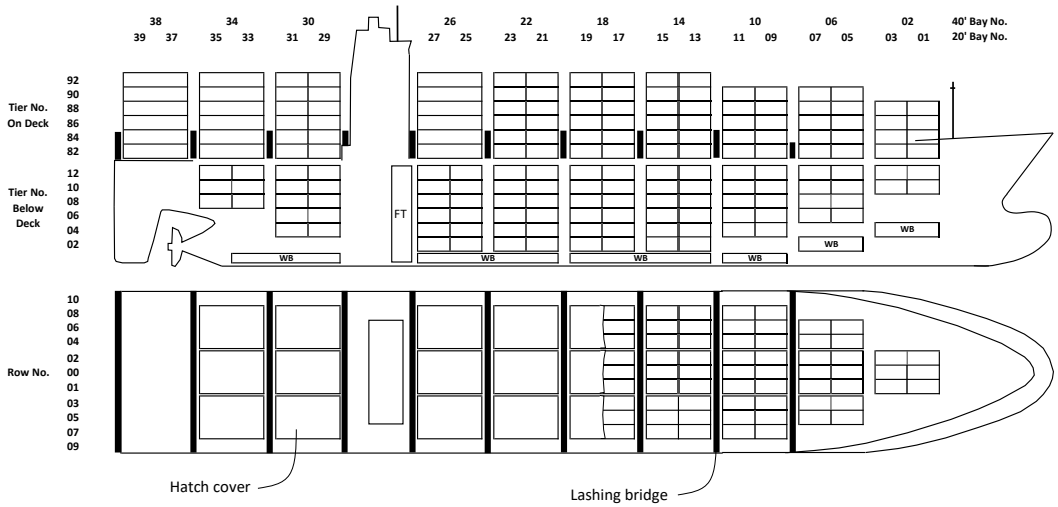
connected to a power outlet to maintain temperature-sensitive cargo. Another special is the *out-of-gauge* (OOG) container, which exceeds standard container dimensions. OOG containers require specific stowage practices to ensure proper placement on the vessel. Furthermore, *dangerous goods* are defined by the *International Maritime Dangerous Goods* (IMDG) code [96]. IMDG containers are classified into specific hazard classes, each with corresponding safety regulations. Operationally, this means that IMDG cargo must adhere to strict segregation rules to ensure safety onboard the vessel. Each container has a *port of load* (POL) and a *port of discharge* (POD), also referred to as an *origin-destination pair* or *transport*. It is important to note that containers are either loaded with cargo or empty, awaiting redistribution across the globe. Containers can be grouped into *weight classes* based on their total weight. When containers share the same dimensions, destination, special type, and weight class, they are generally interchangeable, providing flexibility in the planning process.

2.2 Vessel Characteristics

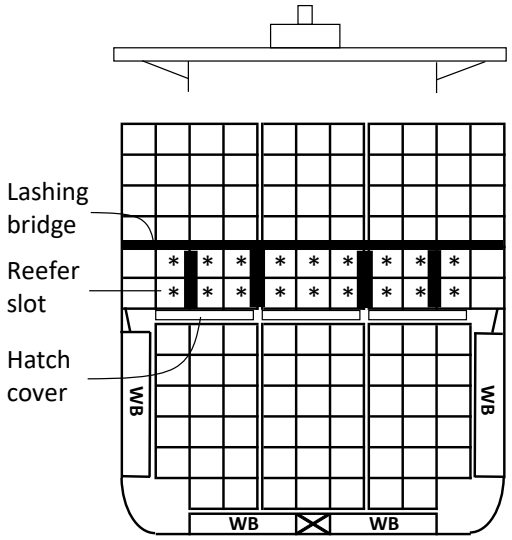
The structure of a container vessel is shown in Figure 2.2. The vessel's *bow*, also called the *fore*, is at the right-hand side of the figure, while the *stern*, or *aft*, is on the left-hand side. The vessel's bottom, known as the *hull*, has a convex shape that enhances hydrodynamic efficiency, buoyancy, and structural integrity by reducing resistance, optimizing load distribution, and ensuring stability in diverse conditions. The bow also features the *bulbous bow*, which reduces wave resistance and improves fuel efficiency by modifying water flow around the hull. To hold cargo, container vessels have a cellular, grid-like structure. Each *cell* can accommodate one 40/45 ft container or two 20 ft containers. The position of one 20 ft container is called a *slot*. A vertical arrangement of cells is called a *stack*. Cells are organized in a three-dimensional *coordinate system (bay, row, tier)*:

- *Bays* divide the vessel longitudinally (fore to aft).
- *Rows* run transversely (side to side).
- *Tiers* represent vertical stacking levels.

The indexing system is standardized [104]. Each bay has three possible indices: an even index (e.g., '02') for a 40 ft container or odd indices for 20 ft container slots (e.g., '01' for the *fore slot* and '03' for the *aft slot*). Each tier is assigned an even number, with *below-deck* tiers starting at 02 and increasing upwards, while *on-deck* tiers typically start at 80 or 82 and continue in even increments. Each row is numbered relative to the centerline (00) of the vessel. Even numbers are assigned to the port side (left), while odd numbers are assigned to the starboard side (right), increasing outward.



(A) Side and top view



(B) Front view

FIGURE 2.2: Container vessel layout with coordinate system [221]

Moreover, on-deck and below-deck container cells are separated by *hatch covers*, which provide structural support and protect the cargo hold. Consequently, the hatch covers also limit the height of below-deck stacks. Depending on the vessel's size and design, each bay typically contains one to four hatch covers. When access to below-deck containers is required, the hatch cover and all containers stacked on top must be removed. The arrangement of containers placed either on top or underneath a hatch cover is named a *block*. Figure 2.2 also show the positions of *fuel tanks* (FT) and *ballast water tanks* (WB).

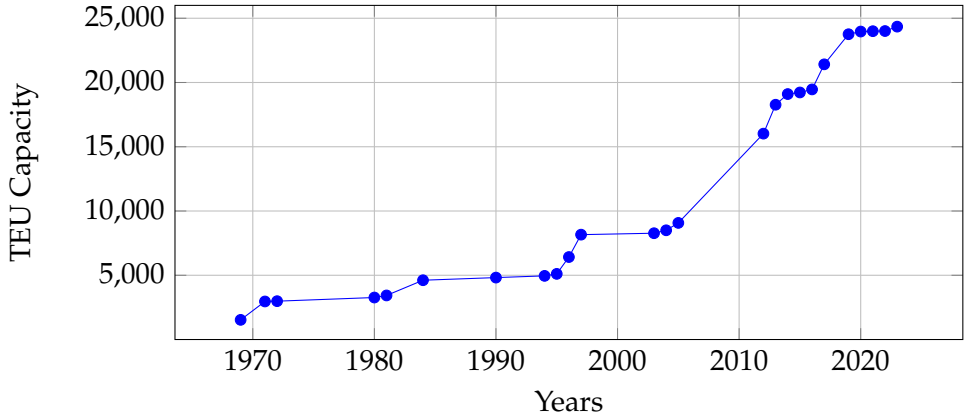


FIGURE 2.3: Trend of largest container ships in TEU capacity [140]

The primary metric for measuring container vessel capacity is the *Twenty-foot Equivalent Unit (TEU)*. Figure 2.3 illustrates the evolution of the largest container ships, with modern vessels reaching capacities of close to 25,000 TEUs. This substantial growth over the past 50 years highlights the increasing reliance on container shipping to support global trade and supply chain demands. Beyond TEU capacity, vessels have a limited number of *reefer slots* with power outlets for reefers, denoted by $*$ in Figure 2.2. Furthermore, a vessel's total load is further constrained by its *deadweight tonnage (DWT)*, the combined weight of cargo, fuel, ballast water, and provisions, alongside the structural mass of the ship named *lightship weight (LWT)*. These capacity metrics impact the overall vessel utilization, stability, and buoyancy.

Containers are secured onboard vessels using multiple methods to ensure safety and stability. First, *twistlocks* are mechanical locking devices inserted into the corner castings of containers, securely locking containers stacked vertically. However, twistlocks alone are insufficient to withstand the significant forces encountered at sea. Second, below-deck containers are stabilized by vertical frames known as *cell guides*, into which standard-size containers slide, providing structural support and preventing lateral movements. Third, containers stacked on deck are secured using *lashing rods*, which connect containers in the lower tiers to fixed structures known as *lashing bridges*. These rods help stabilize stacks by limiting excessive movement caused by the vessel's motions, wind, and rough sea conditions. However, lashing rods have dynamic weight limits; excessive stress due to high stacks or severe weather conditions can compromise the overall safety.

Furthermore, stability is crucial for container vessels to safely withstand dynamic forces experienced at sea. Due to their exceptional size, container vessels are particularly susceptible to stability concerns. Ensuring adequate stability requires careful management of cargo distribution, ballast water, and loading conditions. Several critical parameters influencing vessel stability include the *center of gravity*, which influences *metacentric height (GM)* and *transversal stability*; *vessel trim*, which affects manoeuvrability and fuel efficiency; *listing*, representing lateral inclination due to uneven loading; *draft*, controlling the immersion depth of the vessel; and structural forces such as *shear force, bending moment and torsion moment*, which affect the integrity and structural safety of the hull. Based on the current cargo load, *ballast water* is used to maintain stability and optimize fuel efficiency. When lightly loaded, ballast is added to lower the center of gravity and reduce rolling (tilting rotation

on longitudinal axis); when fully loaded, it is minimized to prevent excessive draft (depth of the vessel below the waterline). It also helps balance uneven cargo distribution to ensure stability. Ballast operations must comply with IMO regulations [93], and modern vessels use ballast water treatment systems to minimize environmental impact. More details on stability and ballast water are provided in Chapter 4.

2.3 Liner Shipping Business

Container carriers are typically liner shipping companies operating container vessels on fixed schedules along established trade routes.

2.3.1 Service Network and Voyage

Liner shipping companies design and optimize their service networks to provide reliable and cost-effective transportation solutions [37]. This involves selecting trade routes, determining port rotations, and deciding on the service frequency. A well-structured network ensures that vessels operate efficiently by balancing transit times with operational costs while meeting market demand. Strategic partnerships, such as vessel-sharing agreements and alliances, play a crucial role in expanding service coverage and improving economies of scale. In some cases, vertical integration of the supply chain, such as investments in inland logistics and terminal operations, can further improve the reliability and efficiency of the service network.

In the liner shipping network, a *voyage* refers to a container vessel following a scheduled route and arriving sequentially at multiple ports, similar to a maritime bus service. Figure 2.4 provides a graphical representation of a voyage, where nodes represent ports and arcs indicate sailing legs. At each port, containers are discharged and (re)loaded, enabling their (trans)shipment to subsequent destinations.

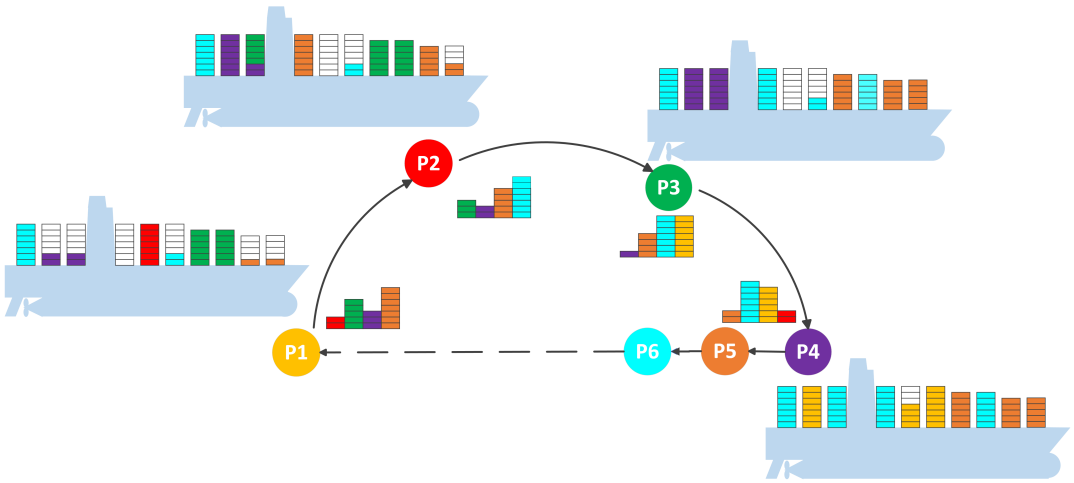


FIGURE 2.4: Container vessel voyage [198]

Voyages typically originate in *supply surplus regions*, such as Asia, and sail toward *high-demand regions*, such as Europe. A common journey from Asia to Europe takes between 30 and 45 days. In industry terms, the *fronthaul* refers to the sailing leg from loading regions to discharge regions, while the *backhaul* is the return leg to the loading regions. Due to trade dynamics, the fronthaul typically carries large cargo

volumes, whereas the backhaul has lower cargo loads and a higher proportion of empty containers. This directional flow of containers creates a global imbalance, requiring the redistribution of empty containers to loading regions.

2.3.2 Fleet Management

Each liner shipping company operates a fleet of vessels to service its network, requiring decisions about *fleet management*. This includes acquiring, chartering, repurposing and retiring vessels based on market conditions. Ultimately, liner shipping must align their fleet capacity with the fluctuating global demand for container shipments to maintain the reliability and efficiency of their services.

2.3.3 Uptake Management

Uptake management involves optimizing cargo bookings to maximize vessel capacity utilization and revenue through dynamic pricing, contract management, and real-time forecasting. Typically, *long-term contracts* ensure higher and less volatile container volumes but generate lower revenue, whereas *spot cargo* offers lower but more volatile volumes with the potential for higher revenue due to market-driven pricing. Due to the lack of no-show fees, both customer types face inherent uncertainty, often leading to double bookings and frequent no-shows. Carriers balance long-term contracts with spot bookings while implementing overbooking strategies to mitigate the effects of no-shows. Forecasting models analyze historical trends to adjust bookings dynamically, while digital platforms provide real-time monitoring, enabling proactive adjustments for disruptions, congestion, and schedule changes.

2.3.4 Cargo Flow Management

Cargo flow management ensures efficient container movement across the network by coordinating vessel schedules, terminal operations, and intermodal transport to minimize delays and bottlenecks. Key tasks include monitoring cargo allocation, managing transshipments, and balancing container repositioning to prevent equipment shortages. A core focus is balancing global container flows to minimize repositioning costs and ensure equipment availability in key regions. Digital tracking and AI-driven analytics provide real-time visibility, enabling proactive responses to disruptions, while collaboration with ports, hinterland logistics, and alliance partners optimizes routing and cost efficiency.

2.3.5 Stowage Planning

Throughout a voyage, *stowage planners* are human experts responsible for allocating containers to available slots on the vessel. Their primary objective is to ensure the efficient and safe transportation of containers while balancing multiple objectives, including maximizing cargo uptake and minimizing operational costs such as terminal and fuel costs. Additionally, stowage plans must account for various constraints, including space and weight limits, hydrostatic stability requirements, lashing force restrictions, the segregation of hazardous materials, and stowage stacking rules. Moreover, factors as vessel size, voyage length with port rotations, uncertain cargo demand, and terminal operation efficiency significantly impact stowage planning. A detailed discussion of objectives and constraints is provided in Chapter 4.

When the vessel is sailing to a port, the planner receives updated information on the container demand. In general, the closer the vessel is to the port, the more accurate the information. The planner must continuously revise the stowage plan in response to these updates. To support this dynamic process, decision-support systems should deliver results within short runtimes, ideally under 10 minutes, to handle frequent changes and maintain efficient planning. Nonetheless, the information remains inherently uncertain, as cargo often does not arrive at the port before the *cargo cut-off time*. This deadline is set some hours before the arrival of the vessel, which ensures there is some time to perform a deterministic plan for the single port. To accommodate future uncertainty, planners utilize planning strategies, such as block stowage patterns and maximizing available space. *Block stowage* involves grouping containers according to their destination ports or similar handling requirements, supporting efficient planning, smooth terminal operations and simplified vessel stability. *Maximizing available space* helps planners retain flexibility, allowing them to accommodate additional cargo or rearrange containers efficiently when unexpected changes occur. These strategies collectively help mitigate the effects of uncertain and fluctuating container demand.

2.4 Supply Chain

To illustrate the supply chain of container shipping, consider the following shipment from a factory in Shenzhen, China, to a retailer in Copenhagen, Denmark.

2.4.1 Hinterland Transport

The journey begins in Shenzhen, China, where a manufacturer loads the goods into a sealed 40 ft container. The cargo is then transported to a nearby port by *hinterland transport* - the inland leg connecting the production sites to the ports. Depending on the available infrastructure and logistics networks, this transport may involve barges, trains, trucks, or a combination of these modes.

2.4.2 Container Terminal and Yard Management

Upon arrival at the Shenzhen port terminal, the container undergoes customs clearance before being assigned a location in the yard. Figure 2.5 illustrates the layout of the yard and quay in a typical container terminal. The *yard* serves as the storage area where containers are organized based on shared characteristics, such as destination, container type, or special cargo requirements, as discussed in Section 2.1. To optimize yard utilization and efficiency, containers are often grouped and periodically rearranged through a process called *pre-marshalling*. This helps minimize retrieval time, reduce terminal congestion, and streamline transport to the quay cranes. Within the yard, cranes and forklifts are responsible for moving containers, and placing them onto trucks or automated guided vehicles (AGVs) for transport to the quay. The *quay* acts as the interface between the terminal and the vessels. Often, it is equipped with multiple *quay cranes* that can operate container vessels simultaneously. The smooth operation of quay cranes is essential for minimizing vessel turnaround times and ensuring timely container transfer. Additional terminal components, such as gates and rail facilities, are crucial for the efficient movement of cargo in and out of the port. These facilities ensure that containers can be processed quickly, maintaining a steady flow of inbound and outbound cargo. To maintain operational efficiency, the terminal defines several key parameters, including berthing

time windows, a minimum number of quay crane moves, a maximum number of free *restows*, i.e., containers discharged and reloaded onto the vessel, and various handling fees.

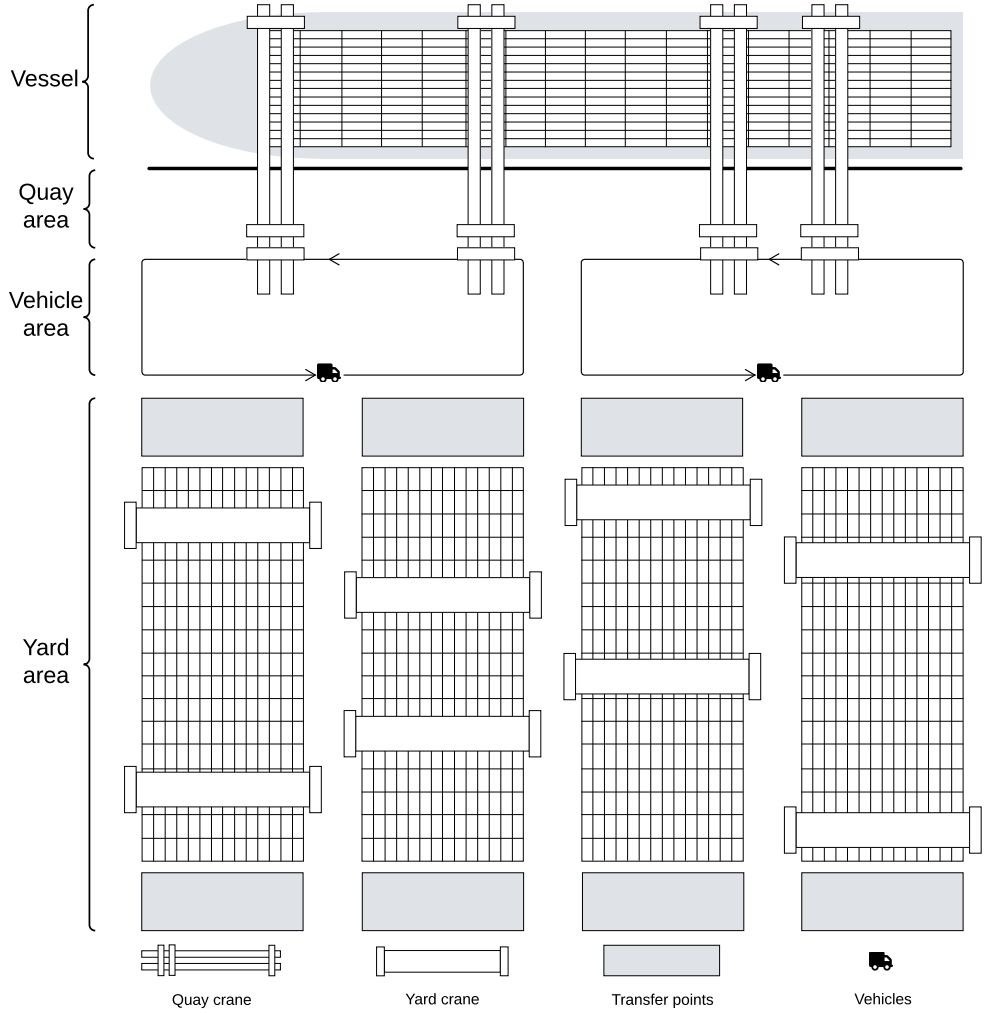


FIGURE 2.5: Layout of yard and quay areas in a container terminal

2.4.3 Berth Allocation

As our container awaits its departure, *berth allocation* assigns its vessel a berth slot at the terminal. At any time, multiple vessels can be berthed, meaning that this planning process is crucial for reducing transport time between yard storage and the quay cranes' operating vessels. Arrival delays or processing time of ships might hinder a vessel from berthing at the originally planned position.

2.4.4 Quay Crane Scheduling

Provided yard arrangements and berth allocation, *quay crane scheduling* determines the number of cranes assigned to each vessel and the order of discharge and loading

operations. The schedule considers contractual agreements between the vessel operator and the terminal, ensuring a minimum number of crane moves is met. Consequently, the terminal plans quay cranes to operate on various berthed vessels during the day, optimizing crane utilization and makespan.

2.4.5 Stowage Planning

A stowage plan is based on the availability of quay cranes and the vessel's arrival condition to assign containers to the vessel's available capacity. This plan must comply with seaworthiness and safety regulations, ensuring proper weight distribution and vessel stability, which also affects fuel consumption. At the same time, the plan seeks to maximize the utilization of available vessel capacity throughout the voyage. The stowage plan must remain flexible to accommodate demand uncertainty, ensuring sufficient capacity for future port calls.

From a terminal perspective, stowage plans should minimize restows and the quay crane makespan. Restows occur when containers on top of stacks block access to containers below, necessitating their removal and reloading. This process causes unnecessary movements of the quay crane, potentially leading to delays and additional operational costs. Efficient stowage planning also means minimizing the *quay crane makespan*, i.e., the time required for the quay crane(s) to operate the vessel. Prolonged crane operations risk exceeding the vessel's berthing window, leading to costly delays and terminal fees.

2.4.6 Load Sequencing

Once the vessel has berthed and the stowage plan is ready, the terminal initiates the load sequence, moving our container to the quay crane for placement on board. In practice, the terminal selects containers that align with the stowage plan while considering type and weight constraints. Although minor deviations may occur, the operational load ensures compliance with the stowage plan to prevent major violations.

2.4.7 Vessel Sailing

Departing from Shenzhen, the vessel prioritizes fuel efficiency and navigational safety en route to its next destination. Trim optimization reduces hydrodynamic resistance by sailing slightly bow-down through cargo distribution and ballast adjustments, while slow steaming lowers speed to cut fuel consumption, costs, and emissions. The captain generally follows the shortest route but may deviate due to weather, piracy risks, or regulatory constraints. Weather routing leverages favorable currents and avoids extreme conditions, while compliance with traffic separation schemes [95] and emission control areas further influences navigation [94].

2.4.8 Discharge Sequencing

Similar to load sequencing, the quay crane discharges containers from the vessel once berthed at the port of Hamburg, Germany. The container is placed in the terminal's stacking area, after which it undergoes customs inspections and clearance before being released for inland transport.

2.4.9 Hinterland Transport

The container is transported by, e.g., truck to the retailer's distribution center in Copenhagen, Denmark. Upon arrival, the seal is broken, and the goods are unpacked. The empty container is returned to the port, awaiting its next journey.

Chapter 3

Preliminaries

This chapter provides the reader with preliminary knowledge required for reading this thesis, including:

- The foundations of CO (Section 3.1).
- An introduction to ML (Section 3.2).
- The fundamentals of RL (Section 3.3).

3.1 Foundations of Combinatorial Optimization

This section introduces several fundamental concepts of CO. While it does not provide a comprehensive overview, it offers essential background knowledge. Readers seeking a detailed introduction to CO and heuristics should consult [74, 183, 228].

3.1.1 Definitions

Combinatorial optimization (CO) is a branch of mathematical optimization where the set of feasible solutions is discrete or can be reduced to a discrete structure. Formally, a CO problem with constraints is defined as follows:

$$\min_{x \in X} \mathcal{F}(x) \tag{3.1}$$

$$\text{subject to:} \quad \mathcal{G}_i(x) \leq b_i, \quad \forall i \in I \tag{3.2}$$

$$\mathcal{H}_j(x) = c_j, \quad \forall j \in J\ , \tag{3.3}$$

where:

- X is a finite or countably infinite set of feasible solutions.
- $\mathcal{F} : X \to \mathbb{R}$ is the objective function to be optimized.
- $\mathcal{G}_i : X \to \mathbb{R}$ are inequality constraints with bounds b_i for indices i in the set I.
- $\mathcal{H}_j : X \to \mathbb{R}$ are equality constraints with values c_j for indices j in the set J.

A common class of combinatorial optimization problems is integer programming (IP), which is formulated as:

$$\min_{x \in \mathbb{Z}^n} c^T x \tag{3.4}$$

$$\text{subject to:} \quad Ax \leq b, \tag{3.5}$$

$$A \in \mathbb{R}^{m \times n} \tag{3.6}$$

$$b \in \mathbb{R}^m \tag{3.7}$$

$$x_i \in \mathbb{Z}\ , \tag{3.8}$$

where:

- $x \in \mathbb{Z}^n$ is the integer decision variable vector of size n.
- $c \in \mathbb{R}^n$ is the cost coefficient vector of size n.
- $A \in \mathbb{R}^{m \times n}$ is the constraint coefficient matrix of size $m \times n$.
- $b \in \mathbb{R}^m$ is the constraint bound vector of size m.
- $x_i \in \mathbb{Z}$ for general IPs, or $x_i \in \{0, 1\}$ for binary integer programming (0-1 IP).

Additionally, linear relaxations of IPs are often used in solution methods by allowing $x \in \mathbb{R}^n$. Whenever the decision variables include both integer ($x_i \in \mathbb{Z}$, $x_i \in \{0, 1\}$) and continuous values ($x_i \in \mathbb{R}$), then a mixed integer program (MIP) is obtained.

Furthermore, the computational complexity of problems varies depending on their specific characteristics. Complexity is often expressed in Big-O notation $\mathcal{O}(f(n))$, which describes an algorithm's upper bound growth rate as a function $f(n)$ of input size n. Problems are categorized into the following complexity classes [14]:

- *Polynomial-time (P)*: The class of decision problems that can be solved by a deterministic Turing machine in polynomial time. That is, there exists a constant k such that for every instance $\omega \in I$, the problem can be solved in time $\mathcal{O}(|\omega|^k)$.
- *Nondeterministic polynomial-time (NP)*: The class of decision problems for which a given solution x can be verified in polynomial time by a deterministic Turing machine. Formally, a decision problem I is in NP if there exists a polynomial-time verifier $\mathcal{V}(\omega, x)$ such that:

  $$\omega \in I \iff \exists x \text{ such that } |x| = \mathcal{O}(\text{poly}(|\omega|)) \text{ and } \mathcal{V}(\omega, x) \in \mathcal{O}(\text{poly}(|\omega|)). \tag{3.9}$$

  Here, $\omega \in I$ represents a problem instance, and x is a certificate (proposed solution) whose size is polynomially bounded in the size of the input.
- *NP-hard*: The class of problems that are at least as hard as the hardest problems in NP. A problem I is NP-hard if for every decision problem $I' \in$ NP, we have:

  $$I' \leq_{\text{poly}} I. \tag{3.10}$$

That is, there exists a polynomial-time computable function Ψ such that for every instance ω of I',

$$\omega \in I' \iff \Psi(\omega) \in I. \tag{3.11}$$

In other words, solving I efficiently allows us to solve all problems in NP efficiently. NP-hard problems may neither be decision problems nor belong to NP; they can be optimization problems or even undecidable.

- *NP-complete*: The class of decision problems that are both in NP and NP-hard. A problem I is NP-complete if:

$$I \in \text{NP} \quad \text{and} \quad \forall I' \in \text{NP}, \quad I' \leq_{\text{poly}} I. \tag{3.12}$$

That is, I is a decision problem for which a proposed solution can be verified in polynomial time, and to which every other decision problem I' in NP can be reduced in polynomial time.

3.1.2 Applications

There is a wide range of applications across various scientific disciplines. Several key application areas include but are not limited to:

- *Networks and graphs*: Shortest paths, spanning trees, multi-commodity flow, graph partitioning [3]; network design [37], and facility location [99], and influence maximization [114] in, e.g., telecommunications, transport, and social networks.

- *Scheduling and timetabling*: Flow shop, open shop, job shop scheduling (JSSP), resource-constrained scheduling, crew scheduling (e.g., airlines, hospitals, railways) [171]; university timetabling [38], and cloud task scheduling [173].

- *Logistics and resource allocation*: Traveling salesperson (TSP), vehicle routing problems (VRP) [129]; knapsack variations, and bin packing [144]; applied in supply chains, transport, and e-commerce.

- *Finance and economics*: Portfolio optimization, risk-aware financial optimization [143]; combinatorial auctions [181], and market equilibria [154].

- *Energy and infrastructure*: Unit commitment, economic dispatch, optimal power flow [229]; renewable energy planning, smart grids, and electric vehicle charging [80].

3.1.3 Solution Methods

To solve CO problems, solution methods are categorized into three main groups: exact algorithms, approximation algorithms, and (meta-)heuristics. Although this is not an exhaustive list, an overview of common algorithms in each category is provided to establish a foundational understanding of existing CO methods.

Exact Algorithms

Exact methods find optimal solutions with optimality and feasibility guarantees.

A classical exact algorithm for solving CO problems optimally is *branch-and-bound* (B&B). This method explores the solution space by solving a linear relaxation, recursively partitioning it into smaller subproblems (branching), and eliminating suboptimal solutions using upper and lower bounds (bounding) [128]. This process constructs a search tree with a branching factor B and solution size or depth n. In the worst case, B&B explores the entire search tree, leading to an exponential time complexity of $\mathcal{O}(B^n)$. To improve efficiency, B&B can be extended in different ways. *Branch-and-cut* (B&C) integrates cutting planes in the framework, adding valid inequalities to tighten the relaxation and accelerate pruning [164]. *Branch-and-price* (B&P) embeds column generation in the framework, solving the LP relaxation over a restricted variable set and iteratively introducing new variables via a pricing subproblem until an optimal solution is reached [25]. Note that both methods operate within the B&B search tree, hence, the worst-case time complexity remains exponential.

Dynamic programming (DP) is a technique that breaks problems into sequential subproblems that exhibit optimal substructures and overlapping subproblems [26]. By recursively solving subproblems and storing their solutions in memory, DP avoids redundant computations, often achieving polynomial-time complexity $O(n^k)$. However, DP's main drawback is its high memory consumption. As n grows, memory requirements can quickly exceed modern hardware capacities, making it impractical for large-scale CO problems. Techniques like state space reduction, iterative DP, and bitmasking are often used to mitigate these limitations.

Constraint programming (CP) is an exact approach that models CO problems as a set of variables, domains, and constraints, solving them via systematic search and constraint propagation [180]. Similar to B&B, CP explores the solution space by pruning infeasible regions using domain reduction, backtracking, and constraint inference. CP is particularly suited for highly constrained discrete problems, where constraint propagation significantly reduces search effort. However, its worst-case complexity remains exponential $\mathcal{O}(B^n)$, making scalability a challenge. Techniques like global constraints, hybrid CP-IP models, and heuristic-guided search improve efficiency.

Approximation Algorithms

Using approximation algorithms, bounded suboptimal solutions can be found with feasibility guarantees and reduced time complexity.

Greedy algorithms iteratively construct solutions by making local optimal decisions, such that the complete solution might be near the global optimum [183]. Typically useful if local decisions contribute to the global optimum, otherwise it likely underperforms. Often, greedy algorithms use sorting or priority queues, giving it a worst-case complexity of $\mathcal{O}(n \log n)$. *Relaxation and rounding* performs a linear relaxation of a discrete problem, after which the fractional solution is rounded into a feasible discrete solution [228]. This ensures feasibility while maintaining an approximation guarantee. The relaxation determines complexity, typically $\mathcal{O}(n^k)$ for LP relaxations, where C indicates polynomial complexity. The *primal-dual* method constructs primal and dual solutions simultaneously, ensuring feasibility and maintaining an approximation ratio [223]. Dual variables guide the near-optimal primal solution. The worst-case complexity is typically polynomial, ranging $\mathcal{O}(m \log n)$ to $\mathcal{O}(n^k)$, depending on the problem, with m being the number of constraints. The

dual fitting method scales a dual LP solution to derive an approximation bound, ensuring feasibility by weak duality [100]. It provides logarithmic or constant-factor approximations, with polynomial time complexity, usually $\mathcal{O}(m)$ to $\mathcal{O}(m \log n)$ with m being the number of constraints.

Heuristics and Metaheuristics

The main idea of (meta-)heuristics is efficiently generating solutions without or with weak optimality or feasibility guarantees. For the sake of brevity, common (meta-)heuristics are grouped.

First, *search-based heuristics* iteratively improves candidate solutions by making modifications. The time complexity can be introduced as $\mathcal{O}(n \times E \times (EV + VL))$, with solution size n, number of iterations E, and time to evaluate EV and verify VL solutions. Common examples of these search-based heuristics include:

1. *Hill climbing:* Incremental changes to progressively enhance solutions, but tends to get stuck in local optima [183].

2. *Local search:* A heuristic that explores the neighboring state of a current solution and accepts the neighbor state if it offers an improvement [183].

3. *Simulated annealing:* Inspired by the physical process of heating and slowly cooling materials. In early stages, the heuristic accepts worse solutions to escape local optima (high *temperature*), after which it becomes more selective over time (lower temperature) [121].

4. *Tabu search:* A memory structure, i.e. *tabu list*, is used to prevent the heuristic from revisiting recently explored areas to encourage exploration of new parts of the solution space [75].

5. *Neighborhood search:* Search strategies that dynamically alter the search scope. *Large neighborhood search* (LNS) reconstructs parts of the solution to explore wider areas [172], whereas *variable neighborhood search* (VNS) incrementally perturbs the solution and systematically varies the neighborhood structure to enhance diversity in the search process [149].

Second, *population-based heuristics* operate by evolving a group of candidate solutions simultaneously. The time complexity of these algorithms can be expressed as $\mathcal{O}(PS \times n \times E \times (EV + VL)$, where PS represents the population size, n the solution size, E the number of generations (iterations), EV the time to evaluate the objective function for each solution, and VL the time to verify each solution's feasibility. Well-known examples of population-based heuristics include:

1. *Genetic algorithm* (GA): Inspired by natural selection, where solutions evolve through crossover and mutation. Parents are selected from the population to undergo crossover, creating offspring that inherit traits. Mutations introduce random variations to maintain exploration of the solution space [84].

2. *Evolutionary strategies:* Similar to GA, but focuses on self-adaptation rather than crossovers [30].

3. *Particle swarm optimization:* Inspired by the swarm intelligence of flocks of birds

or schools of fish. Each solution (particle) moves through the solution space by updating its velocity based on both its local best-known position and the best-known global position of the swarm, balancing exploration and exploitation [115].

4. *Ant colony optimization:* Inspired by the behavior of ants. Solutions are built probabilistically based on pheromone trails, where stronger pheromone levels reinforce better solutions over iterations, guiding future searches toward optimal paths [61].

Third, hybrid methods integrate exact, approximation, and (meta-)heuristic techniques to exploit individual strengths while mitigating weaknesses. The time complexity varies based on the integrated algorithms. Common hybrid approaches are:

- *Matheuristics:* Combining exact methods with (meta-)heuristics to efficiently navigate large search spaces while maintaining feasibility [142].
- *Decomposition-based heuristics:* Splitting a large problem into smaller subproblems solved with different techniques, such as exact solvers for smaller subproblems and heuristics for larger, complex ones [226, 85].
- *Hyper-heuristics:* A high-level strategy to select or generate heuristics dynamically to solve a problem, often adapting to different instances [39].
- *Hybrid (meta-)heuristics:* Combining multiple (meta-)heuristics within a unified framework to leverage their complementary strengths [71].

3.1.4 Challenges and Limitations

CO methods face exponential search spaces, making exact approaches computationally infeasible for large instances, even with refinements like B&C and B&P. Memory constraints in DP and CP further limit scalability, as storing large state spaces quickly exceeds hardware capacities. Approximation algorithms offer quality guarantees but can be computationally expensive, while heuristics and metaheuristics improve efficiency at the cost of optimality and feasibility guarantees and require extensive tuning. Bridging theory and practice remains challenging, as CO techniques often assume static and explicit formulations, whereas real-world applications involve uncertainty and dynamic environments that expand these formulations and the search space. Hence, addressing such challenges is inherently difficult.

3.2 Introduction to Machine Learning

This section introduces several fundamental concepts in machine learning (ML). For a comprehensive, formal introduction to pattern recognition and ML, readers are advised to consult [33, 76].

3.2.1 Definitions

ML is a subfield of artificial intelligence (AI) that enables systems to learn and make decisions based on pattern recognition and data analysis. Within ML, there are different types of learning problems.

In supervised learning (SL), some parameterized function $f_\theta : X \to Y$ maps input $x_i \in X$ to prediction $\hat{y}_i \in Y$. For each input x_i, a target value $y_i \in Y$ exists. Hence, the aim is to minimize the difference between the target and the prediction through some loss function $\mathcal{L}$.

$$\min_\theta \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i) \tag{3.13}$$

In unsupervised learning (UL), the objective is to model the underlying structure or distribution of input data X without labeled target values. This is typically formulated as an optimization problem where the model parameters θ are learned by minimizing an unsupervised loss function $\mathcal{L}_u$:

$$\min_\theta \mathcal{L}_u(f_\theta(X)). \tag{3.14}$$

In semi-supervised learning (SSL), a dataset of labeled and unlabeled samples exists. The objective is to leverage the unlabeled data to improve the model's performance while minimizing the loss of labeled data. A common approach is to minimize both a supervised loss $\mathcal{L}$ on labeled data and an unsupervised loss $\mathcal{L}_u$ on the unlabeled data, weighted by a regularization parameter λ_r:

$$\min_\theta \sum_{i=1}^{M} \mathcal{L}(f_\theta(x_i), y_i) + \lambda_r \sum_{j=M+1}^{N} \mathcal{L}_u(f_\theta(x_j)), \qquad M < N. \tag{3.15}$$

In contrast to the tasks discussed above, reinforcement learning (RL) involves learning an optimal policy through interactions with an environment. Section 3.3 will be dedicated to RL. Some dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ is commonly split into three subsets:

- *Training set* ($\mathcal{D}_{train}$): Used to optimize the model parameters θ by minimizing the loss function.
- *Validation set* ($\mathcal{D}_{val}$): Used to tune hyperparameters and assess model performance during training, helping to prevent overfitting.
- *Test set* ($\mathcal{D}_{test}$): Used for inference to measure performance on unseen data.

Given some task (e.g., SL task), a parameterized function f_θ and a dataset $\mathcal{D}_{train} = \{(x_i, y_i)\}_{i=1}^N$, *training* aims to find the parameters θ^* that minimize loss $\mathcal{L}$:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i). \tag{3.16}$$

The parameters θ are iteratively updated during training using an optimization algorithm. Gradient-based optimization techniques, such as stochastic gradient descent (SGD) and its variants (e.g., Adam, AdaGrad, AdaMax), are commonly used due to their efficiency, scalability, and ability to handle high-dimensional spaces [118]. While alternatives to gradient-based methods are used in ML, they are beyond the scope of this thesis. A key component of gradient-based optimization is *backpropagation*, which efficiently computes gradients by propagating errors backward through

the network using the chain rule. These gradients guide parameter updates, facilitating convergence to local or global minima. The *update rule* is:

$$\theta \leftarrow \theta - \eta \nabla_\theta \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i), \tag{3.17}$$

where $\eta \in (0, 1]$ is the *learning rate*, and ∇_θ denotes the *loss gradient*.

During training, the model f_θ is periodically evaluated on the validation set $\mathcal{D}_{val} = \{(x_j, y_j)\}_{j=1}^{M}$ to assess its performance. The evaluation is based on some metric $\mathcal{M}$:

$$\mathcal{M}(f_\theta, \mathcal{D}_{val}) = \frac{1}{M} \sum_{j=1}^{M} \mathcal{M}(f_\theta(x_j), y_j), \tag{3.18}$$

where $\mathcal{M}$ is a task-specific metric, such as the F1-score for classification or R-squared for regression. The validation process aids in hyperparameter tuning and model selection, ensuring optimal performance before the final evaluation on the test set.

Once training is complete, the parameters θ^* are used to make predictions on unseen data $x' \in \mathcal{D}_{test}$. This process, known as *inference*, does not involve updating θ but only evaluating f_{θ^*} on new inputs:

$$\hat{y}' = f_{\theta^*}(x'). \tag{3.19}$$

Generalization refers to the ability of a parameterized function f_θ to perform well on unseen data. This is influenced by the bias-variance trade-off: models with high bias tend to *underfit*, failing to capture underlying patterns, while models with high variance are prone to *overfitting*, struggling to perform reliably on new instances. *Regularization*, often implemented through the addition of penalty terms to the loss, limits the model's complexity and reduces overfitting.

3.2.2 Applications

ML techniques are widely applied across various fields, enabling automated decision-making and pattern recognition from data. Some key application areas include:

- *Computer vision*: Image classification, object detection [116]; generative models for images [119]; facial recognition and manipulation, and deepfake detection [213].
- *Natural language and speech processing*: Chatbots and virtual assistants, language translation [156]; sentiment analysis, text summarization, speech recognition, and document classification [233, 157].
- *Healthcare and bioinformatics*: Medical imaging analysis, disease diagnosis, personalized treatment drug discovery and development [214] and personalized health monitoring [20].
- *Predictive analytics and forecasting*: Demand prediction, stock market prediction, financial modeling [141]; energy consumption forecasting [174], and climate modeling [68].

- *Fraud detection and cybersecurity*: Banking fraud detection and prevention [79]; spam and phishing detection, malware and intrusion detection [62], and biometric authentication [117].
- *Recommendation and Personalization*: Personalized recommendations (e.g., e-commerce, news feeds, media streaming and personal wellness) [122].
- *Ethics, fairness, and explainability*: Bias mitigation and fairness, explainability and transparency of AI, auditing and compliance of AI, ethical and responsible AI, privacy and human-centered design [199].

3.2.3 Solution Methods

Loss functions define the learning objective by quantifying errors, guiding the optimization process in various ML tasks. The appropriate choice of a loss function depends on the nature of the task. Below, we discuss commonly used loss functions for different yet common learning tasks. For regression tasks, a widely used loss function is the *mean squared error* (MSE), which penalizes larger errors more heavily, making it effective for continuous-valued predictions but sensitive to extreme values.

$$\mathcal{L}_{MSE} = \|f_\theta(x) - y\|_2^2 = \frac{1}{N}\sum_{i=1}^{N}(f_\theta(x_i) - y_i)^2. \tag{3.20}$$

Alternatively, the *mean absolute error* (MAE) provides a more balanced penalty across all errors and is more robust to outliers. However, its non-differentiability at zero can pose optimization challenges.

$$\mathcal{L}_{MAE} = \|f_\theta(x) - y\|_1 = \frac{1}{N}\sum_{i=1}^{N}|f_\theta(x_i) - y_i|. \tag{3.21}$$

A compromise between MSE and MAE is the *Huber loss*, which maintains differentiability while being less sensitive to outliers. It introduces a threshold parameter d that determines the transition from a quadratic loss to a linear loss, balancing sensitivity to large errors with robustness.

$$\mathcal{L}_{Huber} = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}\frac{1}{2}(f_\theta(x_i) - y_i)^2, & \text{if } |f_\theta(x_i) - y_i| \le d,\\ d(|f_\theta(x_i) - y_i| - \frac{1}{2}d), & \text{otherwise.}\end{cases} \tag{3.22}$$

For classification tasks, the *cross-entropy loss (CE)* is commonly used, as it measures the difference between predicted and true probability distributions. This makes it particularly effective for categorical classification problems where the model outputs probabilities for different classes.

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log f_\theta(x_i)_c. \tag{3.23}$$

An extension of cross-entropy loss is the *focal loss*, designed to address class imbalance by focusing more on hard-to-classify examples. The parameter ψ controls the down-weighting of well-classified examples, thereby enhancing performance on

imbalanced datasets.

$$\mathcal{L}_{FL} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}(1 - f_\theta(x_i)_c)^\psi y_{i,c} \log f_\theta(x_i)_c. \tag{3.24}$$

For tasks such as compression and dimensionality reduction, the *reconstruction loss* is often used. This ensures that the reconstructed data closely resembles the original input, preserving essential information while reducing dimensionality.

$$\mathcal{L}_{\text{Rec}} = \|x - f_\theta(x)\|_2^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - f_\theta(x_i))^2. \tag{3.25}$$

Depending on the learning task, various learning algorithms can be implemented. We categorize ML methods into two groups. *Classical ML* refers to traditional algorithms that rely on predefined features and structured data for prediction and decision-making. Classical ML models require manual feature engineering and are relatively interpretable compared to methods discussed below [33].

Deep learning refers to a class of *models* based on *neural networks* (NNs) [76]. These are computational models inspired by the human brain and composed of layers of interconnected neurons that process data hierarchically. They are trained using backpropagation, which computes gradients of the loss function to optimize parameters via gradient-based methods. While NNs are one of many tools in the ML toolbox, they warrant special attention due to their versatility and recent breakthroughs that have significantly advanced various scientific domains, e.g., large language models [156], DNA sequencing [45], and protein folding [106]. Common examples of DL models include:

- *Multilayer perceptrons* (MLPs): Basic neural networks composed of multiple fully connected layers, capable of modeling complex nonlinear relationships in data. They serve as foundational structures in deep learning [76].

- *Convolutional neural networks* (CNNs): Networks specifically designed to process grid-like data, such as images, using convolutional layers to capture spatial patterns. CNNs efficiently identify hierarchical features by exploiting spatial locality [131].

- *Encoder-decoder models:* Neural architectures that convert input data into compact latent representations (encoding) and reconstruct or generate outputs from these representations (decoding). These models are useful for tasks with structured outputs [204].

- *Variational autoencoders*: Models that extend the encoder-decoder framework by encoding input data into probabilistic latent representations. Unlike standard encoder-decoder architectures, VAEs specifically learn latent distributions, enabling the generation of diverse and novel outputs through sampling [119].

- *Generative adversarial*: Models consisting of two competing neural networks, a generator creating synthetic data and a discriminator evaluating its authenticity. This adversarial approach drives continuous improvement in generated data [77].

- *Graph neural networks*: NNs tailored for graph-structured data, explicitly designed to capture relationships among nodes and edges. They effectively handle relational information and structural dependencies within interconnected data [120].

- *Recurrent neural networks*: NNs designed to process sequential or time-dependent data by maintaining internal hidden states. They can model temporal dependencies in sequential data well [182].

- *Pointer networks:* NNs that produce variable-length outputs by directly referencing elements from input data. Their architecture allows for the dynamic selection of input elements as outputs [224].

- *Transformers:* Neural architectures that employ attention mechanisms to model relationships in sequential data. They overcome traditional sequential processing limitations by enabling parallel computation, thereby capturing long-range dependencies efficiently [222].

The time complexity of training models can be expressed as $\mathcal{O}(n \times N_{train} \times MP \times E)$, where n is the input dimensionality, N_{train} is the number of training samples, MP is the number of model parameters, and E is the number of *epochs* (iterations). Similarly, the time complexity of inference can be given by $\mathcal{O}(n \times N_{test} \times MP)$, where N_{test} is the number of test samples. While complexity varies across architectures, DL models generally scale with these key factors, with additional variations based on specific operations (e.g., convolutional layers, attention mechanisms).

3.2.4 Challenges and Limitations

Despite its success, ML faces several challenges and limitations that impact its reliability and applicability. One major concern is *generalization*, where models trained on specific datasets may fail to perform well on unseen data. Additionally, *data quality and availability* remain critical, as biased, noisy, or insufficient data can hinder model performance. *Interpretability* is another key issue, particularly in deep learning, where complex models often act as black boxes, making it difficult to understand their decision-making processes. Furthermore, *computational cost and scalability* pose constraints, as training state-of-the-art models requires significant computational resources. Lastly, *ethical and fairness concerns*, such as bias and privacy risks, highlight the need for responsible AI development and deployment.

3.3 Fundamentals of Reinforcement Learning

This section introduces several foundational topics in RL, which might not be comprehensive. Readers seeking an in-depth introduction to RL are referred to [205].

3.3.1 Definitions

Reinforcement learning (RL) is a branch of ML where an agent interacts with an environment to learn an optimal policy through trial and error. Instead of relying on labeled data, RL optimizes decision-making by maximizing cumulative rewards based on environmental feedback. Typically, RL addresses a sequential decision-making problem, which is formulated as a Markov Decision Process (MDP) defined by the tuple (S, A, P, R, γ), where:

- S is the *set of states* representing possible configurations of the environment.
- A is the *set of actions* the agent can take.
- $P(s' \mid s, a)$ is the *stochastic transition function* $P : S \times A \to \Delta(S)$ of moving from state $s \in S$ to next state $s' \in S$ after taking action $a \in A$. The *deterministic transition function* is $T(s, a)$, where $T : S \times A \to S$.
- $R(s, a)$ defines the reward, either a *deterministic reward function* $R : S \times A \to \mathbb{R}$ or a *stochastic probability distribution over possible rewards* $R(s, a) \sim \mathbb{P}_R(s, a)$.
- $\gamma \in (0, 1)$ is the *discount factor*, balancing immediate and future rewards. It ensures convergence in infinite-horizon problems and stabilizes learning in finite-horizon settings.

A policy can either be described as:

- *Deterministic policy* $\pi(s)$: A function $\pi : S \to A$ that directly maps each state to an action.
- *Stochastic policy* $\pi(a \mid s)$: A probability distribution over actions given a state, where $\pi : S \to \Delta(A)$.

The objective J is to find an *optimal action policy* π^* that maximizes the *expected cumulative reward*, also known as the *return*.

$$\max_{\pi} J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{3.26}$$

The expected return over an infinite time horizon at time step t is given by:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{3.27}$$

where G_t represents the *discounted sum of future rewards* starting from time step t, with γ controlling the weight of future rewards and $r_t = R(a_t, s_t)$ *step-wise reward* of state s_t and action a_t at time step t.

To assess the quality and update a policy π, we define value functions that estimate the expected return under that policy:

- *State value function* (or *value function*) $V^{\pi}(s)$: The expected return when starting from state s and following policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G_t \mid s_t = s\right]. \tag{3.28}$$

- *State-action value function* (or *Q function*) $Q^{\pi}(s, a)$: The expected return when starting from state s, taking action a, and subsequently following policy π:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[G_t \mid s_t = s, a_t = a\right]. \tag{3.29}$$

Algorithm 1 illustrates a generic training process for RL algorithms, while Algorithm 2 describes a generic inference process.

Algorithm 1 Training of RL Algorithm

Input: Environment $\mathcal{M}$, dataset $\mathcal{D}$, policy π, value functions V, Q, horizon T, training iterations N_{train}
Output: Optimized policy π^*

Initialize policy π and value function V
for each training iteration $\iota = 1, \ldots, N_{train}$ **do**
  if Online RL **then**
    Collect trajectory $(s_0, a_0, r_0, \ldots, s_T)$ by interacting with $\mathcal{M}$ using π
  else if Offline RL **then**
    Sample trajectory $(s_0, a_0, r_0, \ldots, s_T)$ from dataset $\mathcal{D}$
  end if
  Compute return G_t
  Update value functions to support policy update $V(s_t), Q(s_t, a_t)$
  Update policy $\pi(a_t \mid s_t)$
end for
Return optimized policy π^*

Algorithm 2 Inference of RL Algorithm

Input: Environment $\mathcal{M}$, policy π^*, horizon T, test episodes N_{test}
Output: Performance metrics

for each test episode $\iota = 1, \ldots, N_{\text{test}}$ **do**
  Observe initial state s_0
  for each step $t = 0, \ldots, T$ or until termination **do**
    Select action a_t based on the trained policy:
    if Deterministic inference **then**
      $a_t \leftarrow \arg\max_a \pi^*(a \mid s_t)$
    else if Stochastic inference **then**
      Sample $a_t \sim \pi^*(a \mid s_t)$
    end if
    Execute a_t, observe next state s_{t+1}
  end for
  Compute episodic return $G_\iota = \sum_{t=0}^{T} \gamma^t r_t$
end for
Return performance metrics

Unlike traditional ML, an RL agent must balance *exploration*, taking new actions to gather information, and *exploitation*, using existing knowledge to maximize rewards, throughout training. A deterministic policy inherently prioritizes exploitation, whereas stochastic policies are often used to encourage exploration.

Generalization in RL helps prevent overfitting and ensures agents perform well in unseen states. However, RL often lacks explicit test sets, making agents prone to memorizing trajectories instead of truly generalizing. Several techniques improve robustness and sample efficiency:

- *Pretraining* involves learning simpler tasks before tackling more complex ones. By progressively building knowledge, agents can acquire useful representations, making it easier to solve difficult tasks.

- *Domain randomization* introduces variations in environment parameters during training, exposing agents to diverse conditions. This technique is particularly useful in scenarios where training occurs in simulations, ensuring that agents can generalize effectively to real-world environments.

- *Meta-learning* trains agents to quickly adapt to new tasks by optimizing for fast learning. Instead of learning a single policy, agents learn how to generalize across tasks, improving their ability to handle novel scenarios.

- *Transfer learning* accelerates adaptation by pretrained policies, value functions, or feature representations. This reduces the need for exploration when encountering new but related tasks. The objective minimizes policy divergence:

$$\min_{\pi_{\theta_t}} \mathbb{E}_{\pi_{\theta_s}} \left[D(\pi_{\theta_s} \parallel \pi_{\theta_t}) \right], \tag{3.30}$$

  where $D(\cdot \parallel \cdot)$ quantifies the shift between policies (e.g., KL-divergence).

3.3.2 Applications

Despite these challenges, RL has led to breakthroughs in areas such as robotics, game playing, and autonomous control.

- *Robotics and Automation*: Self-driving cars, adaptive cruise control, collision avoidance systems, robotic arm manipulation, autonomous vehicles, and energy efficiency improvements [206].

- *Gaming and simulations*: Real-time game decision-making [150], boardgame engines [196], and digital twins [186]

- *Data-driven decision-making*: Algorithmic trading, risk management systems, pricing strategies [81]; decision-making under uncertainty [184], combinatorial optimization [147].

- *Healthcare and life sciences*: AI-driven treatment strategies, drug discovery, medical image analysis [234].

- *Smart infrastructure and scientific research*: Smart devices, energy management [197]; spacecraft control [212], sustainable agriculture [207], and scientific discovery [231].

3.3.3 Solution Methods

RL algorithms can be categorized into model-based and model-free methods. *Model-based RL* constructs an explicit model of environment dynamics, estimating or using the transition function $P(s' \mid s, a)$ for planning. *Model-free RL*, in contrast, learns directly from experience. While it is often less *sample-efficient* - requiring more data to learn a task - it is more practical since transition dynamics are usually unknown.

Another key distinction is between online and offline RL. *Online RL* continuously interacts with the environment and is categorized into on-policy and off-policy methods. *On-policy methods* update using only recent data in an *experience buffer*, ensuring learning stability but reducing sample efficiency. O*ff-policy methods* leverage a *replay buffer* for better sample reuse, improving efficiency but increasing instability. In contrast, *offline RL* learns from a fixed dataset without environment interaction, which is useful in real-world applications where data collection is costly or unsafe.

Traditional RL methods perform well in theory but struggle with high-dimensional problems due to poor generalization and sample inefficiency. *Deep RL* (DRL) addresses these limitations by leveraging NNs as function approximators for value functions (V_ϕ, Q_ϕ) or policies (π_θ), where θ, ϕ represent model parameters. This enables agents to learn directly from raw sensory inputs without requiring handcrafted features. DRL allows for generalization in high-dimensional spaces but also introduces new challenges, such as training instability, high variance, and significant computational demands due to sample inefficiency. DRL has also led to several scientific breakthroughs, e.g., mastering the game of Go [196], fast chip design [148] and autonomous drone racing [110].

Value-based methods optimize a value function to derive a policy indirectly and are commonly used for discrete spaces [26, 205]. As an illustration, the following value-based algorithms are presented using state values $V(s)$, though similar approaches apply to state-action values $Q(s, a)$ as well:

- *Dynamic programming* (DP): A class of model-based algorithms (e.g., value iteration, policy iteration) that recursively update value functions using the *Bellman expectation equation*:

$$V^\pi(s) = \sum_{a \in A} \pi(a \mid s) \sum_{s'} P(s' \mid s, a) \left[R(s, a) + \gamma V^\pi(s') \right]. \tag{3.31}$$

  The *Bellman optimality equation* defines the optimal value function:

$$V^*(s) = \max_{a \in A} \sum_{s'} P(s' \mid s, a) \left[R(s, a) + \gamma V^*(s') \right], \tag{3.32}$$

  from which the optimal policy π^* is derived. DP requires a known transition model and updates all states iteratively, making it computationally feasible only for manageable state spaces. Its sample complexity is relatively efficient at $\mathcal{O}(|S|^2|A|/(1-\gamma))$.

- *Monte Carlo methods* (MC): Model-free algorithms (e.g., first-visit MC, off-policy MC with importance sampling) that estimate value functions by averaging returns over complete episodes. MC requires simulating an entire episode before updating the value function:

$$V(s) \leftarrow V(s) + \eta(G_t - V(s)), \tag{3.33}$$

  where G_t is the cumulative reward from time t to the episode's end. While MC provides *unbiased* estimates, it suffers from *high variance*, making it less sample-efficient. On-policy MC typically has a sample complexity of $\mathcal{O}(|S||A|/(1-\gamma)^2)$, while off-policy MC, which introduces additional variance, requires more samples at $\mathcal{O}(|S||A|/(1-\gamma)^3)$.

- *Temporal Difference Learning* (TD): A class of model-free methods (e.g., SARSA, Q-learning) that update value estimates incrementally using the TD error:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{3.34}$$

Values are then updated in a single step (referred to as TD(0)), while TD(t) represents updates over $t+1$ steps:

$$V(s_t) \leftarrow V(s_t) + \eta \delta_t. \tag{3.35}$$

TD incorporates *bootstrapping*, allowing updates before an episode concludes. This makes TD generally more sample-efficient than MC methods in practice, even though their worst-case sample complexities remain similar: $\mathcal{O}(|S||A|/(1-\gamma)^2)$ for on-policy learning and $\mathcal{O}(|S||A|/(1-\gamma)^3)$ for off-policy learning. However, TD introduces *bias* since it relies on estimates of future values rather than complete returns.

Policy-based methods (e.g., REINFORCE, natural policy gradients) directly optimize the policy $\pi(a \mid s)$, making them effective for high-dimensional and continuous action spaces [205]. These model-free methods explicitly learn the optimal policy by interacting in simulation environments through various techniques:

- *Policy gradients* (PG): A parameterized policy π_θ is optimized by maximizing the objective:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[G_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]. \tag{3.36}$$

The policy parameters are updated via:

$$\theta \leftarrow \theta + \eta \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) G_t. \tag{3.37}$$

However, PG methods suffer from high variance, making training unstable. PG methods are also relatively inefficient, with a high sample complexity of $\mathcal{O}(1/(1-\gamma)^4)$.

- *Variance reduction:* To improve stability without impacting sample complexity, a control variate (or baseline) $b(s_t)$ is subtracted from G_t without altering the expected gradient, yielding the modified update:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[(G_t - b(s_t))\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]. \tag{3.38}$$

A common choice for $b(s_t)$ is $V^\pi(s_t)$, leading to the advantage function:

$$A_t = G_t - V^\pi(s_t), \tag{3.39}$$

measuring the relative benefit of an action compared to the expected return.

Actor-critic methods (AC) combine policy-based and value-based approaches, leveraging a *critic* to estimate a value function for variance reduction and an *actor* to optimize the policy based on advantage estimates [188, 78, 205]. Unlike pure policy gradient methods, actor-critic updates are more stable due to the critic's guidance,

improving convergence. The actor's objective is:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[A^\pi(s,a)\log \pi_\theta(a \mid s)\right]. \tag{3.40}$$

Generalized Advantage Estimation (GAE) is common in AC methods to further refine learning by smoothing advantage estimates, balancing bias and variance through [187]:

$$A_t^{\text{GAE}} = \sum_{l=0}^{\infty}(\gamma\lambda)^l \delta_{t+l}, \tag{3.41}$$

where $\lambda \in [0,1]$ controls the tradeoff: $\lambda = 1$ recovers Monte Carlo estimation with high variance, while $\lambda = 0$ reduces to TD(0) with low variance but high bias.

AC methods fall into two main categories:

- *On-policy actor-critic* (e.g., A2C, PPO) updates both actor and critic using data collected under the current policy, ensuring stability but limiting sample reuse, leading to a sample complexity of $\mathcal{O}(1/(1-\gamma)^3)$.
- *Off-policy actor-critic* (e.g., SAC, TD3, DDPG) improves sample efficiency by leveraging a replay buffer, significantly reducing interactions with the environment and achieving a lower sample complexity of $\mathcal{O}(|S||A|/(1-\gamma)^2)$.

In addition to the RL algorithms, *exploration strategie*s play a crucial role in learning optimal policies [205, 183]. Common techniques include:

- *Epsilon-greedy:* With probability ϵ, choose a random action; otherwise, choose the best-known action.

$$\pi(a \mid s) = \begin{cases} \arg\max_a Q(s,a), & \text{with probability } 1-\epsilon, \\ \text{random action}, & \text{with probability } \epsilon. \end{cases} \tag{3.42}$$

  For continuous actions, Gaussian noise $\mathcal{N}(0,\sigma)$ is added to the chosen action:

$$\pi(a \mid s) = \arg\max_a Q(s,a) + \mathcal{N}(0,\sigma). \tag{3.43}$$

- *Boltzmann exploration:* For discrete actions, use softmax to probabilistically select actions based on state-action estimations Q and temperature τ:

$$\pi(a \mid s) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}. \tag{3.44}$$

  In continuous spaces, stochastic policies sample from, e.g., Gaussian distributions parameterized by estimated mean $\mu(s)$ and standard deviation $\sigma(s)$:

$$\pi(a \mid s) = \mathcal{N}(\mu(s), \sigma(s)). \tag{3.45}$$

- *Entropy regularization:* Encourages stochastic policies by maximizing entropy:

$$\pi(a \mid s) = \arg\max_\pi \mathbb{E}_\pi[G_t] + \alpha H(\pi). \tag{3.46}$$

where $\mathbb{E}_\pi[G_t]$ is the expected return under policy π and $H(\pi)$ is the entropy of the policy, given by $H(\pi) = -\sum_a \pi(a \mid s) \log \pi(a \mid s)$, and α is the entropy coefficient, balancing reward maximization and exploration.

- *Upper confidence bound (UCB):* Encourages exploration by selecting discrete actions with high uncertainty:

$$\pi(a \mid s) = \arg\max_a \left[Q(s,a) + c\sqrt{\frac{\ln t}{N(a)}} \right], \tag{3.47}$$

where $c \geq 0$ is the *exploration coefficient* to control the balance between exploration and exploitation, t being the current timestep, $N(a)$ being the *selection frequency of action a*.

For continuous actions, stochastic policies with UCB are defined by:

$$\pi(a \mid s) = \arg\max_a \left[\mu(s,a) + c \cdot \sigma(s,a) \right], \tag{3.48}$$

where estimates of mean and standard deviations are defined by $\mu(s,a), \sigma(s,a)$.

3.3.4 Challenges and Limitations

Despite its advancements, RL faces several challenges and limitations that affect its practical deployment. One major concern is *sample inefficiency*, as RL models often require many interactions with the environment to learn effective policies, making training costly and time-consuming. Additionally, *generalization* remains a challenge, as policies trained in simulated environments may struggle to adapt to real-world scenarios. *Exploration-exploitation trade-offs* further complicate learning, as agents must balance trying new actions versus optimizing known rewards. *Reward design* is another critical issue, as poorly defined reward functions can lead to unintended or suboptimal behaviors. Furthermore, *stability, safety and feasibility* concerns arise in real-world applications, where unpredictable agent actions may lead to failures or hazardous outcomes. Lastly, *ethical and deployment risks*, such as bias in learned policies and lack of transparency in decision-making, necessitate responsible RL practices.

Chapter 4

Literature Review

This chapter presents the article: *"Literature survey on the container stowage planning problem"* published in the European Journal of Operational Research in 2024 [221]. This study addresses the sub-objective 1 of the thesis.

The literature review synthesizes research on CSPP and its related sub-problems, identifies key methodological and theoretical challenges, and proposes future research directions to advance the field. The insights gained in this chapter provide the foundation for the articles discussed in subsequent chapters, ensuring that the articles build on existing research and address relevant scientific challenges in stowage planning research.

This chapter mirrors the content of the article [221], with each section corresponding directly to a section in the original work. The chapter begins in Section 4.1 with an introduction to the article. Section 4.2 provides an overview of the CSPP, highlighting its relevance and combinatorial complexities. Section 4.3 presents a classification scheme, distinguishing between single-port and multi-port CSPPs and exploring their hierarchical decomposition into master planning and slot planning. Section 4.4 introduces mathematical models for these sub-problems and synthesizes prior research by classifying and discussing existing work. Section 4.5 outlines a research agenda, identifying key gaps and future research directions, including refining problem formulations, developing benchmark instances, and enhancing solution scalability. Finally, Section 4.6 presents the conclusion of the article, whereas the appendices are found in Appendix A.

4.1 Introduction

Container shipping is an underappreciated business. Most people know little about container vessels, and the media coverage often focuses on negative aspects. The truth is that container shipping is the most environmentally friendly mode of transportation with the least CO2 emissions per metric ton of goods shipped per kilometer [92]. Economically, it runs the supply chains of the world, and is in fact believed to have been more important for globalization than freer markets [208].

From an operations research (OR) point of view, the overall objective of container shipping is to maximize the utilization of vessels while minimizing operational costs. Container vessels, however, are challenging. To stow containers on them according to this objective is a combinatorial optimization problem with an unusually wide range of complex constraints and objectives including seaworthiness requirements,

stacking rules, crane utilization, and fuel consumption. For instance, minimizing the number of containers that block containers in lower stacks tiers is NP-hard [17].

Research on the container stowage planning problem (CSPP) is, unfortunately, scarce compared to other areas of OR. In this survey article, we were only able to find 54 key contributions in our literature search covering the 67 years that have passed since the first container vessel sailed in 1956. CSPP studies are challenged in several ways. First, container shipping was deregulated about thirty years after the airline industry when the conference system was outlawed in Europe in 2008. For that reason, there has been less focus on advanced capacity management systems. Second, as mentioned above, the domain is unknown to most researchers and it is only recently that a comprehensive description [104] and benchmark suite [130] was published. Finally, the problem is highly complex and important aspects are subtle and hard to model. It is not clear how to study it in a reduced representative form suitable for scientific research.

In this article, we survey the CSPP literature and propose a research agenda with directions for future work. Our main conclusion is that while several algorithmic frameworks and problem decompositions have been investigated, it is still a maturing research area with relatively few publications and a lack of benchmark suites and problem definition consensus. As a large and sustainable mode of transportation, this state of affairs is important to change.

4.2 Container Stowage Planning Problem

The CSPP is extensive, and it is beyond the scope of this article to describe it in detail. For a full introduction, we refer the reader to [104]. This section gives an overview of the problem and identifies combinatorial aspects that are important to include in representative models of it.

Shipping lines are similar to bus lines but on water. Their fleet of vessels is assigned to closed-loop services with fixed schedules. The CSPP is to decide where on the vessel the booked cargo to load is stowed. It is an operational problem that is solved by a stowage team. Even though stowage plans are made one port at a time, the CSPP is multi-port in nature, as the cargo placed in the current port affects the vessel condition and free capacity in future ports. The input to the CSPP in this multi-port form is the arrival condition of the vessel in the first port, the *loadlist* of cargo to load for each port call, and the vessel and terminal data. The result of the CSPP is a stowage plan for the first or all port calls.

The primary objective of the CSPP is to load all booked cargo by maximizing the available capacity of the vessel. The secondary objective is to minimize terminal fees and the operational costs of the vessel. Finally, since cargo bookings are uncertain, stowage plans must be robust and allow many different cargo compositions in future ports.

The complexity of stowage planning is due to the large size of container vessels that today can be more than 24,000 *twenty-foot equivalent units* (TEU) and a myriad of interacting seaworthiness requirements. To understand the essence of these combinatorial aspects, we need some physical insight into the problem.

Figure 4.1 and 4.2 show a typical cellular design of a container vessel. The storage area consists of *bays* with *stacks* or *rows* of *cells* that normally can either hold one 40′ container or two 20′ containers. The securing system *below deck* consists of *cell guides* that hold the stacks in place. Each hold is sealed with *hatch covers*. The stacks *on deck* rest on the hatch covers or the deck of the ship. The stacks are kept in place by *twist locks* that bind containers together and *lashing rods* that tie container corners to the deck or lashing bridges that are raised to increase stability. All stacks have weight limits. Below deck and fore on deck also have height limits. Both are essential to the model to get the volume and weight capacity of the vessel right. Some cells have power plugs for refrigerated containers (*reefers*). They are indicated with stars in Figure 4.2. The bottom of Figure 4.1 shows the stress forces acting on the vessel. These are described further below.

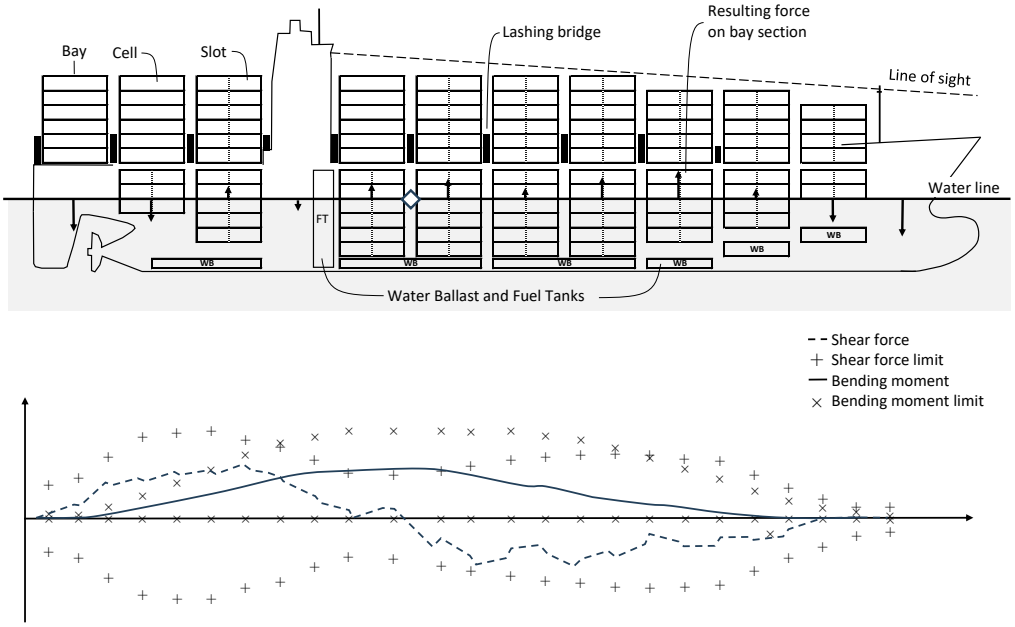


FIGURE 4.1: Vessel side view and stress force graph

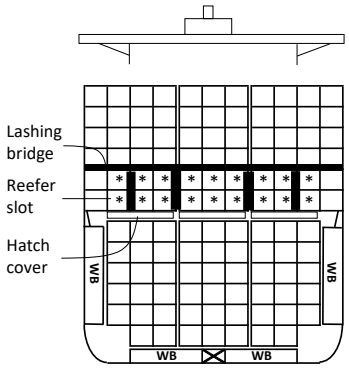


FIGURE 4.2: Vessel bay front view

Most containers are 20′, 40′ and 45′ long, 8′ wide, and 8′6" high. Some of these (mostly 40′ and 45′) are 9′6" high and are called *highcubes*. The weight ranges from about four tons for empty containers to about 30 tons for heavy containers. The port where a container is loaded is referred to as the *port of load* (POL), while the port where it is unloaded is called the *port of discharge* (POD). *Specials* include reefers,

IMDG, and OOG containers. IMDGs carry dangerous cargo and may be required to be segregated. OOGs (Out-of-Gauges) are over-dimensioned, and reefers are refrigerated containers and require electric power.

An important combinatorial aspect of stowage planning is *restow* minimization. A restow happens when a container is unloaded and loaded again before its POD. Restows can be *voluntary* or *mandatory*. Mandatory restows are well-studied and are caused by *overstowage*. *Stack overstowage* happens when a container in a stack is stowed on top of a container at an earlier port. The top container must be restowed in order for the crane to reach the container below it. Restow fees are high relative to the profit margin of each container. For that reason, the number of restows must be minimized. Unfortunately, the minimization of mandatory restows has been shown to be an NP-hard problem [17]. Even when overstowage is simplified to *hatch overstowage*, which is overstowage between containers stowed above and below a hatch cover [211]. Voluntary restows are also subject to restow fees, but these are made on purpose to increase the capacity of the vessel. For instance, a non-reefer container in a reefer slot can be restowed to fit an extra reefer. Since restow minimization is NP-hard and plays a central role in stowage planning, a proper optimization model should include the POL and POD of containers and represent voluntary and mandatory restows.

Another combinatorial element is stowage rules. 20′ containers can not be stowed on top of 40′ containers, and IMDG containers must be segregated depending on their content. Since reefer containers are spark generators, they must be placed away from most IMDGs. Moreover, since reefer plugs typically are at the bottom of stacks, a 40′ reefer kills 20′ capacity. With regards to length, the interactions above can be modeled by just 20′ and 40′ containers. 45′ containers must be placed above lashing bridges or in 45′ bays, but this could be a minor combinatorial issue.

It is an open question whether IMDG segregation is NP-hard. In practice, it is challenging to deal with and should for that reason be modeled on IMDG heavy services. [1] Reefer containers should always be modeled as they have limited positions available and affect other containers. Highcubes are frequent and often take up more than half of the volume. Due to height limitations below deck and in fore bays on deck, it is important to mix highcubes and normal containers right to utilize all the stack volume. It is unclear to us, though, how critical this combinatorial aspect is.

Container weights have a high impact on stowage conditions. They can vary considerably and affect many seaworthiness requirements that are associated with the weight distribution of the vessel. The transversal stability of a vessel is caused by the center of buoyancy (B) moving faster to the side of inclination than the vertical center of gravity (G) as shown in Figure 4.3. For small inclinations, the buoyancy force vector intersects the center line at a fixed point called the metacenter M. The distance between G and M is called the *metacentric height* (GM) and must be positive to avoid the vessel capsizes. A high GM, on the other hand, makes the vessel stiff and may cause the *lashing forces* in lashing rods to be exceeded, particularly if heavy containers are stowed high in stacks on deck. The longitudinal center of gravity (LCG) determines the difference between the fore and aft draft of the vessel called the *trim*. A vessel has a maximum allowed draft (maximum *loadline*), which may be

[1] Carriers also can have their own rules, so-called *handling instructions*. They are easier to deal with than IMDGs and can be ignored.

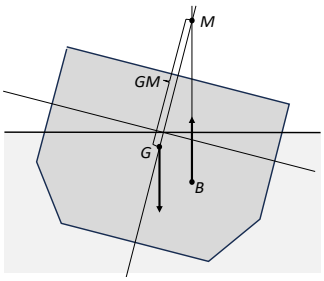


FIGURE 4.3: Metacentric height (*GM*).

further reduced due to port draft restrictions. The trim affects the *line of sight* (LOS) from the bridge, which must be sufficient. The trim also affects the energy efficiency of the engine. The longitudinal weight distribution further causes stress forces on the vessel that all must be within limits.

The question is to what level of granularity we need to model these weight-related constraints of stowage planning. With respect to container weight, homogeneous weight is out of the question as it is unrealistic and compromises a reasonable representation of most constraints above. However, even just three or four weight classes enable us to model the combinatorial interactions of these constraints. With respect to the weight-related constraints, it is key to model GM as it both affects transversal stability and lashing forces. Lashing forces are computed using complex mechanical simulation that makes them hard to embed in optimization models. Nevertheless, they should not be ignored as they easily make one or more top tiers on deck impossible to use (i.e., more than 5% of the total volume capacity). Trim and list requirements can be translated into total weight and box constraints on LCG and TCG. Stress forces also impact capacity. Differences in the gravity and buoyancy forces of each bay section cause resulting forces acting on the vessel, as shown in Figure 4.1. Consider the forces acting on the cross-section between the two first bays in front of the accommodation indicated by a diamond in the figure. The *shear force* (SF) of this longitudinal position is the sum of resulting forces acting fore of the point. The *bending moment* (BM) of the position is the sum of resulting forces fore of this point times the distance to the force. [2]. The bottom graph of Figure 4.1 shows typical levels and limits of SF and BM. In particular, BM can reduce the weight capacity in the fore and aft of the vessel and, for that reason, should not be ignored. SF can be high if mixing full and empty bays, but BM normally has more impact on capacity and, for that reason, is the most important to model. With respect to the vessel structure, the hull can be represented by a sequence of box-shaped sections such that hydrostatic equilibrium can be linearly defined. This representation is fairly accurate for draft, trim, and stress forces down to just six sections [104]. Ballast water is also important to model as it provides a flexible weight buffer that can be used to ease most weight-related constraints.

Another combinatorial aspect of stowage planning is terminal constraints and objectives. Among the constraints, we have draft restrictions and crane work height. Normally, the impact of these restrictions is limited. The hard combinatorial aspects have to do with minimizing the port stay. This is important since a short port stay may allow the vessel to catch up on the schedule or save fuel by reducing the speed

[2]Notice that since the vessel is in hydrostatic equilibrium, we might as well have summed the forces and moments acting aft of the point as they would be equal but with opposite direction.

between ports. There mainly are two ways to minimize the port stay: 1) minimize the total number of quay crane moves, and 2) minimize the makespan of the cranes. With respect to 1), we need to minimize the number of restows, which we already saw is NP-hard. With respect to 2), it is important to understand the relationship between the terminal and the shipping line. Typically, the terminal guarantees a certain number of moves within a time window without specifying the number of cranes working on the vessel during this period. The shipping line can observe the average number of assigned cranes and distribute the moves along the vessel such that these cranes can work in parallel. A complicating matter is that two quay cranes, due to their width, are unable to work simultaneously on two adjacent bays. The total work time is therefore given by the *long crane*, which is the largest number of moves m of any pair of adjacent bays. Let M denote the total number of moves of the vessel. The *crane intensity* (CI) is then defined as M/m. It is an upper bound on the number of cranes that can work in parallel on the vessel without any of them being idle. The stowage plan should have a CI that is larger than the average number of cranes assigned to the vessel by the terminal.

Finally, we turn to a rather hidden combinatorial aspect of stowage planning that has to do with the robustness of plans. To achieve high flexibility for the kind of containers that can be loaded in future ports, stowage planners use certain stowage patterns. To define these patterns, let a *block* denote a storage space either above or below a hatch cover. Hence, if a bay has three hatch covers, it has six blocks in total: one center and two wing blocks on and below deck. A basic stowage pattern is to avoid mixing PODs in a block. This ensures by design that the block has no stack restows and empties the whole block in the port of discharge such that the POD choice of the containers to load back into the block is as free as possible. To avoid hatch restows by design as well, we need to require that the two blocks above and below a hatch cover hold the same POD. We call this pattern *block stowage*. To avoid only changing weight on one side of the vessel and risk excessive torsion moments, the block stowage pattern is often extended to *paired block stowage*, where blocks in the wing hold the same POD, while the center may hold another POD.

Despite the fact that stowage patterns reduce the space of possible plans, they seem to increase the problem complexity substantially [50]. Due to this impact and the fact that the patterns are industry standard, they should be included to some degree in a representative model of the problem.

4.3 Classification Scheme

Before diving into the survey and the proposed classification scheme, let us clarify the search strategy used to collect and select the relevant literature. The articles were found by using Google Scholar using *Stowage Planning* as the keyword. The title of the resulting articles has then been evaluated, and publications that were clearly not relevant have been removed. The remaining publications were further filtered by reading the abstracts. It is here that publications focusing on, e.g., the packing of cargo into a single container were removed. All the references of the remaining publications have been analyzed and missing contributions have been added to the list. A number of publications presenting minor incremental work have been removed from the analysis. It is a key finding that few if any studies fulfill even the minimal representation requirements discussed in Section 4.2. Hence,

TABLE 4.1: Classification scheme

Label	Value	Description
Cargo		Cargo characteristics
	Uni	Uniform weight containers
	Class	Container grouped by weight classes
	Mix	Mixed weight containers
Hydro		Hydrostatics
	Rich	Stability and stress force constraints
	Stab	Stability constraints only
	Equi	Longitudinal, vertical and/or transversal equilibriums
	None	No hydrostatics constraints
CSPP aspects		CSPP aspects present in the problem formulations
	MinRe	Involuntary container restows are allowed as well as minimized
	VolRe	Voluntary and involuntary container restows are allowed and minimized
	NARe	Container restows are not allowed
	HR	Hatch restows created by hatch cover lifts
	RF	Refrigerated containers
	DG	Dangerous cargo
	BW	Ballast water
	La	Lashing forces
	CO	Crane operations
	BS	Block stowage
Obj		Elements of the objective function
	PS	Minimize port stay by minimizing overstowage and optimizing cranes work
	VU	Maximize vessel utilization, consolidation
	H	Minimize fuel consulption, improve hydrostatics
HD		Hierarchical decomposition
Sc		Size of computational study
	S	Small. Vessels with a capacity below 2,500 TEU, for slot planner blocks below 75 TEU
	M	Medium. Vessels with a capacity between 2,500 and 15,000 TEU, for slot planner blocks between 75 and 150 TEU
	L	Large. Vessels with a capacity above 15,000 TEU, for slot planner blocks above 150 TEU
Solution methods		Applied optimization techniques
	Greedy	Greedy approach
	Exact	Exact algorithms
	Method 1/Method 2	Hybrid of two methods, e.g. exact and heuristic
	MatHeu	Matheuristic
	TreeB	Tree-based approach
	NeighMeta	Neighborhood metaheuristic
	PopulMeta	Population metaheuristic
	ML	Machine Learning

the 54 publications selected for this survey either treat most of them or present an original solution approach.

Articles that do not focus directly on solving the CSPP (or one of its sub-problems) were also not considered in the survey, e.g., various descriptions of visualization tools [21, 200], loading computers [155, 230], container data sharing systems [52], crane scheduling tools [89] or loading sequence planners [191]. Survey papers [235], decision support tools [145, 185] and instance generators [53] are not discussed in this classification either. We also consider the papers on inland shipping out-of-scope, as a dedicated literature for this research area already exists [70].

To have an overview of the classified publications and to be able to easily compare the problem formulations presented in them, we developed the classification scheme that is shown in Table 4.1. The *Cargo* attribute determines what types of containers the problem formulation includes. They are divided into three categories: there can be models where all containers have the same weight (*Uni*) or varying weights (*Mix*). Weight classes (*Class*) can also be used, where each weight class corresponds

to a weight range.

The attribute *Hydro* specifies the level of hydrostatics constraints included in the problem formulation. The *Rich* level indicates that both stability and stress force constraints are part of the formulation, whereas, *Stab* means that only stability constraints like GM, trim and/or list are included. *Equi* is related to longitudinal, vertical and/or transversal equilibrium. There is also a possibility that the problem formulation doesn't contain any hydrostatic considerations (*None*).

We note that basic capacity limitations of a container vessel, e.g., constrained height and weight of stacks, are part of every problem formulation included in our classification.

The group called *CSPP aspects* specifies which elements of the problem are included in the problem formulation. The first group is restow handling, which can also vary in the problem formulations. A common approach is to minimize the number of restows in the stowage plan (*MinRe*), whereas, in some cases, they are completely forbidden (*NARe*). In some models, an attempt to create voluntary restows to be able to stow more containers is made while minimizing the number of involuntary restows (*VolRe*). A few formulations include hatch restows (*HR*) caused by hatch overstowage. Crane operations (*CO*) indicate that crane work optimization (for more than one crane) is incorporated in the problem formulation. It can be considered either in the objective function or as a constraint. The inclusion of (*BS*) indicates whether block stowage best practices are modeled. *BW* indicates that it is allowed to use ballast water to fix hydrostatics, and *La* means that lashing forces are part of the problem formulation. Special containers might be included in the load list. It is indicated by *RF* (refrigerated containers) and *DG* (dangerous cargo).

The objective function (*Obj*) might include different aspects of the CSPP. The focus might be on the port stay (*PS*), vessel volume utilization (*VU*), and hydrostatics (*H*) to for example minimize fuel consumption or a combination of these.

The problem might be divided into two or more sub-problems and solved using hierarchical decomposition (*HD*). The attribute, *SC*, indicates the scale of the computational study in terms of problem size and is grouped into three categories: small (*S*), medium (*M*) and large (*L*).

Notice that elements of the table might be written in parentheses. This indicates that this aspect is only partially incorporated into the problem formulation.

4.4 Literature Review

The first works on the CSPP were based on industrial collaborations and featured rich vessel details and problem constraints (e.g, [35, 195]). It was, however, evident that the problem was too complex and that more in-depth studies were needed. Looking at the history of publications, it can be seen that the container stowage planning community has been split between those focusing on the combinatorial elements of the problem (e.g., [18]) and those aiming for industrial strength application (e.g., [226, 109]).

At the core of the first group of research studies, we find the k-shift problem, a simplification of the CSPP where vessel stability and container types are disregarded to

accommodate a more in-depth study of the combinatorial complexity of the mandatory and voluntary container re-stows through a sailing route.

The second group aimed to include all the industrial details of the problem by imposing more pragmatic solution approaches. The seminal work of [226] introduced a hierarchical decomposition to obtain a sequence of tractable interdependent sub-problems. It quickly became popular and widely used in several studies (e.g., [107, 160]). For that reason, the authors propose a division of the problem into two sub-problems, as shown in Figure 4.4. The first sub-problem is named the *Master Planning Problem* (MPP), and its solution (a *master plan*) is the assignment of groups of containers to storage areas of the vessel (blocks). The idea is to address high-level constraints and objectives, such as overall weight distribution, crane utilization, hatch cover moves, and cargo consolidation. Containers in the MPP are often grouped by their weights and types, and a master plan is created for all the ports in the voyage. The second sub-problem is called the *Slot Planning Problem* (SPP), which uses the MPP as input. Given the assignment of groups of containers to each block, the SPP assigns individual containers to slots in the block for every port separately. The assignment fulfills only low-level constraints, such as stacking rules, capacity constraints, and overstowage constraints. This results in a complete (multi-port) stowage plan.

FIGURE 4.4: Hierarchical decomposition of stowage planning into master and slot planning from [160]

We have structured our analysis following the major trends described above to ease the literature review process. Starting from the k-shift and related problems, we move on to multi-port stowage planning problems. Here, we will explore more in-depth the state-of-the-art on the specific master and slot planning problems. Finally, we will explore the results of the single-port stowage planning problem (a special case of the multi-port version). For each of those categories, we provide a qualitative (and, when possible, quantitative) comparison of the main and most influential contributions. The last two subsections discuss studies of computational complexity and other relevant contributions.

4.4.1 k-Shift and Related Problems

The k-shift problem aims at assigning homogenous cargo to a box-shaped vessel over a number of ports. A stack number s and tier number t indicate each possible cargo position. An optimal k-shift plan minimizes the number of container moves, and those include mandatory shifts (due to overstowing cargo) and voluntary shifts. The state-of-the-art model uses a pattern-based formulation with an exponential number of variables. The model is solved with a decomposition method that uses column generation. The model is based on two exponential sets of variables: $x_s \in \mathbb{Z}_+$ indicates the number of *stack-plans* $s \in \mathcal{S}$ that is part of the solution, and $y_{il} \in \{0,1\}$ indicates if at port $i \in P$ the solution uses *port-layout* $l \in \mathcal{L}_i$. A stack-plan indicates the position of j-containers (those being discharged at port j) in the stack for all ports, and it is represented by the constant e^s_{ijr}, which is equal to 1

if stack-plan s includes r j-containers when leaving port i, and 0 otherwise. A port-layout is represented by the constant a_{rs}^{il}, which is equal to the number of stacks in port-layout l for port i, that include r j-containers. The model is as follows:

$$\min \sum_{s \in \mathcal{S}} c_s x_s \tag{4.1}$$

$$\text{s.t} \sum_{l \in \mathcal{L}_i} y_{il} \geq 1 \quad \forall i \in P \tag{4.2}$$

$$\sum_{s \in \mathcal{S}} e_{ijr}^{s} x_s \geq \sum_{l \in \mathcal{L}_i} a_{jr}^{il} y_{il} \quad \forall (i,j) \in \mathcal{T}, r \in R \tag{4.3}$$

$$x_s \in \mathbb{Z}_+ \quad \forall s \in \mathcal{S} \tag{4.4}$$

$$y_{il} \quad \forall i \in P, l \in \mathcal{L}_i \tag{4.5}$$

Each stack plan has an associated number of container moves c_s, which are minimized in objective (4.1). At each port of departure, constraint (4.2) ensures that at least one port layout must be selected. The port-layout and the stack-plans variables are linked together with constraint (4.3). Here, it is ensured that the model selects the necessary stack plans to fulfill the port layout (where R is the set of possible quantities of stack plans). At last, we have the domain constraints (4.4)-(4.5). Recently, this problem has been extended to multiple cargo weights and lengths for medium-sized vessels [167, 169].

Regarding solution methods, construction heuristics are used to find upper bounds in [18, 58]. Moreover, [177] suggests a branch-and-price framework to solve the problem for medium-sized vessels, while [168] formulates a 0-1 IP with valid inequalities to improve the linear relaxation. A compact formulation of this 0-1 IP is suggested in [167, 169]. Despite these efforts, exact methods struggle with solving instances of large vessels. This motivates the use of heuristics, for which promising results are found by GRASP and matheuristics [168, 167, 169]. Other heuristics, such as genetic algorithms, beam search, and simulated annealing, can solve small vessel instances [63, 22, 23].

The first benchmarks are proposed by [18], i.e. *Mixed*, *Long*, and *Short* instances that refer to the expected time cargo is on-board. The *Authentic* instances by [58] have ensured fully loaded vessels at each port, while the *Required* instances by [177] have ensured the presence of shifts in optimal solutions.

Table 4.2 summarizes the computational results of the exact methods for the k-shift problem, which contains formulations of [18], [177], [168]and [169]. The number of optimal solutions and upper bounds (feasible solutions) are presented for each method/formulation and instance group. All formulations, except PAP, can find a feasible upper bound for almost every instance. Moreover, the contributions of PCAR and RP find optimal solutions to almost each Long, Mixed, Short and Authentic instance, while APSW and PAP achieve optimal solutions in most of those instances. Nevertheless, only RP can find optimal solutions in 61 of the Required instances.

4.4.2 Multi-Port Container Stowage Planning

In a realistic setting, stowage planners need to consider vessel conditions and cargo forecasts at future ports in multi-port planning [56]. The goal is to find robust plans

TABLE 4.2: Results summary of exact methods for the k-shift problem. Column Inst. group indicates the instance group, while column # shows the number of instances in each group. Two columns are presented for each method: *Opt.* indicating the number of optimal solutions found and *UB* indicating feasible solutions. The four methods are [18] (*APSW*), [168] (*PAP*), [169] (*PCAR*), and [177] (*RP*).

		APSW		PAP		PCAR		RP	
Inst. Group	#	Opt.	UB	Opt.	UB	Opt.	UB	Opt.	UB
Short	81	78	81	79	79	81	81	81	81
Mixed	81	78	81	77	77	81	81	81	81
Authentic	81	77	81	72	81	79	81	81	81
Long	81	76	81	76	81	81	81	81	81
Required	81	1	81	13	60	14	81	61	79

that maximize vessel utilization and minimize operational costs during the voyage while satisfying constraints that ensure, for example, seaworthiness and safety.

Table 4.3 shows the classification of multi-port work (including the k-shift problem), from which it can be derived that little consensus exists on the modeling of cargo weights and stability. Most contributions combine stability with varying cargo types, e.g., varying cargo weights [137, 90, 169], or special cargo as reefers and dangerous goods [82, 137, 42]. Vessel stability constraints are also extended with hydrostatic calculations such as shear forces [195, 160] and bending moments [35]. The contributions by [107, 22, 23, 135] approximate stability as [12] by balancing cargo weight with respect to the vertical, longitudinal and transversal dimensions. There is, however, also work that disregards hydrostatics in order to focus on other combinatorial aspects (e.g., [18, 226, 159]).

Most studies in Table 4.3 aim to minimize load and discharge moves or mandatory restows. There are, however, two exceptions, namely, voluntary restows (e.g., [18, 58, 177]) and not allowing restows [135, 177, 159]. Furthermore, earlier work considers hatch restows (e.g., [195, 109, 160]), while stowage plans with crane operations are more frequently recurring (e.g., [82, 23, 42]). In less frequent cases, block stowage strategies are proposed by [226, 137, 159], and ballast water is modeled in [195, 35]. Lashing forces are only considered in [195].

At the core of multi-port stowage is the planning of loading and discharging containers (i.e., shifts), voluntary restows, and efficient port operations. The minimization of restows (whether voluntary or not) can be traced back to [35], which modeled two decision variables to load or remove container c into/from a slot with bay b, row r, and tier t at any port p between loading port l and discharge port d. However, this formulation was not broadly adopted by the community as it was too complex when vessel stability was to be considered. The previously discussed k-shift problem can be seen as the continuation of the work of [35], where vessel stability is disregarded. The work by [109] assigns cargo to blocks and subsequently creates stacks ordered by destination and weight, thereby reducing mandatory restows. This assignment formulation often scales to medium-sized vessels (e.g., [107, 82, 160]). It is extended to a stochastic program with uncertain container weights [135] and a block stowage problem with crane intensity [159]. This assignment problem coincides with the combination of the master and slot planning problems in Subsections 4.4.2 and 4.4.2; we refer to those models for an example of the formulation.

TABLE 4.3: Classification of multi-port container stowage planning problems

Paper	Cargo	Hydro	CSPP aspects	Obj	Sc	HD	Solution methods
[195]	Class	Rich	MinRe, HR, BW, La, RF	PS, VU, H	M		Greedy
[35]	Mix	Rich	VolRe, HR, BW	PS	S	✓	Exact/TreeB
[19]	Uni	None	VolRe	PS	S		Greedy
[18]	Uni	None	VolRe	PS	S		Greedy
[226]	Class	None	MinRe, HR, CO, BS, DG, RF	PS, VU	S	✓	Exact/NeighMeta
[63]	Uni	Stab	VolRe	PS	S		PopulMeta
[109]	Class	Stab	MinRe, HR	PS	M	✓	Greedy/TreeB
[107]	Class	Equi	MinRe, DG, RF	PS	M	✓	Exact/NeighMeta
[82]	Mix	Stab	MinRe, CO, DG, RF	PS	M		PopulMeta
[137]	Class	Stab	MinRe, HR, CO, BS, DG, RF	PS, VU, H	M		Greedy/NeighMeta
[160]	Mix	Rich	MinRe, HR, CO, RF	PS	M	✓	Exact/NeighMeta
[22]	Uni	Equi	VolRe	PS, H	S		NeighMeta, TreeB, PopulMeta
[58]	Uni	None	VolRe	PS	M		Greedy
[90]	Mix	Stab	MinRe	PS, H	S		PopulMeta
[23]	Uni	Equi	VolRe, CO	PS	S		PopulMeta
[135]	Class	Equi	NARe	VU	S		NeighMeta, Exact
[159]	Class	None	NARe, CO, BS	PS	L		NeighMeta
[177]	Uni	None	VolRe, NARe	VU	M		Exact
[168]	Uni	None	VolRe	PS	L		Exact, NeighMeta
[167]	Class	Stab	VolRe	PS	L		Exact, MatHeu, NeighMeta
[169]	Class	Stab	VolRe	PS	M		Exact, MatHeu
[42]	Mix	Stab	MinRe, CO, (DG), RF	PS	S		PopulMeta

An optimal solution to a representative multi-port problem is yet to be found. Due to the complexity of the multi-port CSPP, many attempted to address it by dividing it into separate master and slot planning subproblems using decomposition methods [35, 226, 109, 107, 160]. Several different combinations of solution methods were presented. The early work of [35] suggests a hierarchical decomposition that combines a 0-1 IP with an enumeration tree. Consequently, hybrids of exact and neighborhood-based methods [226, 107, 160] or greedy and tree-search heuristics [109] are proposed to solve master and slot planning sequentially.

Early computational experiments have been published by [109], who iteratively solves the problem at each port and adds constraints to the MPP to minimize restows. In [160], we instead see a focus on larger problem instances and an increased complexity of SPP constraints. The decomposition is similar to the one proposed in [109] but without iterative interaction between MPP and SPP. Though the experiments are based on very different hardware, it is still possible to analyze the impact of vessel size and the number of planned ports on the computation time. Figure 4.5 shows a plot of the reported execution time from the two papers. An analysis of the results shows a weak negative correlation between the vessel size and the execution time (-0.29). In contrast, a stronger correlation appears between the execution time and the number of ports (0.67). The red line in the figure shows the increasing linear trend of the results[3]. The collected data is shown in Table A.1 of Appendix A.1. As many publications only include computational studies over single instances, we could not include more experiments in this comparison.

Recent contributions have proposed heuristic frameworks to deal with multi-port problems, such as genetic algorithms [82, 90, 42], (large) neighborhood search [159, 135], or a framework of greedy and tabu-search heuristics [137]. Despite their algorithmic efficiency, more computational studies on a general problem definition with benchmark instances are needed to verify the findings.

[3]The analysis was done discarding results where the algorithm timed out. Also, 2 results with extremely long runtimes were considered outliers.

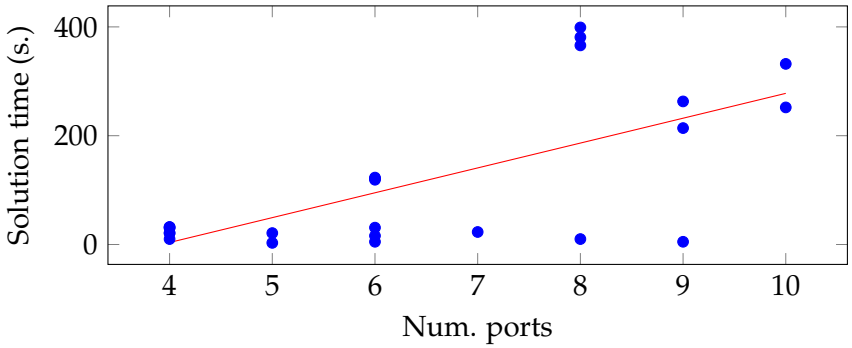


FIGURE 4.5: Execution time over the number of ports in the tested instance [160, 109]. In red is the linear trend line.

TABLE 4.4: Classification of master planning problems

Paper	Cargo	Hydro	CSPP aspects	Obj	Sc	Solution methods
[161]	Class	Rich	BW, RF	H	L	Exact
[158]	Class	Rich	(MinRe), HR, CO, RF	PS	M	NeighMeta
[8]	Class	Equi	HR, CO, RF	PS	L	MatHeu
[9]	Class	Equi	HR, CO, RF	PS	L	Exact
[10]	Class	Equi	HR, CO	PS, VU	M	Exact, MatHeu
[11]	Class	Rich	HR, CO, RF	PS	L	Exact, MatHeu
[111]	Class	Rich	HR, CO, DG, RF	PS, VU	L	Exact
[31]	Class	Rich	MinRe, HR, BW	PS, H	L	Exact/NeighMeta
[43]	Class	None	HR	PS, VU	M	Exact

Despite the many implementations in Table 4.3, it is challenging to compare their performance adequately. On the one hand, little consensus exists on a general problem definition or benchmark instances. Hence, the problem is attempted to be solved from different angles.

In the following subsections, publications concerning solely MPP and SPP are presented and compared, supplemented by relevant mathematical models. Notice that the two models can trivially be combined into a single multi-port container stowage planning model (though very likely not an efficient model).

Master Planning

Though it is an important part of the hierarchical decomposition that many researchers use, the SPP (when solved heuristically) has a minor impact on the runtime of the solution approach and its general objectives. For that reason, a number of scholars find it legitimate to focus their studies on solving the Master Planning Problem (MPP).

The MPP aims to allocate cargo to subsections of bays. Those are often called locations [160] or blocks [48]. A block can either be a logical grouping of containers or be defined by the position of the hatch covers. The assignment of containers to blocks must ensure that the vessel is seaworthy while minimizing the handling time of the vessel. The state-of-the-art model for this problem is based on the formulation of [160], and is as follows. The notation for the sets and parameters of the model is presented in Tables 4.5 and 4.6.

The formulation is based on a main set of decision variables indicating the number of containers of a specific type to be stowed on a block during a transport leg. Other

TABLE 4.5: Common sets for the MPP

Sets	
B	The set of bays
BL	The set of blocks
BL_b	The set of blocks in bay $b \in B$
BL^O	The set of blocks over deck
BL_l^U	The set of blocks below deck under block $l \in L^O$
Bin	The set of adjacent bays $(b_1, b_2) \in B \times B$
T	The set of container types
$T^{\{20,40,R\}}$	The set of 20-, 40-foot and Reefer container types
P	The set of ports
TR	The set of transports (port pairs $(o, d) \in P \times P, o < p$)
TR_p^{ON}	The set of transports $(o, d) \in TR$ where $o < p$ and $d > p$
TR_p^A	The set of active transports at port $p \in P$ where $(o, d) \in TR$ where $o = p$ or $d = p$
TR_p^{OV}	The set of overstowing transports at port $p \in P$ where $(o, d) \in TR$ where $o < p$ and $d > p$

TABLE 4.6: Common parameters of the MPP

Parameters	
$K_l^{\{20,40,R\}}$	The 20-,40-foot, and Reefer capacity of block $l \in L$
TEU^τ	The Twenty-Foot Equivalent units of container type $\tau \in T$
W_τ	The weight of container type $\tau \in T$
L_τ	The length of container type $\tau \in T$
LD_t^τ	The number of container of type $\tau \in T$ to be loaded for transport $t \in TR$
$R_{p,l}^\tau$	The number of containers of type $\tau \in T$, already on board the vessel with destination $p \in P$.
W_b^K	The lightship weight at bay $b \in B$
$W_l^{\max}$	The maximum weight limit of block $l \in L$
D_p	The total displacement of the vessel leaving port $p \in P$
CG_l^α	The center of gravity components $\alpha \in L = LCG, V = VCG, T = TCG$ of block $l \in BL$
$\overline{CG}_b^\alpha$	The center of gravity components $\alpha \in L = LCG, V = VCG, T = TCG$ of bay $b \in B$
$LCG^{\{Min,Max\}}$	The limits for the vessel's longitudinal center of gravity
$VCG^{\{ax\}}$	The maximum vertical center of gravity of the the vessel
$TCG^{\{Min,Max\}}$	The limits for the vessel's transversal center of gravity
$Shear_b^{\{Min,Max\}}$	The shear limits at bay $b \in B$

indicator variables are used for the calculation of the objective value. A description of the variables follows.

$x_{tl}^{\tau} \in \mathbb{Z}^{+}$	The number of containers of type $\tau \in T$ stowed in block $l \in L$ during transport $t \in TR$.
$\delta_{pl} \in \mathbb{B}$	Stowage indicator equal to 1 if any load or discharge containers are present in block $l \in L$ at port $p \in P$.
$y_{pl}^{O} \in \mathbb{R}^{+}$	The number of hatch-overstow containers in block $l \in L$ at port $p \in P$
$y_{p}^{T} \in \mathbb{R}^{+}$	The long-crane (or makespan lower bound) at port $p \in P$.

Following is the mathematical formulation and its description.

$$\textbf{min} \sum_{p \in P} \left(C^{T} y_{p}^{T} + \sum_{l \in L} C^{O} y_{p}^{O} l \right) \tag{4.6}$$

s.t.

$$\sum_{l \in L} x_{tl}^{\tau} = LD_{t}^{\tau} \quad \forall \tau \in T, t \in TR \tag{4.7}$$

$$x_{(1,p)l}^{\tau} = R_{p}^{\tau} \quad \forall \tau \in T, p \in P \tag{4.8}$$

$$\sum_{t \in TR_{p}^{ON}} \sum_{\tau \in T} TEU^{\tau} x_{tl}^{\tau} \leq K_{l}^{20} \quad \forall p \in P, l \in L \tag{4.9}$$

$$\sum_{t \in TR_{p}^{ON}} \sum_{\tau \in T^{\alpha}} x_{tl}^{\tau} \leq K_{l}^{\alpha} \quad \forall p \in P, l \in L, \alpha \in \{20, 40, R\} \tag{4.10}$$

$$\sum_{t \in TR_{p}^{ON}} W_{\tau} x_{tl}^{\tau} \leq W_{l}^{\max} \quad \forall p \in P, l \in L \tag{4.11}$$

$$\sum_{b \in B} W_{b}^{K} \overline{CG}_{b}^{L} + \sum_{t \in TR_{p}^{ON}} \sum_{l \in L} CG_{l}^{L} W_{\tau} x_{tl}^{\tau} \geq LCG^{Min} D_{p} \quad \forall p \in P \tag{4.12}$$

$$\sum_{b \in B} W_{b}^{K} \overline{CG}_{b}^{L} + \sum_{t \in TR_{p}^{ON}} \sum_{l \in L} CG_{l}^{L} W_{\tau} x_{tl}^{\tau} \leq LCG^{Max} D_{p} \quad \forall p \in P \tag{4.13}$$

$$\sum_{b \in B} W_{b}^{K} \overline{CG}_{b}^{T} + \sum_{t \in TR_{p}^{ON}} \sum_{l \in L} CG_{l}^{T} W_{\tau} x_{tl}^{\tau} \geq TCG^{Min} D_{p} \quad \forall p \in P \tag{4.14}$$

$$\sum_{b \in B} W_{b}^{K} \overline{CG}_{b}^{T} + \sum_{t \in TR_{p}^{ON}} \sum_{l \in L} CG_{l}^{T} W_{\tau} x_{tl}^{\tau} \leq TCG^{Max} D_{p} \quad \forall p \in P \tag{4.15}$$

$$\sum_{b \in B} W_{b}^{K} \overline{CG}_{b}^{V} + \sum_{t \in TR_{p}^{ON}} \sum_{l \in L} CG_{l}^{V} W_{\tau} x_{tl}^{\tau} \leq VCG^{Max} D_{p} \quad \forall p \in P \tag{4.16}$$

$$\sum_{b'=1}^{b} \left(W_{b'}^{K} + \sum_{l \in BL_{b'}} \sum_{\tau \in T} \sum_{t \in TR_{p}^{ON}} W_{\tau} x_{tl}^{\tau} \right) \geq Shear_{b}^{Min} \quad \forall p \in P, b \in B \tag{4.17}$$

$$\sum_{b'=1}^{b} \left(W_{b'}^{K} + \sum_{l \in BL_{b'}} \sum_{\tau \in T} \sum_{t \in TR_{p}^{ON}} W_{\tau} x_{tl}^{\tau} \right) \leq Shear_{b}^{Max} \quad \forall p \in P, b \in B \tag{4.18}$$

$$\sum_{\tau \in T} \sum_{t \in TR_{p}^{A}} \sum_{l' \in BL_{l}^{U}} x_{tl'}^{\tau} \leq M \delta_{pl} \quad \forall p \in P, l \in BL^{O} \tag{4.19}$$

$$\sum_{\tau \in T} \sum_{t \in TR_{p}^{OV}} x_{tl}^{\tau} - M(1 - \delta_{pl}) \leq y_{p}^{O} \quad \forall p \in P, l \in BL^{O} \tag{4.20}$$

$$\sum_{b \in N} \sum_{\tau \in T} \sum_{t \in TR_{p}^{ON}} \sum_{l \in B} x_{tl}^{\tau} \leq y_{p}^{T} \quad \forall p \in P, N \in Bin \tag{4.21}$$

The formulation minimizes the makespan and the hatch overstowage at each port (4.6). Constraint (4.7) ensures that all cargo must be stowed on the vessel, while constraint (4.8) enforces that cargo already on board does not change position. The total block capacity, the type-specific block capacity, and the block weight capacity are constrained by (4.9), (4.10), (4.11), respectively. To ensure vessel stability, the longitudinal, transversal, and vertical centers of gravity are constrained within the given limits by constraints (4.12) - (4.16).Constraints (4.17) and (4.18) impose minimum and maximum levels for the shear forces that act on the vessel.

Using an indicator variable, constraint (4.19) identifies, for each port, blocks below deck that require container moves. Should those container moves be blocked by containers on-deck, they will be captured in constraint (4.20) as overstowing. Finally, the bay pair with the maximum number of movements at each port is identified with constraint (4.21), representing the makespan.

Table 4.4 compares publications and shows trends in master planning research. In the MPP, cargo is grouped by weight and sorted into corresponding weight classes (the knowledge of the true container weights is less important when building a master plan). While reefer containers are included in most test instances, only one study handles dangerous cargo [111].

The problem formulations tend to include hydrostatic calculations of varying degrees, often more than trim and GM estimations. [10, 8, 9] based hydrostatics calculations on a bit older and less accurate model including equilibrium consideration that entails balancing weights on the vessel [12]. [11] uses a richer stability model that includes trim, GM, and shear forces. [43] did not consider stability constraints in their problem definition at all but focused instead on a new IP formulation based on a minimum-cost flow problem with a multi-commodity network structure. The objective function contains the cost of the assignment of a container and the cost of using extra bays.

There are only a few studies that allow using ballast water to fix possible instabilities [161, 31]. [161] proposed an IP with an approximation of the displacement and a linearization of the center of gravity calculations to include the effect of the ballast water on the hydrostatic values.

The objective function mostly included balancing crane work and minimizing hatch restows, but some of the formulations also focused on maximizing vessel utilization in addition to port stay optimization [10, 111, 43]. While accounting for unnecessary hatch cover moves is considered in the MPP publications, there is rarely a focus on minimizing restows within blocks. Only [31, 158] presented related constraints.

From a mathematical modeling point of view, most models ([158, 11, 111, 31]) are inspired by or extensions of the formulation proposed by [12] and [160]. The only scientific work that has proposed a different formulation is that of [43], where the problem is modeled using a network-flow representation.

A summary of the solution methods is shown in Table 4.4. [8, 11] introduced the use of matheuristics that decompose the MIP. Firstly, an assignment of container destinations to blocks was found by solving the relaxed MIP with linearly relaxed variables indicating the assignment of containers to blocks. Secondly, a heuristic

called the progressive random fixing procedure was used to obtain the feasible solution, where the assignment of container destinations to blocks is given from the previous phase, and there is no relaxation of the MIP. [9, 10] introduced two MIP formulations: the first one was a binary representation of the problem, and the second one expressed the number of containers in TEU and resulted in a more compact formulation. For the solution approach, the authors proposed to primarily solve a relaxation of the second model and then use its solution as input for solving the first model. [158] explored the use of large neighborhood search with the results from a relaxed IP as a warm start. [31] proposed a heuristic, where, firstly, a problem was solved with relaxed trim constraints, and, secondly, a local search was performed to fix eventual instabilities.

By looking at Table 4.4, we can observe that most of the proposed approaches were tested against large instances (above 15,000 containers).

Though the problems and the formulations are very similar, the lack of a common benchmark has made it impossible to compare results. To remedy this situation, we propose a new set of publicly available benchmark instances based on the vessel data from [130]. For each of the 3 available vessels, 2 random instances are generated for each combination of ports $\{5, 7, 10\}$ percentage of cargo already on board $\{0\%, 15\%, 30\%\}$ and vessel utilization $\{60\%, 70\%, 80\%\}$. The cargo list simulates an ocean-going service where the long-haul leg (sailing from one region to another) guarantees the provided vessel utilization. The benchmark has a total of 162 instances, all of which are available at https://doi.org/10.11583/DTU.22293412.

Given this new set of benchmark instances, comparing the efficiency of the assignment-based [160] and the network-flow formulation [43] is now possible. As mentioned before, the network-flow formulation does not include any stability constraints and enforces zero hatch overstowage. The assignment-based formulation has thus been adjusted to follow this problem definition. [43] proposes an objective function that diverges from the makespan minimization from the literature. Unfortunately, its description is not accurate enough to be reproduced, hence, we modified the formulation to minimize the makespan as described in [160]. We have also corrected the capacity constraints since, in the original work, they were posted per container type and not for the block as a whole (the modified [43] formulation is presented in Appendix A.2).

TABLE 4.7: Runtime comparison of relaxation of the network-flow and the assignment-based formulation for the MPP without stability constraints

		[43]		[160]	
Vessel	**Ports**	**Objective**	**Time**	**Objective**	**Time**
Small	5	1544.33	5.41	1544.33	2.48
	7	1922.61	21.15	1922.61	9.77
	10	2326.74	588.16	2326.74	47.04
Medium	5	1936.94	6.16	1936.94	3.25
	7	2425.49	27.86	2425.49	12.78
	10	2326.74	588.16	2326.74	47.04
Large	5	2713.70	53.16	2713.68	16.62
	7	3376.71	1406.08	3376.60	450.90
	10	-	-	3821.90	756.61

Table 4.7 shows the solution time comparison between the two formulations. The

TABLE 4.8: Runtime comparison of the network-flow and the assignment-based formulation for the MPP without stability constraints

		[43]			[160]		
Vessel	**Ports**	**Objective**	**MIP Gap**	**Time**	**Objective**	MIP Gap	Time
Small	5	1546.17	0%	22.51	1546.17	0%	22.81
	7	1925.83	0%	344.28	1925.83	0%	77.97
	10	2328.83	0%	2295.48	2329.00	0%	1254.14
Medium	5	1939.33	0%	65.80	1939.33	0%	13.54
	7	2428.50	0%	117.70	2428.50	0%	70.60
	10	2328.83	0%	2295.48	2329.00	0%	1254.14
Large	5	3063.00	9%	3202.49	2726.00	0%	3023.26
	7	4750.00	21%	3604.85	3445.17	2%	3600.04
	10	6422.00	38%	3600.14	4210.00	5%	3600.09

first three columns indicate the vessel's size, the number of ports, and the number of instances solved. Six instances are solved for each of these combinations, which represent all the instances in the benchmark for which the ROB condition is empty (needed due to the hatch overstow constraint). The next two columns are the optimal solution and the time needed to find it. Notice that the problem is solved in the relaxed version, with continuous decision variables. As can be seen, both formulations can find optimal solutions to this relaxed problem within a reasonable time for small and medium-sized vessels. The network-flow formulation, however, is clearly underperforming, possibly due to the increasing number of arcs needed as the number of ports increases. The network-flow formulation was not able to find feasible solutions to any of the largest instances (large vessel and 10 ports), while the assignment-based formulation timed out (3600 sec.) only for two of those instances.

Table 4.8 presents the results of the two formulations without the relaxation of the decision variables. The table includes an extra column representing the gap between the returned solution and the lower bound (MIP Gap). The results resemble those of Table 4.7, where the formulations struggle as the instances increase in size.

Despite the fact that the network-flow formulation performs worse than the assignment-based formulation, further studying its application can be interesting as such formulations are well-studied for decomposition methods.

TABLE 4.9: Aggregated results of the assignment-based formulation to the complete problem with relaxed decision variables

Vessel	**Ports**	**N. Inst.**	**Sol/Opt**	**MIP Gap**	**Obj.**	**HO**	**MK**	**Time**
Small	5	18	18(18)	0%	3497.69	4.00	1575.56	12.04
	7	18	18(18)	0%	4752.69	7.61	1936.18	42.82
	10	18	18(18)	0%	4751.59	5.87	2340.71	192.68
Medium	5	18	18(18)	0%	4820.40	6.06	1964.94	17.15
	7	18	18(18)	0%	6260.96	8.61	2433.34	70.09
	10	18	18(18)	0%	5843.55	8.16	2943.79	312.38
Large	5	18	18(18)	0%	6598.71	7.39	2782.16	28.93
	7	18	18(18)	0%	8440.38	7.44	3434.99	163.23
	10	18	18(18)	0%	6862.35	6.33	4151.48	1301.05

Tables 4.9 and 4.10 report the results of the assignment-based formulation on a rich version of the CSPP, where stability and shear-force constraints are included. The

TABLE 4.10: Aggregated results of the assignment-based formulation to the complete problem

Vessel	Ports	N. Inst.	Sol/Opt	MIP Gap	Obj.	HO	MK	Time
Small	5	18	18(14)	0%	3501.44	6.39	1579.22	1228.37
	7	18	17(8)	1%	4532.12	8.18	1943.88	2923.58
	10	18	0(0)	-	-	-	-	-
Medium	5	18	15(14)	0%	4691.73	5.80	1905.07	1243.34
	7	18	10(4)	25%	17796.50	37.60	2626.50	3076.16
	10	18	2(0)	19%	6936.50	10.00	2736.50	3600.16
Large	5	18	9(0)	1%	6550.44	8.44	2583.78	3600.14
	7	18	0(0)	-	-	-	-	-
	10	18	7(0)	82%	108633.00	262.71	5290.14	3600.27

TABLE 4.11: Classification of slot planning problems

Paper	Cargo	Hydro	CSPP aspects	Obj	Sc	Solution methods
[162]	Mix	None	MinRe, RF	PS, VU	L	NeighMeta
[56]	Mix	None	MinRe, RF	PS, VU	L	Exact
[166]	Mix	None	MinRe, DG, RF	PS, VU	M	NeighMeta
[232]	Mix	None	MinRe	PS	S	Greedy/PopulMeta
[105]	Mix	None	MinRe	PS	S	Exact
[112]	Mix	None	MinRe, DG, RF	PS, VU	M	Exact
[124]	Mix	None	MinRe, RF	PS, VU	L	MatHeu
[176]	Mix	None	MinRe, RF	PS, VU	L	NeighMeta

formulation minimizes hatch overstowage and crane makespan. Both tables report aggregated results grouped by vessel size and the number of planned ports (columns 1 and 2). The tables also report the number of instances in each group (N. Inst.), the number of instances solved and which of those are optimal (Sol/Opt), mean values of the optimality gap reported by the solver (MIP Gap), the mean of the objective function (Obj.), the mean number of hatch overstows (HO), the mean of the makespan (MK), and the average runtime (Time).

It can be easily seen from the tables that as the vessel size and the number of ports increase, so does the time required by the solver to find solutions. For the version of the problem where the assignment variables are relaxed, it is possible to find optimal solutions for all the instances within the 3600 sec. time limit used in the experiments. Without the variable relaxation (Table 4.10), it is hard to find feasible solutions even for instances with a small vessel. The full set of results can be found in Appendix A.1.

Slot Planning

The goal of the SPP is to create a complete stowage plan, where containers are assigned to slots based on the outcome of the MPP. The mathematical model proposed by [166] is an enriched version of the model by [57]. It is based on the binary decision variable c_{ijk}, which indicates if a container $i \in I$ is stowed in cell $k \in K_j$ of stack $j \in J$. Set I is divided into two sets of 20′ T and 40′ containers F. The model is as follows:

SPP model

$$\min \alpha_1(|I| - \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j} c_{jki}) + \alpha_2 \sum_{j\in J}\sum_{k\in K_j} o_{jk} + \alpha_3 \sum_{j\in J}\sum_{d\in D} p_{jd} + \alpha_4 \sum_{j\in J} e_j$$

$$+ \alpha_5 \sum_{j \in J} \sum_{k \in K_j} (R_{jk} \sum_{i \in F} c_{jki}(1 - R_i^c) + \sum_{i \in T} (\frac{1}{2} R_{jk} - R_i^c)) \tag{4.22}$$

$$\frac{1}{2} \sum_{i \in T} c_{j(k-1)i} + \sum_{i \in F} c_{j(k-1)i} - \sum_{i \in F} c_{jki} \geq 0 \qquad \forall j \in J, k \in K_j - \{1\} \tag{4.23}$$

$$\sum_{i \in T} c_{jki} + \sum_{i \in T} c_{j(k-1)i} \leq 0 \qquad \forall j \in J, k \in K_j - \{1\} \tag{4.24}$$

$$\frac{1}{2} \sum_{i \in T} c_{jki} + \sum_{i \in F} c_{jki} \leq 1 \qquad \forall j \in J, k \in K_j \tag{4.25}$$

$$\frac{1}{2} \sum_{j \in J} \sum_{k \in K_j} c_{jki} \leq 1 \qquad \forall i \in I \tag{4.26}$$

$$\sum_{i' \in T} c_{jki'} - 2c_{jki} \geq 0 \qquad \forall j \in J, k \in K_j, i \in T \tag{4.27}$$

$$\sum_{i \in I} R_i^c c_{jki} - R_{jk} \leq 0 \qquad \forall j \in J, k \in K_j \tag{4.28}$$

$$\sum_{k \in K_j} \sum_{i \in I} W_i^c c_{jki} \leq W_j^s \qquad \forall j \in J \tag{4.29}$$

$$\sum_{k \in K_j} (\frac{1}{2} \sum_{i \in T} H_i^c c_{jki} + \sum_{i \in F} H_i^c c_{jki}) \leq H_j^s \qquad \forall j \in J \tag{4.30}$$

$$\sum_{d'=1}^{d-1} (\sum_{i \in F} A_{id'} c_{jk'i} + \frac{1}{2} \sum_{i \in T} A_{id'} c_{jk'i}) \leq u_{jkd} \qquad \forall j \in J, k \in K_j, k > 1, k' \in K, k' < k, d \in D, d > 1, d' \in D, d' < d \tag{4.31}$$

$$\sum_{i \in F} A_{id} c_{jki} + u_{jkd} \leq 1 + o_{jk} \qquad \forall j \in J, k \in K_j, d \in D \tag{4.32}$$

$$\frac{1}{2} \sum_{i \in T} A_{id} c_{jki} + u_{jkd} \leq 1 + \frac{1}{2} o_{jk} \qquad \forall j \in J, k \in K_j, d \in D \tag{4.33}$$

$$|K_j| e_j - (\sum_{i \in F} c_{j1i} + \frac{1}{2} \sum_{i \in T} c_{j1i}) \geq 0 \qquad \forall j \in J \tag{4.34}$$

$$|K_j| p_{jd} - \sum_{k \in K_j} (\sum_{i \in F} A_{id} c_{jki} + \frac{1}{2} \sum_{i \in T} A_{id} c_{jki}) \geq 0 \qquad \forall j \in J, d \in D \tag{4.35}$$

$$c_{jki} = 1 \qquad \forall (j,k,i) \in L \tag{4.36}$$

$$|K_j| v_{jm} - \sum_{k \in K_j} (\sum_{i \in F, M_i = m} c_{jki} + \frac{1}{2} \sum_{i \in T, M_i = m} c_{jki}) \geq 0 \qquad \forall m \in M, j \in J \tag{4.37}$$

$$v_{jm_1} + v_{(j+1)m_2} \leq 1 \qquad \forall j \leq |J| - 1, (m_1, m_2) \in IMO \tag{4.38}$$

$$v_{jm_1} + v_{jm_2} \leq 1 \qquad \forall j \leq J, (m_1, m_2) \in IMO \tag{4.39}$$

Constraints (4.23) and (4.24) guarantee that stacking rules are applied, and constraints (4.25) and (4.26) ensure compliance with cell capacities. Constraint (4.27) ensures that there are no unpaired 20′ containers in cells, and constraint (4.28) enforces that reefer containers are stowed in reefer slots. Constant R_{jk} expresses the number of reefer plugs in cell k, and R_i^c indicates if container i is a reefer. Stack

height (H_j^s) and weight (W_j^s) limits are respected in constraints (4.29) and (4.30). Constraints (4.31 - 4.33) guarantee accurate tracking of restow containers o_{jk}, for both 20′ and 40′ containers separately. Set D denotes different PODs. The auxiliary variable u_{jkd} identifies if a container below cell k of stack j has to be unloaded before port d. To accurately reflect stacks containing stowed containers, an auxiliary variable e_j is incorporated into constraint (4.34). Constraint (4.35) establishes variable p_{jd} that indicates whether at least one container in stack j must be unloaded at port d. Constraint (4.36) ensures that already loaded containers in previous ports will stay on board. Constraints (4.37 - 4.39) enforce the IMDG segregations rules, where M is the index set of IMDG categories, IMO is a set of category pairs that require segregation and auxiliary variable v_{jm} says if at least one IMDG container of category m is stowed in stack j. The objective function (4.22) has five elements, where the goal is to minimize the number of left-out containers, the number of restow containers (o_{jk}), mixing containers with different PODs (p_{jd}), number of stacks with containers (e_j), number of non-reefer containers in reefer slots.

Table 4.11 provides a summary of the literature concerning exclusively the SPP. As the table shows, some of the important constraints, e.g., hydrostatics or hatch restows, are not present in the problem formulations. Since the output of the MPP is the input for the SPP, and these constraints are already fulfilled in the MPP part of the problem, they can be ignored in this phase.

There is more consensus reached on the SPP than on the MPP. A plausible reason is that a common definition of the problem and a set of benchmark instances have been available since the publication of [57]. The proposed formulation was an inspiration to several researchers in this area. This is seen in Table 4.11, where the definitions of the problem are quite uniform. [57] considered a wide spectrum of container types, including reefers and highcubes, and important low-level constraints, including stacking rules and capacity constraints. The objective function contained several aspects of the CSPP: minimizing port stay by avoiding unnecessary crane moves, consolidation by minimizing the number of used stacks, and preserving reefer slots for reefer containers. The assumption is that all the containers from the load list can be stowed in the block. [166] modified the objective function based on the fact that not all containers can be stowed in their proposed formulation, so the aim is to load most of them. Since we need to solve the SPP for every block on the vessel separately, the computation time of the proposed approach has to be low, such that the whole process of creating a slot plan is finished in a reasonable time. A slot plan for one block should be created ideally in less than one second [56].

What is important to underline is that the model proposed by [57] considered only creating stowage plans for below-deck stacks. This made it possible to ignore constraints related to lashing, line of sight, and 45′ containers. The study of [112, 105] took the on-deck section of the vessel into account by using arbitrary height or weight limits to mimic, in a simple way, lashing constraints.

[166, 112] introduced the handling of hazardous cargo and segregation rules in the SPP. [105, 232] included port operations objectives by minimizing restow containers on the yard and crane moves while loading and discharging the vessel.

Table 4.11 shows the solution methods suggested for the SPP. [57, 56] proposed constraint programming (CP) with the usage of, among others, symmetry-breaking constraints and branching strategies to achieve better computation time. Constraint-based local search (CBLS) explored by [162] and [166] proposed a Greedy Randomized Adaptive Search Procedure (GRASP). A hybrid method involving A* and a Genetic Algorithm (GA) was developed by [232]. A* was used to find a feasible loading sequence, and GA was used to find a feasible allocation of containers to slots. A fuzzy logic algorithm with a rule-based search was presented in [176]. Additionally, a matheuristic was developed by [124]. It combined a large neighborhood search with a mathematical solver to iteratively destroy and rebuild parts of the solution.

Table 4.12 shows a comparison between all the slot planning approaches that have adopted the benchmark taken from the work of [57]. The first column indicates the instance group (we refer the reader to [56] for a detailed description), and the second the number of instances in that group. Next, the table is divided into 6 sections each representing the results of a publication: CBLS is the constraint-based local search of [163], IP is the integer programming formulation of [56] (with 10 seconds time limit), CP is the constraint programming model of [57] (with 10 seconds time limit), Fuzzy is the fuzzy logic approach of [176], Matheuristic is the matheuristic approach of [124], and GRASP is the GRASP approach of [166] (run for 1 second). For each of the publications, the table reports the percentage of feasible solutions (Sol) and optimal solutions (Opt) found, plus the time used to compute all the instances in the group. The best results are highlighted in bold.

TABLE 4.12: Slot planning methods summary

		CBLS			IP (10s)			CP(10s)		
Group	Inst	Sol	Opt	Time	Sol	Opt	Time	Sol	Opt	Time
1	13	**100**	59	0.10	**100**	**100**	1.80	**100**	**100**	0.10
2	22	**100**	77	3.60	95	91	50.40	31	91	21.60
3	13	**100**	92	0.50	92	85	35.30	**100**	**100**	0.50
4	78	**100**	92	6.00	96	94	87.00	99	99	19.70
5	36	97	58	7.10	72	56	192.00	92	92	39.00
6	15	93	80	1.20	**100**	93	13.00	**100**	**100**	5.40
7	14	93	79	2.30	64	29	102.80	64	64	53.50
8	14	93	43	1.50	79	64	74.10	93	93	10.50
9	17	94	47	5.20	53	41	112.30	88	**88**	36.50
10	8	**100**	88	0.70	88	62	31.50	**100**	**100**	0.70
11	6	50	17	1.30	67	50	30.50	83	**83**	10.30

		Fuzzy			Matheuristic			GRASP (1s)		
Group	Inst	Sol	Opt	Time	Sol	Opt	Time	Sol	Opt	Time
1	13	**100**	**100**	4.52	**100**	**100**	1.30	**100**	**100**	6.30
2	22	**100**	95	8.12	95	86	12.10	**100**	**100**	15.40
3	13	**100**	**100**	5.10	92	92	8.00	**100**	**100**	7.80
4	78	**100**	**100**	28.32	**100**	**100**	17.70	**100**	99	37.30
5	36	**100**	**94**	16.41	89	83	29.40	**100**	**94**	22.30
6	15	**100**	93	4.68	**100**	**100**	2.50	**100**	**100**	6.00
7	14	**100**	71	6.66	86	71	11.00	**100**	**93**	9.00
8	14	**100**	**100**	6.40	**100**	79	5.40	**100**	**100**	6.10
9	17	**100**	82	8.69	94	76	13.00	94	82	12.30
10	8	94	88	3.33	**100**	88	2.50	**100**	**100**	4.80
11	6	**100**	67	2.56	83	**83**	3.30	**100**	**83**	3.20

The results in Table 4.12 cannot be fully compared as experiments have been run

on different hardware. That said, the CPUs used in [124] and [176] are comparable, and even though the hardware used in [166] is older, it can be assumed that some improvement can be expected if run on modern machines. With this in mind, the table shows a clear improvement from the original work of [57, 163]. Feasible solutions have been found for all instances, and only for a few instances, optimal solutions do not exist. Given these results, this set of benchmarks seems to have achieved its purpose. In [166], it was pointed out that the benchmark contains a very limited set of discharge ports, which reduces drastically the complexity of restows, and hence propose a more challenging set of instances and a revised version of the problem, including a load maximization objective and the handling of dangerous goods. On the one hand, this new set of benchmark instances brings new challenges to the problem. On the other hand, it is less representative of the kind of instances that a slot planning problem will face when being part of a decomposition algorithm. In the latter case, it is to be expected that the master plan will ensure to have as many containers as possible with the same discharge port. The new benchmark from [166] (including the set of instances from [57]) can be found at https://doi.org/10.11583/DTU.22284475.

A different direction is taken by [112], where the focus is on the modeling of a large set (compared to that of [166]) of rules for dangerous goods. Unfortunately, the publication did not present the mathematical formulation and did not present results on the original benchmark. The new instances generated in [112] can, however, be found at https://doi.org/10.11583/DTU.22293991 and be used for future comparison.

4.4.3 Single-Port Container Stowage Planning

The high impact that a stowage plan of an earlier port can have on later ports is what dictates the inclusion of cargo forecasts and, as a consequence, the solution of multi-port versions of the problem. Single-port versions of the problem are, however, still interesting as they can be seen as more lightweight operational plans or as subproblems [56].

Table 4.13 shows the classification of the single-port studies in this review. From the table, it is easy to see a general consensus that contributions must include the modeling of several container types and some aspects of vessel stability. A noticeable exception is the work of [56, 194], where a weight distribution of the cargo is assumed to be an input to the algorithms, and the work of [236], where no explanation is given for this omission. In the studies of [189, 12, 13, 190, 7, 54, 134], vessel stability is only considered as balanced weights on the four sections of the vessel (bow, stern, port, and starboard). Works including more accurate measures are more recent [47, 91, 237, 130, 65].

The vast majority of the literature focuses on the minimization of time at port, either in the form of the time spent moving containers or in the minimization of the number of restows. Only a subset of the studies, though, includes the modeling of restows due to hatch covers [47, 56, 237, 130], only [130] and [190] includes workload distribution of the quay cranes, and a block stowage strategy is proposed in [130]. As single-port models are a simplification of multi-port models, we refrain from presenting a mathematical formulation, as it can be trivially derived by including the stability constraints from the master planning model to the slot planning formulation previously described.

All the mathematical models proposed for the single-port stowage planning problem use or adapt the formulation introduced by [12]. It is a four-index formulation indicating whether a container c is assigned to a slot in bay b, row/stack r and tier t. Exceptions are the work of [130], where symmetries in the container index are broken by the modeling of container classes, and [237], where the decision variable is split in two. The first variable assigns containers to blocks, and the second variable assigns containers to tiers within blocks, thereby effectively abstracting away the stack/row position. This formulation was able to solve problems up to 1000 TEUs, compared to the 198 TEUs of the original formulation [12]. The model by [130] is deemed intractable even for medium-sized vessels (7300 TEUs; no data is available for smaller vessels). The formulation has also been extended to integrate other problems: the blocks relocation problem [134], and barge assignment [65].

No efficient mathematical formulation or exact method has yet been found that can solve the single-port container stowage problem for real-size vessels, which explains the focus of the literature on heuristic approaches. Most solution methods rely on metaheuristics. Local search procedures that exchange containers with the aim of solving vessel stability are used by [47, 13, 134]. Ant colony optimization is proposed by [7], two genetic algorithms are introduced in [91, 65], and an adaptive large neighborhood search is presented in [130].

Construction heuristics are proposed by [189] and [56], where the latter is based on a cargo distribution obtained with a linear program. Other approaches include a heuristics branching procedure [190], a tree-search-based heuristics [236], and machine learning [194].

Though several approaches have been proposed, the lack of a common benchmark and problem definition makes it hard to compare their performance. [130] has recently published a benchmark (available at https://doi.org/10.11583/DTU.9916760) in order to address this issue. Given that most approaches have been tested on rather small instances, further research on their performance on larger instances is valuable.

TABLE 4.13: Classification of single-port container stowage planning

Paper	Cargo	Hydro	CSPP aspects	Obj	Sc	HD	Solution methods
[47]	Mix	Stab	NARe, RF	PS, VU	S		Exact/NeighMeta
[189]	Class	Equi	NARe, RF	PS	S		Exact/Greedy
[12]	Class	Equi	NARe	PS	S		Exact
[13]	Class	Equi	NARe	PS	S		Exact/NeighMeta
[190]	Class	Equi	NARe, CO	PS	S		Exact/NeighMeta
[7]	Class	Equi	NARe	PS	S		NeighMeta
[56]	Mix	None	NARe, RF	VU	L	✓	Exact/Greedy
[91]	Class	Stab	MinRe	PS, H	S		PopulMeta
[54]	Class	Equi	NARe	PS	S		Exact/Greedy
[194]	Mix	None	MinRe	PS	S		ML
[236]	Mix	None	MinRe	PS	S		TreeB
[134]	Mix	Equi	MinRe, NARe	PS	S		NeighMeta, Exact
[237]	Mix	Rich	MinRe, HR, RF	PS	S		Exact
[130]	Class	Rich	MinRe, HR, CO, BS, RF	PS, VU, H	L		NeighMeta
[65]	Mix	Stab	MinRe, La	PS, H	S		PopulMeta, Exact

4.4.4 Computational Complexity

Relatively little work has focused on the study of the computational complexity of the CSPP. The first study focused on the complexity that stability constraints such as metacentric limits had on single-stack (GM-OSOP) and multi-stack overstowage

problems (GM-MSOP) with uniform cargo [15]. A polynomial time algorithm is proved to exist for the GM-OSOP (with a time complexity of $\mathcal{O}(m^2n^3)$, where m and n refer to ports and containers respectively), while the computational complexity of the GM-MSOP is conjectured to be NP-Complete. An extension of this work is provided by [17], which presented an NP-Completeness proof based on a reduction from the C-coloring problem of circle graphs, where C represents the number of uncapacitated stacks (or the colors of the graph). The authors also proved that a polynomial time algorithm exists for $C < 4$ and provided an algorithm to calculate upper and lower bounds on the number of stacks needed to find a solution with zero shifts. Further research [211] showed that the capacitated version of the k-shift problem described in [17] can be solved in polynomial time for a fixed-sized vessel. The exponent in the polynomial is too large for any practical use, but the proof can be used to demonstrate that conclusions over experimental results conducted on a single vessel are not representative of the problem's complexity. Furthermore, [211] studied the computational complexity of the Hatch Overstow Problem (HOP) and showed that the assignment of containers over and below and hatch cover with at most k hatch overstows is NP-complete by reduction from the set covering problem.

4.4.5 Other Relevant Publications

Some studies have focused on other aspects of stowage optimization than solving the CSPP. They are presented in the following subsection.

An extension of the MPP to a selection problem was introduced by [50, 49, 113]. They considered a revenue management problem called cargo mix, where the goal was to select which bookings to accept in each port of call to maximize profit. A matheuristic was proposed in [50] composed of 3 stages: generating schedules of discharge ports by solving the longest path problem in an acyclic-directed graph, solving relaxed MIP where stability constraints are dropped and the final stage was fixing hydrostatics by possibly removing cargo. Stochastic programming was proposed by [49] by considering uncertain demand per port; a rolling horizon heuristic was introduced that decomposed the problem into sub-problems with shorter planning horizons.

The work of [101, 4, 102] introduced the Standard Capacity Model (SCM). It is a polyhedron model derived from MPP models and contributes the first linear approximations of hydrostatic equilibriums and restows. The purpose of the SCM is to increase the accuracy of cargo network-flow models such as [238] while maintaining their scalability. [102] applied the SCM to a yield optimization problem over 90 days in 2018 of Maersk's Asia - Europe service network with over 250 port calls. Optimal results could be computed in less than 30 minutes and showed that simple fixed capacity models used by carriers today can overestimate revenue by more than 20%.

The study presented in [132] proposed a multimodal deep learning model to predict the expected lashing forces for container stowage plans. Calculations for lashing forces are tedious and, for this reason, are hard to incorporate into models of the CSPP. With the use of machine learning, this process could be faster, and the presented results were promising, i.e., the average gap between predicted and true values was 0.66%.

Two interactive decision support tools were presented with the usage of Binary Decision Diagrams (BDDs) [103] and boolean satisfiability (SAT) [125]. The comparison

of both methods was presented in [126]. The software allowed for marking infeasible areas in a bay, but also suggestions of slots in which containers could be placed, and vice versa for containers and potential slots. The BDDs performed well in real-life instances.

4.5 Research Agenda

In light of this review, we present our conclusions on the state-of-the-art and propose possible areas of future research. We will do so by starting to describe the challenges with respect to problem representation, then moving on to solution methods, and finally discussing future work.

4.5.1 Representation Challenge

As mentioned in Section 4.2, the included combinatorial aspects should be representative of the real-world problem. From Section 4.4, it can be derived that a subset of these aspects has not been studied sufficiently. For instance, we believe that lashing is only modeled by dynamic stack capacity in [65], while [195] only commented on their approach. In practice, the proper use of lashing rods can significantly increase on-deck capacity, while the effects of different lashing models are yet to be investigated. In addition, a substantial body of work implements voluntary restows to reduce restows at future ports (e.g., [35, 18, 177]). Nevertheless, it remains unclear to what extent these impact port stay and vessel utilization. Similarly, block stowage patterns are limited to block purity in [226, 137, 159, 130], even though more sophisticated patterns (e.g., paired block stowage) have been adopted by the industry. Thus, future work should also investigate these best practices.

Despite the individual cases, we assess that the interaction between key combinatorial aspects is studied insufficiently. Overall, each additional constraint reduces the capacity at ports or vessels, but how these interact and jointly impact the objectives should be investigated further. To do so, a fully representative model is necessary, which is yet to be modeled for the CSPP and MPP. With respect to the SPP, most combinatorial aspects, except lashing, have been modeled adequately.

We suggest the following minimum requirements for a representative problem. The cargo model should consider 20/40 ft. lengths, standard and highcube heights, and also special cargo such as reefers and IMDG. In addition, voluntary and hatch restows represent reality well, whereas a combination of GM, trim, list and stress forces must model vessel stability. Any future work worth publishing should be aware of the issue raised from modeling stability constraints as a simple balancing of weight (e.g., [12]). Figure 4.6 shows a simple example of how such simplifications are too far from reality and cannot be used. The figure shows two equal weights, one with transversal position -1 and the other in position 2. Though the two weights are in perfect balance according to the formulation of [12], they result in an imbalance TCG due to their position within the vessel. Since then, many scientific studies reverted to the use of a center of gravity calculation (i.e. [8, 160, 50, 237]), while unfortunately, some remain oblivious to this mistake (e.g., disregarding container weights [22, 23], or disregarding the position of the weights [135, 134, 54, 112]). Furthermore, lashing forces should be included as they impact on-deck stack capacity, while incorporating crane operations and block stowage enables to evaluate and

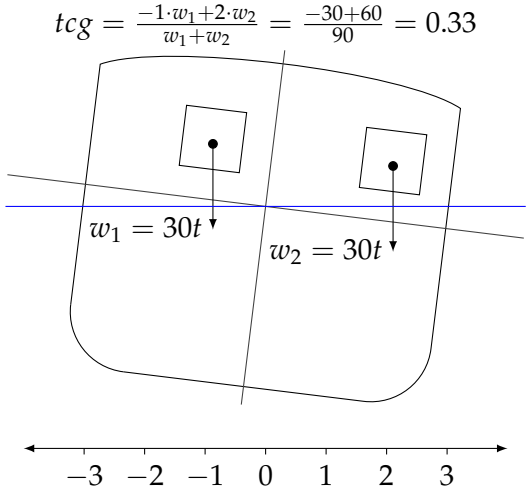


FIGURE 4.6: Example calculation of the transversal center of gravity. According to the balance constraints from [12] (where the total weight at each side of the center should be equal), the presented example is assumed to balance ($tcg = 0$), while it clearly is not the case.

enhance (un)loading efficiency. The main objectives are to minimize port stay and maximize utilization on at least 15,000 TEU vessels.

4.5.2 Solution Methods

As representative problems are scarce, the underlying problem can vary greatly. Hence, we should tread carefully before drawing any conclusions from this comparison. Even though plenty of solution methods are proposed in Figure 4.7, their experiments are often limited. In order to verify their generalizability, implementations should strive for computational studies with multiple realistic instances (e.g., [109, 160, 168]). The use or extension of benchmark instances enables such comparative studies (e.g., [18, 58, 130]). Consequently, these studies will help us to find adequate solution methods.

As in many operations research studies, articles focus on providing new mathematical formulations to a problem (e.g., [35, 12, 56]) or using those formulations to evaluate the efficiency of heuristic solution methods (e.g., [7, 158, 124]). Unfortunately, we have seen articles on the container stowage planning problem that either do not properly cite the origin of a mathematical formulation, or even present it as their own with only minor changes (if any). We encourage future authors and reviewers to be more critical, so that the literature is not overwhelmed with minor contributions that do now enhance the state-of-the-art.

4.5.3 Future Work

In contrast to the work on the k-shift problem, very little consensus can be found on a common definition of the CSPP, a set of benchmark instances or the existence of a research road map. Most of the issues related to this lack of coordination can be attributed to the lack of publicly available data and the high knowledge entry level required to truly understand the calculations behind the vessel stability constraints. It is only recently that a textbook detailing the CSPP has been published [104]. Researchers that were lucky enough to collaborate with the industry were

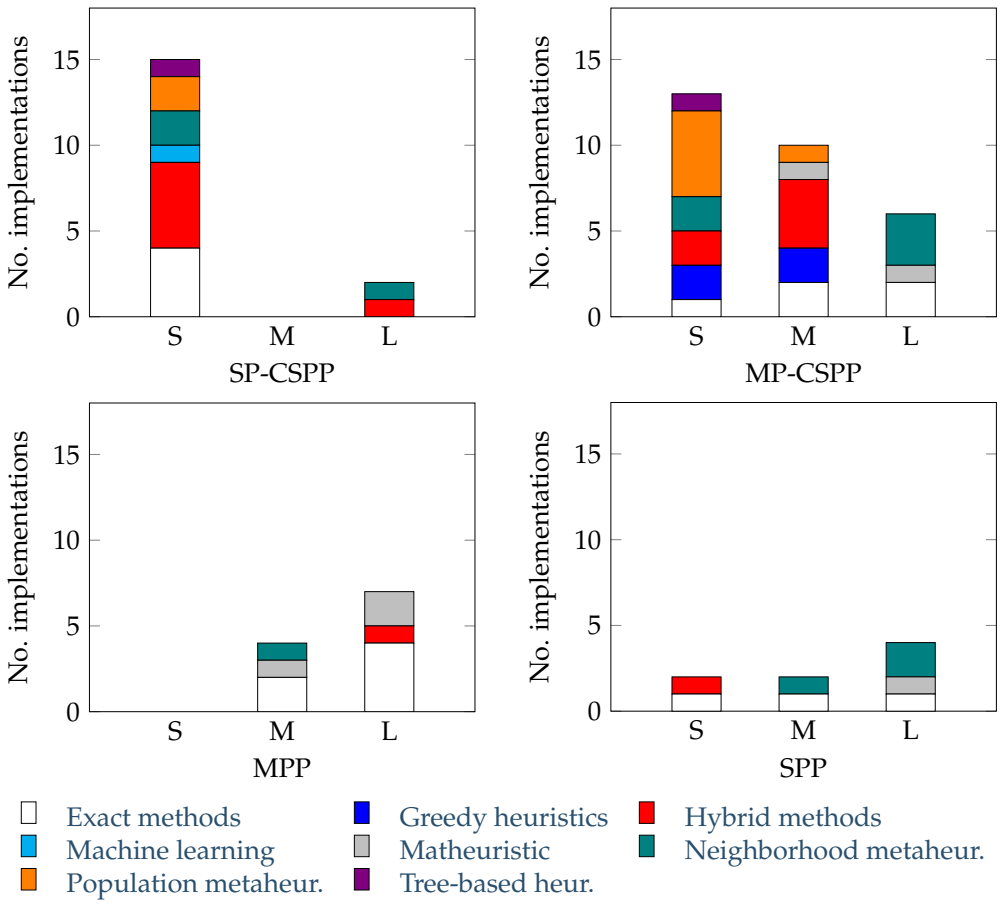


FIGURE 4.7: Number of implemented solution methods for single-port (SP-CSPP), multi-port (MP-CSPP), master bay planning (MPP) and slot planning problems (SPP) with varying vessel sizes as defined in Table 4.1 (S=Small size, M=Medium size, L=Large size).

constrained by non-confidentiality agreements from publishing details of their results (e.g., [227]) or from making available the benchmark data (e.g., [160]).

In this subsection, we suggest future work for each of the areas of research.

k-Shift and Related Problems

With respect to the benchmarks for multi-port container stowage planning, given that the solution approach of [177] provides optimal solutions within a minute, it can be concluded that the Long, Mixed, Short and Authentic instances of Table 4.2 are now closed. As the formulation of [169] is able to find feasible solutions to all instances, further research on heuristics for the *k*-shift problem does not seem to be a valuable future direction any longer.

[169] arrive at the same conclusion and hence propose to extend the *k*-shift problem with variable cargo sizes and simple stability constraints. It is shown that the additional complexity has a negative impact on the IP formulation, and hence a matheuristic approach is proposed. It is likely that similar results could be obtained by extending the work of [177] as the stability constraints would increase the number of constraints posted across the generated columns in the formulation, and hence are likely to worsen the quality of the lower-bound found by the column generation within the approach.

As future research directions, we propose the study of exact and heuristic methods for the *k*-shift CSPP with simple stability constraints. This problem corresponds to the definition provided by [169], where container types, weights and simple stability constraints are added to the original *k*-shift problem. It is unclear from the results presented in [169] whether instances based on the Short, Mixed and Long transport matrices will result in any mandatory shifts; hence merit can be given also to future research that studies or leverages the special case of the zero-shift problem.

It is the authors' opinion that future studies on exact methods are better suited as extensions of the *k*-shift problem (see [169]).

Multi-Port Container Stowage Planning

To the best of the authors' knowledge, limited progress has been made on the identification of single-phase heuristic procedures, or exact approaches, for the multi-port container stowage planning problem. In terms of exact approaches, a natural research direction is to follow the proposed agenda for the *k*-shift problem (see Section 4.5.3). As for heuristic approaches, methods that challenge the classical hierarchical decomposition or incorporate iterative elements are interesting research directions. The vessel data provided by [130], in combination with the cargo lists, which we will describe in the next section, could be used as a common benchmark for future research.

Master Planning Research on master planning is far from concluded. As a part of a hierarchical decomposition, master planning is most often solved using a relaxation of a mixed integer programming formulation (e.g., [160, 48]). Though this has positive outcomes, the method is far from infallible, and its performance is heavily dependent on the features of the specific instance and on the combinatorial aspects included in the problem.

From a problem representation point of view, combinatorial aspects such as block stowage and paired block stowage should be studied further. Only a few works have studied the impact of such stowage patterns on the achieved solutions and the performance of the solution methods (e.g., [226, 137, 159, 130]). Though the use of mathematical modeling has the flexibility of easily allowing additional side constraints to the problem, research on heuristic methods with more stable performance should also be carried out.

It is our hope that the new set of benchmark instances provided in this article (see Section 4.4.2) will increase the quality and quantity of research on this problem.

Slot Planning Thanks to the publicly available benchmarks, slot planning reached a high level of quality, and the problem, as currently defined, is (at least from an industrial point of view) solved. The benchmark, however, focuses on the SPP specific to below-deck blocks. Aspects such as lashing forces have not been explored yet.

Lashing forces are particularly interesting as little knowledge is currently available. The position of the container on deck not only depends on its weight and the general load condition but also on the type of lashing equipment available on the vessel. To which degree the mechanical calculation of the lashing forces can be simplified, and which assumption can be made to better implement solution algorithms is a field yet unexplored. The inclusion of lashing constraints is an important part of stowage planning, as a miscalculation might disallow an entire tier of a container from being loaded.

Being part of a hierarchical decomposition, slot planning has dependencies on the solution of the master planning problem. As of now, it is assumed that a master planning solution always generates feasible slot planning problems. In reality, this is not true (as shown by [160]). Hence, slot planning could be extended to include the entire vessel, thereby allowing for the flexible assignment of containers to exchange between blocks and thus improving the solution quality.

Another interesting extension of the slot planning problem is the integration with terminal operations. Some researchers have already realized this potential [152, 98], where the individual assignment of containers to container types is optimized with respect to the position of the cargo in the terminal. Other possible integrations include quay crane assignment and scheduling and container sequencing.

Single-Port Container Stowage Planning

The computational results of the single-port stowage planning (e.g., [47, 7]) are positive in terms of solution quality and computational efficiency. The impact the procedure has on today's large vessels, however, needs to be better evaluated. Most approaches are tested on small vessels for which the repositioning of a single container can have a significant effect on stability, which is no longer the case for the large vessels the industry now uses.

The KPIs mentioned by [130] are interesting when compared to some of the model enhancements presented in [237], where it was argued that containers should be stowed tier-wise rather than having tall stacks. In contrast, [162, 55, 130] argued that leaving free stacks provides a flexible stowage plan for future ports. As proposed

by [130], validation of such KPIs using simulation approaches is necessary. The vessel data mentioned in Section 4.4.3, though simplified by the authors, still presents itself with a high learning curve. Thus, it is advisable that papers studying a specific version of the CSPP derive simplified data instances. An example can be seen in Section 4.4.2, where a benchmark for the multi-port master planning problem is provided.

Research on the single-port container stowage planning problem is far from finished, and we see the following as important future research directions: the design of exact methods for the identification of optimal solutions, the evaluation of the validity and usefulness of the proposed KPIs, and the evaluation of the use of other heuristic methods, e.g., based on container exchanges as proposed in [47, 13]. Given the currently available data and experimental results, new research that does not include a full set of stability constraints is no longer of scientific interest.

4.6 Conclusion

This paper provides a review of the literature that studies the Container Stowage Planning Problem. The studies are summarized according to a classification scheme that outlines the fundamental characteristics of the problem and the applied solution approaches. As there is a lack of a common understanding of the problem characteristics, this paper provided a description of a representative problem definition based on several years of academic and industrial collaborations. In light of this definition, a research agenda is proposed for each of the major branches of research in the Container Stowage Planning Problem (single-port planning, multi-port planning, master planning, and slot planning). Moreover, this paper identifies and, in one case, provides publicly available benchmark sets in the hope that future research will make use of them as a reference point and a way to compare results. Where possible, these benchmarks have been used to compare recent research results, and provide some computational comparison. It is our hope that this survey will help improve the field and act as inspiration for future developments.

Chapter 5

Integer Programming Model

This chapter will discuss the article: *"An Efficient Integer Programming Model for Solving the Master Planning Problem of Container Vessel Stowage"* published in the proceedings of the International Conference on Computational Logistics in 2024 [220]. This study addresses sub-objectives 2, 3 and 4 of the thesis.

In Chapter 4, the need for efficient mathematical models that accurately incorporate industrial constraints is highlighted. In response, this chapter introduces a novel IP model, referred to as template planning, which integrates valid block stowage patterns, a crucial industrial constraint that has been largely overlooked in research [221]. The computational complexity of the problem is theoretically analyzed, and extensive experiments are conducted to compare the performance of template planning against a traditional allocation-based MIP model of master planning.

This chapter uses the same content as the article [220], with each section corresponding directly to a section in the original work, except for the omission of a redundant domain section. This chapter is organized as follows: Section 5.1 introduces the article, and Section 5.2 describes the related work on master planning optimization. In Section Section 5.3, the allocation and template planning models are defined, while Section 5.4 discusses the computational results of solving both models on benchmark data. Finally, Section 5.5 concludes the main findings of this study.

5.1 Introduction

Containerized shipping is the backbone of world trade. From 1980 to 2023, the volume of international seaborne trade carried by container vessels grew more than 20-fold from 100 million to 2,200 million tons [202]. It is an environmentally friendly mode of transportation that is politically prioritized [51]. From an operational point of view, however, maximizing the volume of cargo transported by a container vessel is challenging. There are several reasons for this. First, in contrast to a land depot, a container vessel is floating on water and must fulfil complex seaworthiness requirements, including draft, stability, stress forces, and lashing force limits. Second, container vessels sail on closed services between ports and are never empty. Crane moves must be distributed along the vessel such that many cranes can work in parallel to minimize the port stay [104]. Moreover, cranes can only reach containers from the top of stacks, and minimizing the number of containers that must be restowed to reach containers below them is NP-hard [17, 211]. This latter challenge

is a focal point of research in the area (e.g., [160, 237]). In practice, however, the problem is mostly avoided by stowing containers with the same port of discharge (POD) on and below deck [104]. These so-called *paired block stowage* patterns also ensure robustness to uncertain cargo in future ports by clearing as much bottom space as possible. For that reason, the patterns are an operational requirement in practice.[1] Despite its significance, there is little previous work on paired block stowage [221].

Due to the dependencies between containers loaded in each service port, maximizing the volume of transported cargo entails solving a multi-port stowage planning optimization problem. A scalable approach is to decompose the problem into a *master planning* and *slot planning* problem [221]. The master planning problem assigns containers to load over a sequence of port calls to bay sections of the vessel. Its main constraints are seaworthiness requirements and sufficient crane utilization. The slot planning problem assigns individual containers to slots in each bay section. Its main constraints are stacking and lashing rules. Experimentally, the master planning problem is the hardest to solve and the most important indicator of loadable volume.

In this paper, we focus on the master planning problem. It has been solved efficiently using MIP models with various sets of constraints [160, 9, 111] but without using paired block stowage patterns. The limited previous work that uses paired block stowage patterns (e.g., [137, 159, 130]) indicates that they are combinatorially hard. To this end, we contribute a new 0-1 IP formulation of the master planning problem that focuses on paired block stowage named *template planning*. In contrast to previous models that allocate cargo to bay sections, we only use decision variables to indicate the port of discharge of each pattern and then require sufficient capacity to load all containers.

Our experimental evaluation uses data from the representative container vessel stowage planning problem suite, i.e., the largest set of benchmark data publically available to date for representative stowage planning problems [198]. Our results show that the new formulation outperforms traditional formulations concerning the optimality gap and efficiency while preserving a sufficiently accurate representation of master planning constraints and objectives. We also contribute the first complexity result on paired block stowage. As mentioned above, previous work indicates that paired block stowage is combinatorially hard. We reduce the set partitioning problem to the template model, thus showing that searching in paired block stowage patterns is NP-hard.

5.2 Related Work

The container vessel stowage planning problem has progressively gained academic attention, emphasizing its significance in the industry [221]. Existing publications can be categorized into two main types. The first type includes theoretical works that address notable combinatorial problems [17, 211, 177, 58]. The second type consists of practical studies that focus on heuristic methods. These methods can handle the complex representations of container vessel stowage planning and could be implemented in the industry [160, 82, 137].

[1] An exception to this is small feeder vessels that are mostly stowed using the same POD in each stack rather than block patterns.

One notable approach to the container vessel stowage planning problem is the hierarchical decomposition method [227]. This method breaks down the problem into two subproblems: the multi-port master planning problem (MPP) and the slot planning problem (SPP). The former involves assigning groups of containers into blocks on the vessel, while the latter focuses on arranging containers in their slots. The practicality of this approach is evident, as slot planning can be efficiently managed using heuristics, with its impact on runtime being significantly less than that of the master planning problem.

In the following section, we will focus on the characteristics of the MPP problem and the solution methods employed to address it. The most well-known model in the literature is from Pacino et al. [160], which draws inspiration from Ambrosino et al. [12]. This model considers the vessel's seaworthiness while aiming to optimize crane work. The decision variables represent the number of containers from a particular group with identical ports of load and discharge to be stowed in a block. Because containers are classified based on their weight, the weight distribution on the vessel can be calculated with a high level of accuracy. Most models found in the literature are inspired by this mathematical formulation [11, 111, 31]. The only alternative method of formulating the master planning problem was suggested by Chao et al. [43], where a network-flow representation is used to model the problem. However, it does not consider aspects of seaworthiness. The master planning problem has primarily been tackled using mathematical solvers [160, 161, 43], matheuristics [8, 11], reinforcement learning [219], or a mixed approach of exact methods and heuristics [163, 31].

Existing problem formulations focus mostly on minimizing hatch overstowage (e.g., [137, 160, 31]) instead of enforcing paired block stowage constraints. The strategy is to cluster containers heading for the same POD into a single block to avoid unnecessary crane work and create a consolidated free space when the containers are discharged. In the work by Wilson and Roach [226], the objective function considers the paired block stowage aspect, aiming to minimize the number of hatches occupied by containers with different ports of discharge (PODs). Liu et al. [137] impose a block stowage restriction, allowing only containers with the same POD to be placed in a block. However, these block sizes are smaller than those in conventional block stowage constraints and vary for spaces above and below the hatch cover. In a more theoretical study, Pacino [159] demonstrates that including block stowage and crane intensity in container vessel stowage optimization is complex. The provided mathematical formulation could not be solved within a one-hour limit, hence an LNS-based metaheuristic is introduced to find viable solutions.

Relative to the existing body of research, we propose a new 0-1 integer programming formulation for the MPP. This formulation is unique in that it searches within the space of valid paired block stowage patterns, as opposed to the traditional approach of allocating containers to available space. Furthermore, our formulation incorporates a comprehensive set of representative problem features, including vessel capacity for various container types, crane makespan, trim, and bending moment.

5.3 Mathematical Programming Models of the MPP

In previous mathematical models for the MPP, decision variables allocate containers of different types to partitions of bays. Recall that a bay with three hatch covers has

six partitions, one over and under each hatch cover. This approach does not scale well when we require paired block stowage patterns. Our new approach is to search in the space of valid paired block stowage patterns. If the vessel has three hatch covers, we obtain two storage areas blocks per bay: the center and the wing pair. In the case of four hatch covers, we obtain three blocks: two centers and a wing pair. We refer to this model as template planning.

Recently, there have been several efforts to leverage machine learning in combinatorial optimization [27], especially reinforcement learning seems able to efficiently construct solutions to hard problems (e.g., [86, 219, 148]). While reinforcement learning has potential, its current limitations in guaranteeing optimality and feasibility for large-scale problems are challenging, as shown in [219]. Additionally, due to the relative immaturity of stowage planning, we ought to search for novel problem formulations that outperform traditional formulations [221]. Given the aforementioned considerations, we believe that utilizing mathematical programming in conjunction with well-established solvers is a worthwhile endeavour in this context.

Section 5.3.1 describes sets, parameters, and assumptions to support the mathematical models. Subsequently, we define the MIP model for allocation planning in Section 5.3.2, while the 0-1 IP model for template planning is defined in Section 5.3.3. To show the computational complexity of template planning, we reduce the set partitioning problem to template planning in Section 5.3.4.

5.3.1 Definitions and Assumptions

We introduce relevant sets and problem parameters for the MPP in Tables 5.1 and 5.2, respectively. Most definitions speak for themselves, except for the sets and parameters explained in the following subsections. These parameters are extracted from benchmark data, which will be described in Section 5.4. Additionally, the following assumptions are made to obtain a simplified version of reality:

- Cargo only includes 20 ft. and 40 ft. containers, as well as regular and reefer containers.
- Each vessel has an arrival condition, represented in the voyage as port 0. Any demand in subsequent ports must be loaded onto the vessel.
- Loading and discharge times are equal for all ports and types of cargo.
- During any voyage, ballast water tanks are constantly half full.

5.3.2 Allocation Planning Model

Here, we define a typical MIP formulation that allocates cargo to blocks with capacity, crane makespan, hydrostatics, and block stowage constraints inspired by [159]. At the core of the allocation model is the minimization of hatch overstowage, which is an NP-hard task [211]. Let $x_{i,j}^{k} \in \{0,1\}$ indicate whether block k contains cargo of transport (i,j). Let $y_{i,j}^{k,l} \in \mathbb{N}_0$ be stowed non-reefer containers and $z_{i,j}^{k,l} \in \mathbb{N}_0$ represent stowed reefer containers for transport (i,j), cargo length l and block k.

TABLE 5.1: Sets of the MPP

Ports	$p \in P = \{0, 1, 2, \ldots\}$
Ports between i and j	$p \in P_i^j = \{p \in P \mid i \leq p \leq j\}$
Transport pairs	$(i,j) \in TR = \{(i,j) \in P^2 \mid i < j\}$
Onboard transports	$(i,j) \in TR_p^{OB} = \{(i,j) \in P^2 \mid i \leq p, j > p\}$
Discharge transports	$(i,j) \in TR_p^D = \{(i,p) \in P^2 \mid i < p\}$
Load transports	$(i,j) \in TR_p^L = \{(p,j) \in P^2 \mid j > p\}$
Load or discharge transports	$(i,j) \in TR_p^M = TR_p^L \cup TR_p^D$
Vessel bays	$b \in B = \{1, 2, \ldots\}$
Blocks in bay b	$k \in BL_b = \{1, 2, \ldots, HC_b - 1\}$
Adjacent bays	$b' \in B' = \{(1,2), (2,3), \ldots, (\lvert B\rvert - 1, \lvert B\rvert)\}$
Bays on fore side of bay b	$b' \in B_b^{fore} = \{1, 2, \ldots, b\}$
Cargo length	$l \in CL = \{20', 40'\}$

TABLE 5.2: Parameters of the MPP

Hatch covers per bay (#)	$HC_b \; \forall b \in B$
Regular cargo demand (#)	$D_{i,j}^l \; \forall (i,j) \in TR, l \in CL$
Reefer demand (#)	$R_{i,j}^l \; \forall (i,j) \in TR, l \in CL$
Volume per container (TEU)	$V_l \; \forall l \in CL$
Average container weight for transport (i,j) (tonnes)	$\bar{W}_{i,j} \; \forall (i,j) \in TR$
Estimated crane operations (#)	$\hat{O}^k \; \forall k \in BL_b, b \in B$
Average longitudinal position of bays (meters)	$L_b \; \forall b \in BL$
Fore longitudinal position of bays (meters)	$F_b \; \forall b \in BL$
Longitudinal distance between L_b and $F_{b'}$ (meters)	$LD_b^{b'} \; \forall b \in B, b' \in B$
center of buoyancy (meters)	$Z_{p,b} \; \forall p \in P, b \in B$
Slot capacity (TEU)	$C_V^k \; \forall k \in BL_b, b \in B$
Reefer capacity (#)	$C_R^k \; \forall k \in BL_b, b \in B$
Weight capacity (tonnes)	$C_W^k \; \forall k \in BL_b, b \in B$
Maximum crane makespan (#)	$\overline{Y}_p \; \forall p \in P$
LCG bounds (meters)	$\underline{LCG}_p, \overline{LCG}_p \; \forall p \in P$
Bending moment bounds (Newton meters)	$\underline{BM}_p, \overline{BM}_p \; \forall p \in P$

$$\min \sum_{p \in P} \sum_{(i,j) \in TR_p^{OB}} \sum_{b \in B} \sum_{k \in BL_b} x_{i,j}^k \tag{5.1}$$

$$\text{s.t.} \sum_{j \in P_{p+1}^n} x_{p,j}^k \leq 1$$

$$\forall p \in P_0^{n-1}, k \in BL_b, b \in B \tag{5.2}$$

$$\sum_{l \in CL} y_{i,j}^{k,l} + z_{i,j}^{k,l} \leq M x_{i,j}^k$$

$$\forall (i,j) \in TR, k \in BL_b, b \in B \tag{5.3}$$

$$\sum_{b \in B} \sum_{k \in BL_b} y_{i,j}^{k,l} + z_{i,j}^{k,l} = D_{i,j}^l$$

$$\forall (i,j) \in TR, l \in CL \tag{5.4}$$

$$\sum_{b \in B} \sum_{k \in BL_b} z_{i,j}^{k,l} = R_{i,j}^l$$

$$\forall (i,j) \in TR, l \in CL \tag{5.5}$$

$$\sum_{(i,j) \in TR_p^{OB}} \sum_{l \in CL} V_l (y_{i,j}^{k,l} + z_{i,j}^{k,l}) \leq C_V^k \sum_{(i,j) \in TR_p^{OB}} x_{i,j}^k$$

$$\forall p \in P, k \in BL_b, b \in B \tag{5.6}$$

$$\sum_{(i,j) \in TR_p^{OB}} \sum_{l \in CL} z_{i,j}^{k,l} \leq C_R^k \sum_{(i,j) \in TR_p^{OB}} x_{i,j}^k$$

$$\forall p \in P, k \in BL_b, b \in B \tag{5.7}$$

$$\sum_{(i,j) \in TR_p^{OB}} \sum_{l \in CL} \bar{W}_{i,j} (y_{i,j}^{k,l} + z_{i,j}^{k,l}) \leq C_W^k \sum_{(i,j) \in TR_p^{OB}} x_{i,j}^k$$

$$\forall p \in P, k \in BL_b, b \in B \tag{5.8}$$

$$\sum_{b \in b'} \sum_{k \in BL_b} \sum_{(i,j) \in TR_p^M} \sum_{l \in CL} y_{i,j}^{k,l} + z_{i,j}^{k,l} \leq \overline{Y}_p$$

$$\forall p \in P_1^{n-1}, b' \in B' \tag{5.9}$$

$$\underline{LCG}_p \leq \sum_{b \in B} L_b \sum_{k \in BL_b} \sum_{l \in CL} \sum_{(i,j) \in TR_p^{OB}} \bar{W}_{i,j} (y_{i,j}^{k,l} + z_{i,j}^{k,l}) \leq \overline{LCG}_p$$

$$\forall p \in LP_1^{n-1} \tag{5.10}$$

$$\underline{BM}_b \leq \sum_{b' \in B_b^{fore}} LD_b^{b'} \sum_{k \in BL_{b'}} \sum_{l \in CL} \sum_{(i,j) \in TR_p^{OB}} (\bar{W}_{i,j} (y_{i,j}^{k,l} + z_{i,j}^{k,l}) - Z_{p,b'}) \leq \overline{BM}_b$$

$$\forall b \in B, p \in LP_1^{n-1} \tag{5.11}$$

The objective (5.1) expresses the desire to minimize the use of blocks. Constraint (5.2) ensures that blocks have at most one POD during the legs of the voyage. Note that a voyage with n ports has $n-1$ legs. Constraint (5.3) links the decision variables, where x-variables will equal 1 if the sum of y, z-variables are positive. Constraint (5.4) enforces that all cargo in $D_{i,j}^l$ must be loaded by y, z, whereas Constraint (5.5) enforces that all reefers in $R_{i,j}^l$ must be loaded by z.

In Constraints (5.6)-(5.8), we define block capacity constraints that act as on-off constraints based on onboard x-variables. Constraint (5.6) limits the total TEU per block

by the capacity parameter C_k^V and parameter V_l represents the TEU volume of one container with length l. Constraint (5.7) limits reefer utilization by reefer capacity C_k^R per block k. In this model, a single reefer plug is used for both 20′ and 40′ reefers because individual cells are not a concern. Constraint (5.8) limits the total weight of block k by maximum weight C_k^W and computes the expected weight of regular and reefers cargo by $\bar{W}_{i,j}$ as the average weight per TEU during transport (i, j). It is worth noting that the on-off capacities are not strictly necessary, as, e.g., $C_V^k \sum_{(i,j)\in TR_p^{OB}} x_{i,j}^k$ can be replaced by C_V^k. Nonetheless, some initial experimenting showed that the on-off capacities shorten the runtime of the allocation model.

Constraint (5.9) limits the total moves per port with upper bound $\overline{Y}_p$. Let us define the maximum crane makespan as $\overline{Y}_p = \max(Y_p, \hat{Y}_p)$, where Y_p refers to the maximum long crane provided by data and $\hat{Y}_p = \max_k(C_V^k/1.5)$ is the expected number of container moves of the largest block assuming an equal mix of 20-40 containers. Due to the estimation of crane operations $\hat{O}^k$ in template planning, this formulation is required to ensure feasibility. Details are provided in the next subsection.

In the hydrostatic Constraints (5.10) and (5.11), we quantify over load ports between port i and j, which is defined by set $p \in LP_i^j = \{p \in P \mid \sum_{(i,j)\in TR_p^L} \sum_{l\in CL}(D_{i,j}^l + R_{i,j}^l) > 0\}$. Constraint (5.10) sets the upper and lower limit on the longitudinal center of gravity (LCG) to conform with limits posed on the vessel's trim. These limits, as defined in Equations (5.12) and (5.13), depend on the displacement d_p (the weight of the containers, tanks W^{tw}, and lightship W^{lsw} at departure from port p), which could be computed for each instance since it is given that all containers from the load list have to be loaded. Subsequently, the longitudinal center of buoyancy $lcb(d_p)$ and the trim factor $trf(d_p)$ are interpolated from the hydrostatics table. Additionally, $\underline{t}$ and $\bar{t}$ are the lower and upper bounds of trim, where small instances use $-2.5, 2.5$ meters, and the rest use $-2, 2$ meters, respectively. L_b is the longitudinal position of bay b's midpoint in meters.

$$\underline{LCG}_p = d_p(lcb(d_p) - \frac{\bar{t}}{trf(d_p)}) - \sum_{b\in B} L_b(W_b^{tw} + W_b^{lsw}) \tag{5.12}$$

$$\overline{LCG}_p = d_p(lcb(d_p) - \frac{\underline{t}}{trf(d_p)}) - \sum_{b\in B} L_b(W_b^{tw} + W_b^{lsw}) \tag{5.13}$$

Constraint (5.11) sets the upper and lower limits on the bending moment. The bounds are defined by Equations (5.14) and (5.15), where $\underline{bm}_b, \overline{bm}_b$ are bounds provided by data, and W^{tw}, W^{lsw}, are the tank and lightship weight. Additionally, $LD_b^{b'}$ is a longitudinal distance between the fore endpoint of bay b and the midpoint of bay b', $Z_{p,b}$ is buoyancy force at departure from port p in bay b interpolated from Bonjean data using displacement d_p, and B_b^{fore} is a set of bays positioned on the fore side of bay b. The bending moment for bay b is calculated by summing the product of the resulting forces per bay situated on the fore side of b and their respective distances to the fore side point of bay b.

$$\underline{BM}_b = \underline{bm}_b - \sum_{b'\in B_b^{fore}} LD_b^{b'}(W_b^{tw} + W_b^{lsw}) \tag{5.14}$$

$$\overline{BM}_b = \overline{bm}_b - \sum_{b' \in B_b^{fore}} LD_b^{b'} (W_b^{tw} + W_b^{lsw}) \tag{5.15}$$

5.3.3 Template Planning Model

The primary contribution of this paper is the template planning model, which, to our knowledge, has not been previously considered. The main idea is to eliminate all decision variables except the block indicators $x_{i,j}^k$ and ensure that the blocks designated for storing containers provide sufficient capacity. This approach potentially enhances scalability. However, it is also less expressive because it does not specify exactly which and how many containers should be stowed in each block. Despite this limitation, reasonable assumptions can be made to address the issue. First, since the objective is to minimize the number of used blocks, we can assume that these blocks are fully utilized when in use. Second, although we cannot model the weight of individual containers, it is realistic to assume that each transport is characterized by a specific weight profile. These assumptions are applied to both the allocation and template planning models to enable direct comparison. With these assumptions, we can sufficiently approximate hydrostatics and crane moves for master planning. To maintain brevity, the definitions provided in Subsection 5.3.2 are also applicable in this subsection.

$$\min \sum_{p \in P} \sum_{(i,j) \in TR_p^{OB}} \sum_{b \in B} \sum_{k \in BL_b} x_{i,j}^k \tag{5.16}$$

$$\text{s.t.} \sum_{j \in P_{p+1}^n} x_{p,j}^k \leq 1 \qquad \forall p \in P_0^{n-1}, k \in BL_b, b \in B \tag{5.17}$$

$$\sum_{l \in CL} V_l (D_{i,j}^l + R_{i,j}^l) \leq M \sum_{b \in B} \sum_{k \in BL_b} x_{i,j}^k \qquad \forall (i,j) \in TR \tag{5.18}$$

$$\sum_{(i,j) \in TR_p^{OB}} \sum_{l \in CL} V_l (D_{i,j}^l + R_{i,j}^l) \leq \sum_{b \in B} \sum_{k \in BL_b} C_V^k \sum_{(i,j) \in TR_p^{OB}} x_{i,j}^k \qquad \forall p \in P \tag{5.19}$$

$$\sum_{(i,j) \in TR_p^{OB}} \sum_{l \in C} R_{i,j}^l \leq \sum_{b \in B} \sum_{k \in BL_b} C_R^k \sum_{(i,j) \in TR_p^{OB}} x_{i,j}^k \qquad \forall p \in P \tag{5.20}$$

$$\sum_{(i,j) \in TR_p^{OB}} \bar{W}_{i,j} (\sum_{l \in CL} D_{i,j}^l + R_{i,j}^l) \leq \sum_{b \in B} \sum_{k \in BL_b} C_W^k \sum_{(i,j) \in TR_p^{OB}} x_{i,j}^k \qquad \forall p \in P \tag{5.21}$$

$$\sum_{b \in b'} \sum_{k \in BL_b} \sum_{(i,j) \in TR_p^M} \hat{O}^k x_{i,j}^k \leq \overline{Y}_p \qquad \forall p \in P_1^{n-1}, b' \in B' \tag{5.22}$$

$$\underline{LCG}_p \leq \sum_{b \in B} L_b \sum_{k \in BL_b} \sum_{(i,j) \in TR_p^{OB}} \bar{W}_{i,j} C_k^V x_{i,j}^k \leq \overline{LCG}_p \qquad \forall p \in LP_1^{n-1} \tag{5.23}$$

$$\underline{BM}_b \leq \sum_{b' \in B_b^{fore}} LD_b^{b'} \Big(\sum_{k \in BL_{b'}} \sum_{(i,j) \in TR_p^{OB}} \bar{W}_{i,j} C_k^V x_{i,j}^k - Z_{p,b'} \Big) \leq \overline{BM}_b \quad \forall p \in LP_1^{n-1}, b \in B \tag{5.24}$$

The Objective (5.16) expresses that we want to use as few blocks as possible. Constraint (5.17) expresses that a block at most can be assigned to one POD at a time. Constraint (5.18) links the x-variables to the cargo demand, where at least one x-variables among the blocks must be equal to 1 if $\sum_{l \in CL} V_l (D_{i,j}^l + R_{i,j}^l)$ is positive for some transport (i,j). Constraint (5.19) enforces that there must be enough TEU capacity to fit all onboard demand, Constraint (5.20) enforces this for onboard reefers, and Constraint (5.21) ensures this for onboard weight.

In Constraint (5.22), we must assume the number of crane moves required by some block to approximate crane makespan. We could define an expected containers per transport parameter $\hat{O}_{i,j} = \sum_{l \in CL} (D_{i,j}^l + R_{i,j}^l) / \sum_{b \in B} \sum_{k \in BL_b} x_{i,j}^k \ \forall (i,j) \in TR_p^M$. However, this causes quadratic terms in Equation (5.22). Instead, we can estimate both parameters by assuming blocks are fully loaded. The minimization of block usage causes highly utilized blocks, which causes most blocks to be loaded fully. Hence, the crane workload is approximated by $\hat{O}^k = C_V^k / 1.5$, which obtains the expected number of containers per block k by assuming an equal mix of 20 and 40 ft. cargo. This approximation overestimates the moves in blocks as multiple load moves can be associated with a block. Nonetheless, it provides an estimate of how many crane moves are needed in adjacent bays.

Constraints (5.23) and (5.24) have matching interpretations as Constraints (5.10) and (5.11) in the allocation model. Nonetheless, similar to the crane makespan, we must approximate the weight in a certain block to compute hydrostatics. Since the total weight and block usage are known, one way could be to find the average weight per block by $\bar{W}_{i,j} (\sum_{l \in CL} D_{i,j}^l + R_{i,j}^l) / \sum_{b \in B} \sum_{k \in BL_b} x_{i,j}^k \ \forall (i,j) \in TR_p^{OB}$. As this also leads to a quadratic term, we again assume that blocks are fully loaded, and therefore block weights are approximated by $\bar{W}_{i,j} C_k^V$. Yet again, this overestimates the weight in blocks but also provides an approximation of the vessel's hydrostatics.

5.3.4 Template Planning is NP-hard

Even though paired block stowage ensures a plan without restows, it is still an NP-hard problem. We prove that the template planning problem is NP-hard by reducing the set partitioning problem to a decision version of it. Recall that the set partitioning problem is the task of deciding whether a given multiset S of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2. We translate a set partitioning problem to a template planning problem as follows. Let the vessel consist of $|S|$ blocks. For each element $s \in S$, there is exactly one block k with a volume capacity C_V^k equal to s. There are three ports $P = \{1,2,3\}$. In the first port, there are $\sum_{s \in S} s$ containers to load, each with a volume of one TEU. Half of the containers have POD 2, and the other half have POD 3. There are no containers to load in ports 2 and 3. The containers are assumed to have zero weight, and none are reefers. All lower bounds ($\underline{LCG}_p$,$\underline{BM}_b$) are assumed to be minus infinite. In contrast, all upper bounds ($\overline{LCG}_p$, $\overline{BM}_b$,$\overline{Y}_p$) are assumed to be plus infinite. Hence, Constraints (5.20)-(5.24) have no effect. Since a block in port 1 can only be assigned to one POD that either is 2 or 3, a feasible

solution to this template planning problem divides the blocks into two subsets with equal total capacity. Hence, if the template planning problem has a solution, so does the corresponding set partitioning problem and vice versa. Since the reduction can be done in polynomial time, we have shown that the template planning problem is NP-hard.

5.4 Results

In this section, we will provide a computational comparison between the allocation and template models. We use the representative container vessel stowage planning problem suite, the largest set of benchmark data available for representative stowage planning problems, including vessels and problem instances [198]. A vessel summary is provided in Table 5.3, whereas the test instances are summarized in Table 5.4. Additionally, the vessel data contains the hydrostatics table, from which the longitudinal center of buoyancy $lcb(d_p)$ and trim factor $trf(d_p)$ can be interpolated based on displacement d_p. The vessel data also contains the bonjean data used to interpolate the center of buoyancy $Z_{p,b'}$ based on displacement d_p, as well as the vessel weight parameters, vessel distances, maximum capacities, and hydrostatic limits. The instance data provides cargo and reefer demand, average container weight and crane makespan limits. We refer to the benchmark suite [198] for a detailed explanation. Furthermore, stowage planners need to respond to changed circumstances quickly, therefore an algorithmic runtime of longer than an hour is hard to use in practice [104].

The experiments are run on a Linux machine with AMD EPYC 7742 64-Core Processor and 256 GB memory, running on 2.25GHz/3.4 GHz. The mathematical model is implemented in Python 3.9 and solved with CPLEX 22.1.

TABLE 5.3: Vessel metrics with TEU referring to the total capacity in TEU, Reefers refers to the total reefer capacity in plugs, weight is the total weight capacity in tonnes, bays are the number of bays able to hold cargo and Hatch refers to the maximum number of hatches.

Vessel Size	TEU	Reefers	Weight	Bays	Hatch
Small	1,040	251	22,005	8	1
Medium	6,532	1,160	162,834	18	3
Large	13,482	940	274,298	22	4
Extra Large	18,854	956	342,760	24	4

TABLE 5.4: Instance metrics with # being the number of instances, ports being the average loading ports per voyage, cargo being the sum cargo demand in TEU on average, reefers being the sum of reefer demand in TEU on average, and AC being the arrival condition as the sum of onboard cargo in TEU on average. The averages are found by computing the arithmetic mean over instances.

Vessel Size	#	Ports	Cargo	Reefers	AC
Small	19	8.16	1,716	59	654
Medium	21	5.57	5,768	155	4,890
Large	16	3.06	7,153	211	6,785
Extra Large	13	5.31	17,461	248	11,222

Table 5.5 provides a computational comparison between the allocation and template model applied to the instances mentioned above. It should be mentioned that the solver times out at 3,600 seconds and then returns the objective, gap and runtime of the solution with the best optimality gap found. The optimality gap of the allocation model is consistently larger than that of the template model across all instance sets, highlighting the difficulty the allocation model faces under these specific constraints. Especially in the larger instances, the template model can find near-optimal solutions while the allocation struggles to find solutions with an optimality gap close to the 0% optimum. Moreover, the template model is significantly faster than the allocation model regarding runtime, with an average speed-up for an instance set ranging from 2 to 4.5 times. Furthermore, a steep increase in computational time is observed from small to medium, large, or extra-large instances, showcasing the complexity of solving real-life instances. Consequently, we argue that the template model scales well to industrial-sized instances, which is not the case for the allocation model.

TABLE 5.5: Comparison of allocation and template models on several sets of instances grouped by vessel size with # number of instances. Obj. represents the objective value, the duality gap is denoted by Gap (%), and Time (s) represents the runtime in seconds. The solver either accepts solutions with a duality gap of 1% or returns the best solution after a runtime limit of 3,600 seconds. All metrics are averaged across instances with the arithmetic mean.

		Allocation			Template		
Vessel Size	#	Obj.	Gap (%)	Time (s)	Obj.	Gap (%)	Time (s)
Small	19	88.68	0.21	5.88	**80.80**	**0.05**	**1.82**
Medium	21	176.90	8.49	3,371.93	**149.05**	**0.91**	**1,387.92**
Large	16	169.50	9.91	3,620.65	**163.00**	**0.99**	**802.58**
Extra Large	13	288.54	12.71	3,614.61	**248.69**	**0.97**	**1,654.78**

From Table 5.5, one may observe that the objective values between the allocation and template models are different. We can assume a linear relationship between the objective and optimality gap to compute the expected optimal objective $\mathbb{E}[Obj^*] = (1 - Gap)Obj$. Table 5.6 shows the difference between the $\mathbb{E}[Obj^*]$ of both models across each set of instances, which shows a mean absolute error μ_{AE} of at most 10% relative to either $\mathbb{E}[Obj^*]$. Additionally, the standard deviation σ_{AE} indicates a reasonable variation in the absolute error, as the coefficient of variation $CV_{AE} = \sigma_{AE}/\mu_{AE}$ remains below 1. Hence, this suggests different but similar expected optimal objective values across instances. The difference, however, is mainly due to the allocation model having more freedom to assign cargo to various blocks and the approximations for the long crane and hydrostatics in template planning, imposing slightly different constraints on the optimization problem. Nonetheless, both models adhere to paired block stowage patterns and aim to minimize block use. Hence, the objective values are within close range.

In conclusion, the template model outperforms the allocation model with respect to optimality and runtime at the cost of approximating the long crane and hydrostatics. This trade-off allows us to solve the IP model with paired block stowage patterns, capacity constraints, maximum crane makespan, trim, and bending moment.

TABLE 5.6: Comparing the expected optimal objective of allocation and template models on several sets of instances grouped by vessel size with # number of instances. The expected optimal objective is represented by $\mathbb{E}[Obj^*] = (1 - Gap)Obj$, absolute error (AE) refers to absolute difference between $\mathbb{E}[Obj^*]$ of both models with arithmetic mean μ_{AE}, standard deviation σ_{AE} and coefficient of variation CV_{AE} over instances.

Vessel Size	#	Allocation $\mathbb{E}[Obj^*]$	Template $\mathbb{E}[Obj^*]$	Absolute error μ_{AE}	σ_{AE}	CV_{AE}
Small	19	88.45	80.75	7.30	5.38	0.738
Medium	21	160.92	147.63	13.29	6.31	0.475
Large	16	153.41	161.31	9.89	8.05	0.814
Extra Large	13	249.47	246.26	3.82	3.19	0.834

5.5 Conclusion

This paper introduces a new 0-1 IP model called template planning to solve the master planning problem with paired block stowage patterns and constraints to limit capacity, crane makespan, trim, and bending moment. In previous work, MIP models allocate containers of different types to blocks on the vessel, which does not scale well if block stowage patterns are included. Instead, our so-called template planning model searches in the space of valid paired block stowage patterns.

The experiments utilize the latest and largest benchmark suite for representative stowage planning problems. Our findings indicate that the template formulation outperforms the allocation model regarding the optimality gap and runtime while preserving an adequate representation of master planning constraints and objectives. Particularly in the larger instance sets, the allocation model struggles to find near-optimal solutions within an hour of runtime, whereas the template model demonstrates scalability by finding near-optimal solutions with an average speed-up between 2 and 4.5 times per set of instances. Additionally, we reduce the set partitioning problem to the template planning model, showing that the computational complexity of searching in valid paired block stowage patterns is NP-hard.

In future work, we aim to improve the computational efficiency of both models by improving the mathematical formulation and leveraging the cutting-planes approach. Moreover, we will enhance the approximations of crane makespan and hydrostatics in the template model, thus minimizing differences between allocation and template planning.

Chapter 6

Exploring Deep Reinforcement Learning

This chapter will discuss the article: *"Towards a deep reinforcement learning model of master bay stowage planning"* published in the proceedings of the International Conference on Computational Logistics in 2023 [219]. This study addresses sub-objectives 2 and 3 of the thesis.

In Chapter 4, the need for scalable heuristic frameworks is highlighted. As an initial response, this chapter introduces an MDP formulation of master planning, where reward-scaling is used to represent both objectives and constraints. A DRL model is trained by interacting with this MDP and is subsequently compared against a traditional allocation-based MIP model for master planning.

This chapter mirrors the content of the article [219], with each section corresponding directly to its counterpart in the original work, aside from the omission of two redundant sections on the problem domain and preliminaries. The remainder of this chapter is structured as follows: Section 6.1 introduces the article, Section 6.2 outlines related work, and Section 6.3 defines the MBPP with a MIP formulation. Our MDP and PPO architecture are described in Section 6.4, while Section 6.5 compares the results of our PPO architecture with an MIP solver, and Section 6.6 concludes the main findings of this study.

6.1 Introduction

In the past century, maritime transport has become the backbone of global trade and modern consumerism. Many transported goods are shipped by container vessels of liner shipping companies. To ensure timely arrivals and resource-efficient operations, these liner companies use stowage planning to allocate containers to vessel slots at each port of the voyage. The goal is to maximize vessel utilization and minimize operational costs by creating robust stowage plans. This is an NP-hard task due to (hatch-)overstowage [211, 104], which is further complicated by the problem size (20,000 Twenty Foot Equivalent Unit (TEU) vessels visiting at least 10 ports) and combinatorial aspects as container dimensions, seaworthiness and stowage regulations, demand uncertainty and planning best practices.

Several contributions have been unable to directly solve their problem formulation (e.g., [35, 130]). Consequently, it is suggested to hierarchically decompose the

stowage problem into master bay and slot planning (e.g., [226, 160]), which is further explained in Section 6.2. Even though plenty of contributions tried, a scalable algorithm for a representative decomposed problem is yet to be found.

This paper will provide a proof of concept for a novel application of deep reinforcement learning (DRL) to solve the master bay planning problem (MBPP) as a Markov decision process (MDP). To the best of our knowledge, the only stowage contributions involving reinforcement learning (RL) relax many of the complex combinatorial aspects [194, 236]. Furthermore, DRL is used to solve optimization problems similar to MBPP [127, 86, 148, 69]. Hence, we believe that DRL implementations can contribute to solving the stowage planning problem.

In our work, we model an episodic MDP that maximizes vessel utilization and minimizes hatch-overstowage for equal-sized cargo with two weight classes while satisfying demand and location capacity, as well as ensuring longitudinal and vertical stability. We refer to Section 6.3 for details on the problem. This is not a full-featured MBPP, but rather a non-trivial problem to provide us with a proof of concept. Each episode generates a Gaussian equivalent to the Mixed instances by [18]. We have implemented a proximal policy optimization (PPO) architecture that learns an actor and critic network to find a policy that can efficiently solve the MDP. This architecture is compared against an equivalent mixed integer program (MIP) to evaluate performance.

Our experiments show that PPO can learn a policy that optimizes the objective function on a limited training budget. Subsequently, the policy can be generalized to efficiently find reasonable solutions for a non-trivial MBPP in a fraction of the MIP runtime on limited hardware. Thus, we have provided preliminary evidence for the potential of DRL in stowage planning.

6.2 Related Work

In essence, container vessel stowage planning is a multi-port problem, which aims to balance a myriad of combinatorial aspects [104]. The field can roughly be subdivided into single-port work to create light-weight operational stowage plans (e.g., [12, 7, 55, 130]), and multi-port work to generate realistic stowage plans (e.g., [35, 18, 226, 160, 177, 169]). As demonstrated by [35, 130], an exact and optimal solution to the multi-port problem is yet to be found. Consequently, [226] suggested a hierarchical decomposition into master bay planning to allocate groups of containers to general locations on the vessel (e.g., [158, 31, 43]), and slot planning to subsequently allocate containers to slots in locations (e.g., [162, 112, 124]). For a comprehensive description of hierarchical decomposition, we refer to [104]. Recently, heuristic frameworks have also gained traction as an alternative to hierarchical decomposition (e.g., [159, 130]).

Regardless, several solution methods have been proposed to solve different stowage planning problems, for instance, exact methods (e.g., [177, 237]), greedy heuristics (e.g., [18, 58]), population-based (e.g., [63, 82]) or neighborhood-based metaheuristics (e.g., [7, 159]), matheuristics (e.g., [124, 169]), tree-based methods (e.g., [22]), or hybrid frameworks (e.g., [226, 31]). To the best of our knowledge, the number of contributions related to RL in stowage planning is limited, which addressed single-port problems without key combinatorial aspects by deep Q-learning [194] and Monte

Carlo tree search [236]. Despite this variety, we are yet to find scalable algorithms that solve representative stowage planning problems.

DRL has been rather successful for various combinatorial optimization problems (e.g., 0-1 knapsack [127], capacitated vehicle routing [127, 87, 69], and job shop scheduling [86]). Similar to stowage planning with the master bay subproblem, chip design is accelerated significantly by optimizing the chip floorplanning subproblem with DRL [148]. Furthermore, actor-critic methods can mitigate known drawbacks of learning from experience and policy gradients by combining both techniques [205]. In general, actor-critic methods are relatively sample-efficient with reliable performance, from which PPO performs best in several continuous control problems [188]. Thus, we believe that PPO merits further investigation.

6.3 Problem Formulation of Master Bay Planning Problem

Given the previous sections, we can introduce the MBPP. During a voyage, the MBPP assigns groups of containers on port loadlists to vessel locations. By doing so, we effectively abstract away individual containers and slots.

The unidirectional voyage is represented by an ordered set $P = \{1, 2, ...\}$ of port calls, of which a set of all possible transport pairs $T = \{\tau = (i, j) \in P^2 \mid i < j\}$ is constructed to specify the POL and POD of cargo. At the departure from an arbitrary port p, the onboard cargo can be characterized by the set $T^{OB}(p) = \{(i, j) \in P^2 \mid i \leq p, j > p\}$. At the next port, this onboard cargo is either discharged or remains on board (*ROB*) as defined by set $T^{ROB}(p) = \{(i, j) \in P^2 \mid i < p, j > p\}$. Each group of containers has the same POL and POD $(i, j) \in T$ as well as class $k \in K$ that defines the dimensions, weight, and type of the containers in the group. Each vessel location is defined by bay $b \in B$ and deck $d \in D = \{D^O, D^H\}$, where D^O represents on-deck locations and D^H represents below-deck locations in the hold.

The capacity expressed in TEU of a location in bay $b \in B$ and deck $d \in D$ is given by $c_{b,d}$. The weight in tons per container in a group with class $k \in K$ is given by w_k. Notice that our cargo is equal-sized and specials are not taken into account. The cargo demand in TEU from port i (POL) to port j (POD) of class k is given by $q_{\tau,k}$, where $\tau = (i, j)$. Different instances are generated by sampling $q_{\tau,k}$ from a Gaussian distribution for each class k and transport τ as shown in Equation (6.1). The expected value μ is a single random instance of the Mixed instances by [18], and σ represents the standard deviation to introduce variability around μ. Hence, Gaussian equivalents of Mixed instances are generated.

$$q_{\tau,k} \sim Q_{\tau,k} = \mathcal{N}(\mu, \sigma) \; \forall \tau \in T, k \in K \tag{6.1}$$

The primary objective is to maximize vessel utilization by loading cargo that satisfies port demand, while the secondary goal is to minimize hatch overstowage and arises from efficiency best practices. To prevent safety hazards (e.g., capsizing or falling containers), vessels must have an even keel and sufficient transverse stability, measured by trim and metacentric height (GM). Which in turn are determined by the longitudinal (LCG) and vertical center of gravity (VCG) [104].

6.3.1 MIP Model of the MBPP

The following model of the MBPP is inspired by the MIP formulation of [160].

$$\max \sum_{p\in P}\left(\sum_{b\in B}\sum_{d\in D}\sum_{k\in K}\sum_{\tau\in T^{OB}(p)} f_1 x^{b,d}_{\tau,k} - \sum_{b\in B} f_2 y_{b,p}\right) \tag{6.2}$$

$$\text{s.t.} \sum_{b\in B}\sum_{d\in D} x^{b,d}_{\tau,k} \le q_{\tau,k} \qquad \forall p\in P, \tau\in T^{OB}(p), k\in K \tag{6.3}$$

$$\sum_{k\in K}\sum_{\tau\in T^{OB}(p)} x^{b,d}_{\tau,k} \le c_{b,d} \qquad \forall p\in P, b\in B, d\in D \tag{6.4}$$

$$\sum_{k\in K}\sum_{j\in P:j>p} x^{b,d}_{(p,j),k} \le M z_{b,p} \qquad \forall p\in P, b\in B, d\in D^H \tag{6.5}$$

$$\sum_{k\in K}\sum_{i\in P:i<p} x^{b,d}_{(i,p),k} \le M z_{b,p} \qquad \forall p\in P, b\in B, d\in D^H \tag{6.6}$$

$$\sum_{k\in K}\sum_{\tau\in T^{ROB}(p)} x^{b,d}_{\tau,k} - M(1-z_{b,p}) \le y_{b,p} \qquad \forall p\in P, b\in B, d\in D^O \tag{6.7}$$

$$LM(p) - TW(p)LCG^* \le v\cdot TW(p) \qquad \forall p\in P \tag{6.8}$$

$$LM(p) - TW(p)LCG^* \le -v\cdot TW(p) \qquad \forall p\in P \tag{6.9}$$

$$VM(p) - TW(p)VCG^* \le v\cdot TW(p) \qquad \forall p\in P \tag{6.10}$$

$$VM(p) - TW(p)VCG^* \le -v\cdot TW(p) \qquad \forall p\in P \tag{6.11}$$

$$TW(p) = \sum_{k\in K} w_k \sum_{\tau\in T^{OB}(p)}\sum_{d\in D}\sum_{b\in B} x^{b,d}_{\tau,k} \qquad \forall p\in P \tag{6.12}$$

$$LM(p) = \sum_{b\in B} LP_b \sum_{k\in K} w_k \sum_{\tau\in T^{OB}(p)}\sum_{d\in D} x^{b,d}_{\tau,k} \qquad \forall p\in P \tag{6.13}$$

$$VM(p) = \sum_{d\in D} VP_d \sum_{k\in K} w_k \sum_{\tau\in T^{OB}(p)}\sum_{b\in B} x^{b,d}_{\tau,k} \qquad \forall p\in P \tag{6.14}$$

The primary decision variable of our MIP model is $x^{b,d}_{(i,j),k} \in \mathbb{R}_{\ge 0}$, which is the number of TEU from port i to j of class k planned to be stowed in bay b and deck d. The secondary decision variable $y_{b,p} \in \mathbb{R}_{\ge 0}$ is the number of TEU that causes hatch overstowage in bay b at port p, while the last decision variable $z_{b,p} \in \{0,1\}$ indicates whether the hatch is opened in bay b at port p. Equation (6.2) shows that the MIP aims to maximize vessel utilization and minimize hatch overstowage, where f_1 is the gain per loaded TEU and f_2 is the cost per hatch overstow in TEU. Equation (6.3) limits the total vessel utilization by demand $q_{\tau,k}$, while Equation (6.4) limits the onboard cargo by the capacity volume $c_{b,d}$. Equation (6.5) and (6.6) connects stowed containers $x^{b,d}_{(i,j),k}$ to hatch moves $z_{b,p}$ using the big M notation, whereas Equation (6.7) computes hatch restows based on whether the cargo is placed on the to-be-opened hatch. Let us define $LCG(p) = LM(p)/TW(p)$ and $VCG(p) = VM(p)/TW(p)$ at port p, and their optimal equivalents $LCG^* = \sum_b LP_b/|B|$ and $VCG^* = \sum_d VP_d/|D|$. Equations (6.8) and (6.9) limit the difference between the actual and optimal LCG

with parameter v. Similarly, Equations (6.10) and (6.11) limit the difference between the actual and optimal VCG with parameter v. Note that Equations (6.8)-(6.11) are linear transformations of $|LCG(p) - LCG^*| < v$ and $|VCG(p) - VCG^*| < v$. Equations (6.12) - (6.14) define the total weight $TW(p)$, longitudinal moment $LM(p)$, and vertical moment $VM(p)$. Notice that LP_b and VP_d refer to the longitudinal and vertical position of locations, respectively.

6.4 Solving MBPP with Reinforcement Learning

The state $s_t \in \mathcal{S}$ at time step t is defined with pair $s_t = (u(t), q)$ and is fully observable. It consists of the vessel utilization $u(t) \in \mathbb{R}^{|B|\times|D|\times|T|\times|K|}$ and a constant voyage demand quantity $q \in \mathbb{R}^{|T|\times|K|}$. As an example $u^{b,d}_{(i,j),k}(t) = 0.05$ means that 5% of the vessel capacity is cargo stowed in bay b at deck d with POL i, POD j and class k at step t. Similarly, $q_{(i,j),k} = 0.13$ means that there is a cargo demand of 13% of the vessel capacity with POL i, POD j and class k.

At port p, the agent takes an action $a_t \in \mathcal{A}$ for every $(i,j) \in T^{OB}(p) : i = p, j = j'$. Each action is a combination of port p and future port j', as shown in Figure 6.1. The action is defined by a pair $a_t = (l(t), \tau_t)$, where $l(t) \in \mathbb{R}^{|B|\times|D|\times|K|}$ is the fraction of vessel capacity to load in bay b at deck d of class k on transport $\tau_t = (p, j')$. In Equation (6.15), we define the transition function to step $t+1$ with an input of τ_t. If cargo remains on board, then $u^{b,d}_{\tau,k}(t)$ is unchanged. If cargo should be loaded (i.e., $\tau = \tau_t$), then the utilization will become $l^{b,d}_k(t)$. Otherwise, cargo that should not be on board will be set to zero.

$$u^{b,d}_{\tau,k}(t+1) = \begin{cases} u^{b,d}_{\tau,k}(t) & \text{if } \tau \in T^{OB}(p) \setminus \{\tau_t\} \\ l^{b,d}_k(t) & \text{if } \tau = \tau_t \\ 0 & \text{otherwise} \end{cases} \qquad \forall b \in B, d \in D, \tau \in T, k \in K \tag{6.15}$$

Since agents struggle with sparsity [66], a constant reward signal is necessary for each action. Equation (6.16) defines the reward of satisfying demand with reward coefficient f_1 and input $\tau_t = (p, j')$ for current port p and future port j'. A penalty is incurred with coefficient f_3 if the agent ships more cargo than demanded. Note that $f_3 > f_1$ holds to enforce feasibility.

$$r^{DS}_{\tau_t} = \begin{cases} f_1 \sum_{b,d,k} \sum_{\tau \in T^{OB}(p)} u^{b,d}_{\tau,k}, & \text{if } \sum_{b,d} u^{b,d}_{\tau_t,k'} \leq q_{\tau_t,k'} \\ -f_3 \sum_{b,d,k} \sum_{\tau \in T^{OB}(p)} u^{b,d}_{\tau,k}, & \text{otherwise} \end{cases} \quad \forall k' \in K \tag{6.16}$$

In addition, the following terms are evaluated at $(p, j') \in T : j' = |P|$. Equations (6.17) and (6.18) enforce that the actual longitudinal or vertical center of gravity (i.e., LCG_t or VCG_t) can at most deviate v from the optimal longitudinal or vertical center of gravity (i.e., LCG^* or VCG^*) with infeasibility penalty. Given the definitions of Equations (6.12) until (6.14), let us substitute $x^{b,d}_{\tau,k}$ by $u^{b,d}_{\tau,k}$ and define $LCG_t = \frac{LM(p)}{TW(p)}$ and $VCG_t = \frac{VM(p)}{TW(p)}$. Equation (6.19) incurs an infeasibility penalty for violating capacity. While hatch overstowage is evaluated in Equation (6.20), where we penalize

hatch restows with coefficient f_2. The cost of restowing a container is lower than its revenue, and thus $f_1 > f_2$ holds.

$$r^{LS}_{(p,j')} = \begin{cases} -f_3|LCG_t - LCG^*|, & \text{if } j' = |P| \wedge |LCG_t - LCG^*| > v \\ 0, & \text{otherwise} \end{cases} \tag{6.17}$$

$$r^{VS}_{(p,j')} = \begin{cases} -f_3|VCG_t - VCG^*|, & \text{if } j' = |P| \wedge |VCG_t - VCG^*| > v \\ 0, & \text{otherwise} \end{cases} \tag{6.18}$$

$$r^{CS}_{(p,j')} = \begin{cases} -f_3 \sum_{b,d,k} \sum_{\tau \in T^{OB}(p)} u^{b,d}_{\tau,k}, & \text{if } j' = |P| \wedge \exists b' \in B, d' \in D. \\ & \quad \sum_{k} \sum_{\tau \in T^{OB}(p)} u^{b',d'}_{\tau,k} > c_{b',d'} \\ 0, & \text{otherwise} \end{cases} \tag{6.19}$$

$$r^{HO}_{(p,j')} = \begin{cases} -f_2 \sum_{b,k,d \in D^O} \sum_{\tau \in T^{ROB}(p)} u^{b,d}_{\tau,k}, & \text{if } j' = |P| \wedge \exists b' \in B, d' \in D^H. \\ & \quad \sum_{k} \sum_{i \in P: i<p} u^{b',d'}_{i,p,k} > 0 \\ -f_2 \sum_{b,k,d \in D^O} \sum_{\tau \in T^{ROB}(p)} u^{b,d}_{\tau,k}, & \text{if } j' = |P| \wedge \exists b' \in B, d' \in D^H. \\ & \quad \sum_{k} \sum_{j \in P: j>p} u^{b',d'}_{p,j,k} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{6.20}$$

Equation (6.21) defines a reward function $\mathcal{R}(s_t, a_t)$ for each step t, where the above-mentioned terms are summed. However, a final term penalizes unsatisfied demand by including the upper bound of $r^{DS}_{\tau_t}$ with parameter f_1. Hence, we maximize a reward function with an upper bound of 0.

$$r(s_t, a_t) = r^{DS}_{\tau_t} + r^{LS}_{\tau_t} + r^{VS}_{\tau_t} + r^{CS}_{\tau_t} + r^{HO}_{\tau_t} - \sum_{k,\tau \in T^{OB}(p)} f_1 q_{\tau,k} \tag{6.21}$$

Figure 6.1 illustrates an example with 4 ports of our episodic MDP. Each episode consists of $|T| = (|P|^2 - |P|)/2$ actions during a unidirectional voyage, corresponding to the number of above-diagonal elements in a matrix P^2. An episode is initialized with an empty vessel in state s_0 (i.e., u_0 is set to 0), whereas a sample $q_{\tau,k}$ is drawn from the demand distribution $Q_{\tau,k}$. During the episode, the agent traverses all ports $p \in P$ and takes an action according to policy $\pi_\theta(a_t|s_t)$ for each $\tau_t = (p, j')$ with $j' \in P : j' > p$. Figure 6.1 differentiates between rewards for each transport pair r_{τ_t} and the last action of each port $r_{(p,|P|)}$. An episode is terminated at arbitrary step t that satisfies $\bmod t/|T| = 0$ as all actions are performed. Finally, the episodic reward is provided as output.

6.4.1 Proximal Policy Optimization Architecture

This on-policy architecture consists of an actor neural network with weights θ to create policy $\pi_\theta(a_t|s_t)$ and a critic neural network with weights ω to evaluate the performance of the policy by approximating the value function $V_\omega(s_t)$. The value function estimation $V_\omega(s_t)$ is used to express the relative advantage of taking an action. Equation (6.22) defines the general advantage estimate $\hat{A}_t$ of (s_t, a_t)-pair at step t, where λ is the exponential decay rate and δ_t is the temporal difference residual

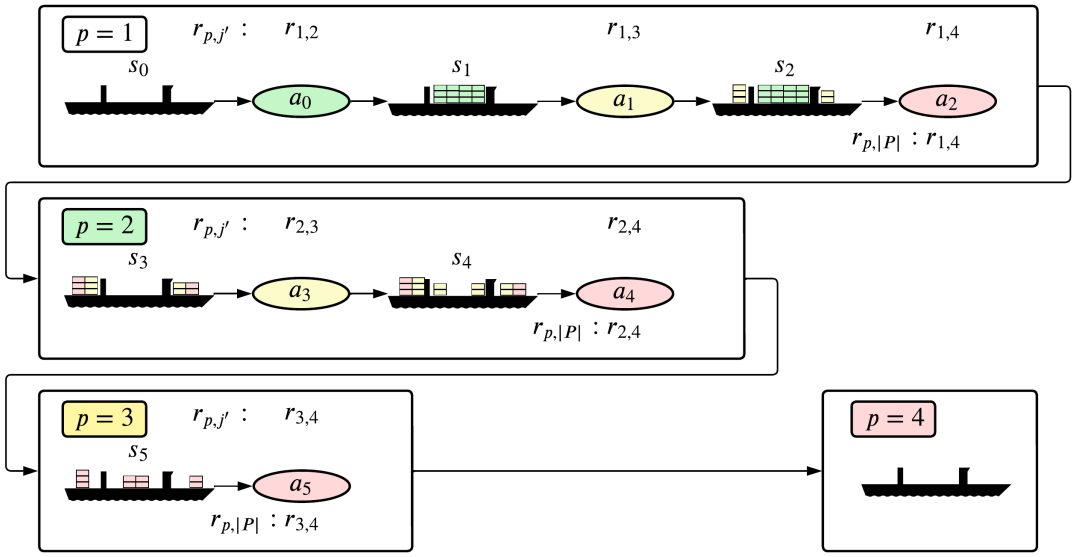


FIGURE 6.1: Overview of master bay planning MDP with colours corresponding to PODs

found by stepwise bootstrapping in Equation (6.23). Both $\pi_\theta(a_t|s_t)$ and $V_\omega(s_t)$ are multilayer perceptrons that approximate non-linear functions by learning the mean and standard deviation of continuous Gaussian distributions.

$$\hat{A}_t = \sum_{l=0}^{H-(t+1)} (\gamma\lambda)^l \delta_{t+l} \quad (6.22) \qquad \delta_t = r(s_t, a_t) + \gamma V_\omega(s_{t+1}) - V_\omega(s_t) \quad (6.23)$$

Let us define the loss function with respect to ω as the mean squared error between the estimated value function and the sum of discounted rewards in Equation (6.24). Given the estimated advantage $\hat{A}_t$, Equation (6.25) defines a clipped loss function with respect to θ, where $p_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the importance sampling ratio between the current and previous policy distribution. The clipping limit ϵ prevents disruptive changes to θ. In Equation (6.26), both functions are combined with entropy regularization, where cf_1 and cf_2 are coefficients and $S[\pi_\theta](s_t)$ is the entropy term that promotes exploration based on π_θ and s_t. Hence, Equation (6.26) is maximized to update θ and thereby policy π_θ.

$$L^{VF}(\omega) = \hat{\mathbb{E}}_t \left[(V_\omega(s_t) - \sum_{i=0}^{t} \gamma^t r(s_t, a_t))^2 \right] \quad (6.24)$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(p_t(\theta)\hat{A}_t, \text{clip}(p_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \quad (6.25)$$

$$L^{CLIP+VF}(\theta) = \hat{\mathbb{E}}_t \left[L^{CLIP}(\theta) - cf_1 L^{VF}(\omega) + cf_2 S[\pi_\theta](s_t) \right] \quad (6.26)$$

A description of PPO is shown in Algorithm 3. For each step t, let N parallel actors run policy $\pi_{\theta_{old}}$ based on parameter θ_{old} for a time horizon of H timesteps. The estimated advantage $\hat{A}_t$ is computed for every step $t \in \{1, ..., H\}$. Using a minibatch of

M steps, we optimize the actor and critic loss w.r.t. θ and ω using the Adam solver [118] for E epochs. If θ and ω are trained for a sufficient number of steps, then we can obtain an actor with $\pi_\theta \approx \pi^*$ and a critic with $V_\omega \approx V^*$.

Algorithm 3 Proximal Policy Optimization

Initialize: $\omega = \omega_0$, $\theta = \theta_0$
for $t = 1, 2, \ldots, Ts$ **do**
  for $actor = 1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{old}}$ in environment for H timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_H$
  end for
  Optimize $L^{VF}(\omega)$ w.r.t. ω with E epochs and minibatch $M \leq NH$
  Optimize $L^{CLIP+VF}(\theta)$ w.r.t. θ with E epochs and minibatch $M \leq NH$
  $\omega_{old} \leftarrow \omega$; $\theta_{old} \leftarrow \theta$
end for
return π_θ

6.4.2 Hyperparameter Tuning

There are many hyperparameters to be determined that impact the performance of PPO. We will, however, limit ourselves to the most impactful ones mentioned below. The time horizon with H steps is defined as the number of samples propagated through the network for each actor, while the number of epochs E is the number of passes through the experience buffer. The total buffer size amounts to NH and the minibatch size equals the buffer $M = NH$. The learning rate $\alpha \in (0, 1]$ defines the update size of network weights (i.e., θ, ω), which decreases over time to reduce the impact of updates. The actor and critic networks have the same number of hidden layers (depth) and neurons in each hidden layer (width). The layer activation function is ReLu to deal with vanishing gradients [76]. Both $\pi_\theta(a_t|s_t)$ and $V_\omega(s_t)$ are learned by their mean and log standard deviation that are initialized at 0 and init_log_std respectively.

We implement a tree-structured Parzen estimator using an optimization framework [5], which is a Bayesian method that can efficiently sample hyperparameters for various optimization use cases [28]. The goal is to maximize the episodic return of trials in 10^6 steps based on samples. To improve efficiency, a median pruner stops trials if its best intermediate result is worse than the median of earlier trials at the same time step [5]. In total, we run 100 trials with different hyperparameters, of which 95 trials use the pruner.

The trials are run with the following MBPP parameters: ports $|P| = 4$, classes $|K| = 2$, bays $|B| = 4$, decks $|D| = 2$, location capacity in TEU $c_{b,d} = 50$, weight class $w_k = \{1, 2\}$, absolute center of gravity tolerance $v = 0.05$, longitudinal position of bays $LP_b = \{0.25, 0.75, 1.25, 1.75\}$, vertical position of decks $VP_d = \{0.5, 1.5\}$, gain per loaded TEU $f_1 = 1$, cost per hatch overstowed TEU $f_2 = 1/3$, and penalty to violate constraints $f_3 = 3$. Given $N = 1$, the highest episodic return is obtained with $H = 512$, $E = 10$, $\alpha = 3\text{e}{-4}$, 3 hidden layers, 1024 neurons, and init_log_std $= 6.75$. Other hyperparameters are set as follows: $Ts = 4 \cdot 10^7$, $M = 512$, $\gamma = 0.99$, $\lambda = 0.95$, $cf_1 = 0.5$ and $cf_2 = 1\text{e}{-6}$.

6.5 Results

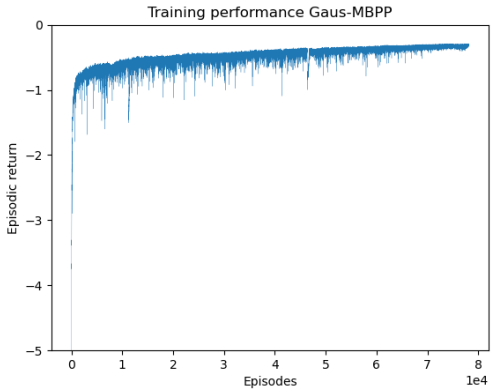
In Subsection 6.4.2, the respective environment and PPO (hyper)parameters are defined. The usefulness of PPO is demonstrated by addressing a small-scale MBPP. Even though the MBPP is not representative of real-life stowage plans, it does capture fundamental combinatorial aspects to form a non-trivial problem. The results analyze whether the policy learns to optimize the objective, but also whether the policy generalizes to new test instances. To evaluate test performance, we compare the objective value and runtime of PPO with a MIP solver on two sets of instances. The first set of 100 instances Gaus-MBPP contains Gaussian equivalents to the Mixed instances of [18]. The second set of 100 instances Unif-MBPP consists of uniform versions of the Mixed instances. Due to reward shaping, the MIP objective and the reward function are proportional but not equal, and therefore, we transform the objective before comparison.

The experiments are run on a Windows machine with an NVIDIA RTX A2000 Laptop GPU with 12.0 GB memory and an Intel Core i7-11800H processor with 8 cores and 32.0 GB memory, running at 2.3/4.6 GHz. The work is implemented in Python 3.9 and supported by libraries such as Gym 0.21 to model the environment, PyTorch 1.11 and Stable Baselines 1.6.2 to implement PPO, Optuna 3.0.3 to tune hyperparameters, and CPLEX 22.1 to solve the MIP model.

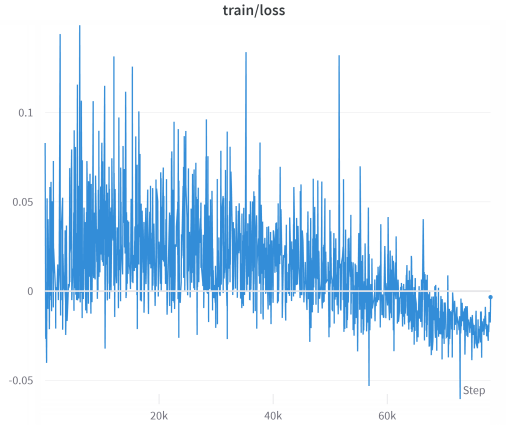
In Figure 6.2, several training metrics are included to analyze training. Figure 6.2a plots the episodic return against episodes to evaluate the training performance of PPO on Gaus-MBPP. In general, the training curve seems to converge towards an increasingly stable return around 70,000 episodes. Due to the stochasticity of Gaus-MBPP, performance will vary over time. In early training, PPO learns to avoid large negative rewards associated with infeasible solutions, after which the episodic return follows an upward trend without converging definitely. In addition, Figure 6.2b shows the loss function optimized by the actor, which has yet to converge but starts to reduce in volatility. To evaluate the difference between distributions, one can use the approximate KL divergence [66]. Figure 6.2c shows that the approximate KL divergence between $\pi_\theta(a_t|s_t)$ and $\pi_{\theta old}(a_t|s_t)$ is stabilizing. Similarly, Figure 6.2d shows that the change to $p_t(\theta)$ is lower than the clipping range ϵ, indicating small actor changes. In Figure 6.2e, we observe that log_std starts to converge, meaning that the variability of $\pi_\theta(a_t|s_t)$ reduces. Thus, we have found an increasingly stable policy with one actor on a limited training budget, suggesting there is room for improvement. Nonetheless, we have found a policy that optimizes the objective function.

In Table 6.1, we evaluate the performance of PPO and MIP on Gaus-MBPP and Unif-MBPP. Since the instance size is relatively small, the benchmark MIP will solve all instances. In comparison, PPO finds 98 feasible solutions with an average gap of 12.5% on Gaus-MBPP with low levels of variability. Even though the results are not near optimal, a larger training budget is likely to improve performance as our policy approaches the global optimum [150, 148]. Moreover, if MIP solvers become intractable for large-scale problems, the significantly shorter runtime of PPO will be advantageous. Hence, we find that PPO generalizes its policy to the Gaus-MBPP instances with reasonable performance.

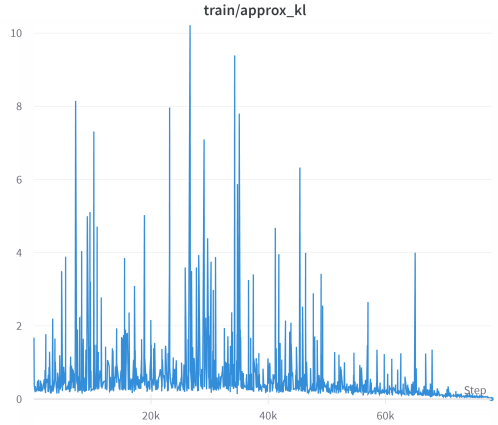
Furthermore, the policy struggles to generalize to the Unif-MBPP instances, which is not the case for the MIP. Even if feasible solutions are found, then the average



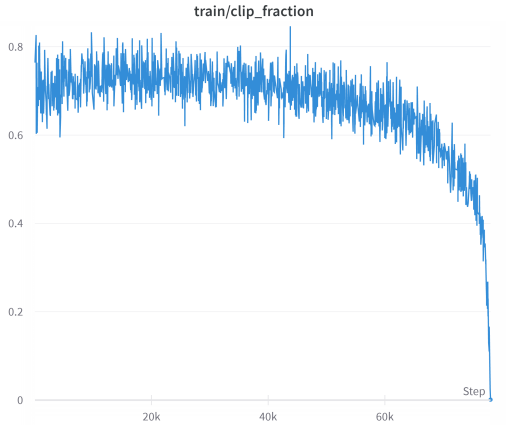
(A) Training performance.



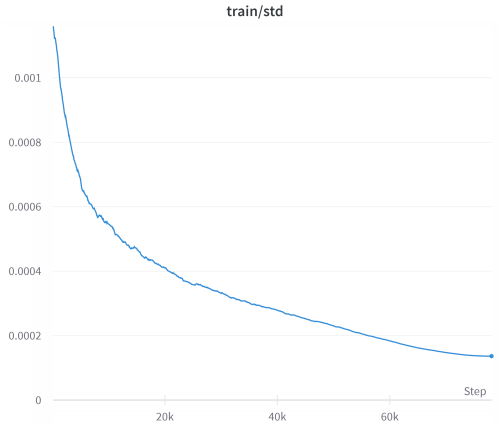
(B) Loss value $L^{\text{CLIP+VF}}(\theta)$.



(C) Approximate KL divergence.



(D) Clip fraction $p_t(\theta)$.



(E) Log standard deviation $\log \pi_\theta(a_t|s_t)$.

FIGURE 6.2: Training performance metrics on Gaus-MBPP instances

optimality gap is 81.3% with large variability in the objective value. Our function approximator struggles to accurately predict uniform instances with a different ratio than Gaus-MBPP. DRL methods generally learn to adapt to the underlying distribution, and thus it is likely that sampling outside of this distribution leads to performance loss. Though this is a clear drawback of PPO, it is our understanding that

demand often follows predictable patterns. Therefore, it is important to accurately model demand based on real data.

Although we have found encouraging results, we realize that DRL demands considerably more computational power than provided by our machine appropriate for MIP optimization. Usually, laptop GPUs have fewer cores, less VRAM, and slower clock speeds than their desktop counterparts. This causes long training times, suboptimal convergence, and reduced test performance. To train a DRL model efficiently, it is therefore recommended to use specialized hardware such as high-end GPUs, TPUs, or cloud-based computing resources. Since this is not easily accessible, we will invest in comparable hardware to a workstation with 4x NVIDIA RTX A6000 with 48 GB memory and a Threadripper Pro 3955 WX with 16 cores and 256 GB memory, running at 3.9/4.3 GHz.

TABLE 6.1: Evaluation of PPO and MIP on two sets of 100 test instances generated from Gaus-MBPP and Unif-MBPP. Results are expressed in the mean average and standard deviation metrics for the objective value (Obj.) and runtime in seconds (Time). The number of feasible solutions (#) is given without std. This also holds for the optimality gap (Gap) that is computed relative to the optimal MIP objective.

		Gaus-MBPP				Unif-MBPP			
Methods	**Metric**	**#**	**Obj.**	**Gap**	**Time**	**#**	**Obj.**	**Gap**	**Time**
PPO	Mean	98	2.092	12.5%	0.013	13	0.444	81.3%	0.006
PPO	Std		0.135		0.002		0.359		0.000
MIP	Mean	100	2.366	0.0%	0.058	100	1.930	0.0%	0.035
MIP	Std		0.261		0.022		0.387		0.016

In conclusion, PPO can learn a policy that optimizes the objective function during training on Gaus-MBPP instances. Afterward, this policy can be generalized to efficiently achieve reasonable performance on unseen Gaus-MBPP test instances. When confronted with the Unif-MBPP instances, PPO mostly obtains infeasible or weak solutions. Using this proof of concept, we argue in favor of using PPO to solve the MBPP.

6.6 Conclusion

This paper presents a DRL approach towards solving the MBPP in container vessel stowage planning. In particular, we introduce an MDP equivalent to a MIP model with NP-hard combinatorial aspects, as well as suggest PPO to solve the MDP. These preliminary experiments show that PPO learns to optimize the objective value during training on limited hardware, after which its policy can be generalized to efficiently find reasonable solutions for test instances. Hence, we have provided a proof of concept for applying PPO to the MBPP.

In future work, we will extend the environment to become a full-featured MBPP with a representative demand simulator. The algorithm will also be improved to increase performance and deal with more complex problems. Since voyages are inherently sequential, we could implement temporal dynamic policies using recurrent neural networks (e.g., long short-term memory networks). Considering the cellular shape of container vessels, we might also leverage graph representation learning in our architecture.

Chapter 7

Deep Reinforcement Learning under Uncertainty

This chapter will discuss the article: *"Navigating Demand Uncertainty in Container Shipping: Deep Reinforcement Learning for Enabling Adaptive and Feasible Master Stowage Planning"* that is currently under review [217]. This study addresses sub-objectives 2, 3 and 4 of the thesis.

In Chapter 4, the need for scalable heuristic frameworks and the incorporation of industrial challenges is highlighted. In response, this chapter introduces an MDP formulation for master planning under demand uncertainty, where the reward function seeks to maximize expected profit under explicit convex feasibility constraints. The policy employs differentiable projection layers to enforce these constraints, and we theoretically analyze their ability to minimize constraint violations. Experimental evaluations compare the proposed architecture with baseline methods from stochastic optimization and deep reinforcement learning.

This chapter uses the same content as the article [217], with each section corresponding directly to its counterpart in the original work, except for the replacement of a domain section with the problem definition and notation section. The chapter opens in Section 7.1 with an introduction to the article. Section 7.2 provides definitions and relevant notation, followed by a review of related work in Section 7.3. Next, Section 7.4 introduces both the formal and decomposed MDP formulations of the MPP under demand uncertainty. The proposed DRL architecture is described in Section 7.5, while Section 7.6 presents the results of the experimental evaluation. Finally, Section 7.7 summarizes the key findings and outlines directions for future research, whereas the appendices to support this chapter can be found in Appendix B.

7.1 Introduction

In recent years, machine learning (ML) for combinatorial optimization (CO) has gained much traction [27]. Learning advanced planning policies yields promising results in solving CO problems in transportation and logistics. Such approaches can outperform existing solution methods in some well-known problems, such as vehicle routing [123, 86] or job shop scheduling [127]. Even though it is important to benchmark algorithms on traditional CO problems, a significant opportunity remains to address lesser-known yet critical planning challenges in supply chains. By

addressing these challenges, we bridge the gap between theory and practice, enhancing the reliability and efficiency of worldwide supply chain operations.

Container shipping is a key component of the global supply chain, responsible for transporting 45% of annual goods valued at $8.1 trillion [215]. Due to its scale, it is often regarded as the cornerstone of worldwide trade and modern consumerism. However, this scale also has a significant environmental impact, with yearly CO2 emissions exceeding 200 million tonnes [138]. Despite this, container vessels are considered an environmentally friendly mode of transportation due to their relatively low emissions per cargo ton-mile [104]. Consequently, container shipping is recognized to play a crucial role in the global green transition [67]. Within container shipping, there are several interdependent and complex planning tasks, such as berth allocation [146], pre-marshalling [87], quay crane scheduling [83], and container vessel stowage planning [221]. Due to decision-making under demand uncertainty and various combinatorial aspects - such as capacity limits, seaworthiness requirements, minimizing overstowage and maximizing revenue - stowage planning is particularly challenging. To mitigate this challenge, stowage planning is often decomposed into the master planning problem (MPP), which involves the assignment of cargo to clusters of slots, and the slot planning problem (SPP), which allocates containers to individual slots [160]. However, these subproblems in representative form, particularly the MPP, remain non-trivial and difficult to solve, especially in the presence of uncertainty [221].

In addition to its complexity, stowage planning faces public data scarcity and relies heavily on human planners supported by limited decision support systems [104]. Planners should maximize capacity utilization while managing demand uncertainty by accounting for multiple future scenarios. However, evaluating such scenarios is computationally intractable, due to the problem's complexity and dynamic nature. These limitations highlight the need for efficient, adaptive, and feasible decision-support systems, offering an opportunity for AI-driven solution methods to enhance the resilience and sustainability of the global supply chain.

This paper presents a deep reinforcement learning (DRL) approach with feasibility projection to construct adaptive and feasible solutions, providing a decision-support policy for the MPP under demand uncertainty.

Our main contributions are as follows:

- **Master Stowage Planning Environment:** We develop a novel Markov decision process (MDP) for master stowage planning under demand uncertainty, incorporating realistic problem-specific constraints. To address data scarcity, we release the environment as an open-source implementation[1].
- **Feasibility Projection:** We incorporate differentiable projection layers, including weighted scaling, policy clipping, and violation projection, to enforce inequality constraint satisfaction in DRL frameworks.
- **Efficient and Adaptive Solutions:** Our experiments demonstrate that our policy efficiently generates adaptive and feasible solutions under demand uncertainty, significantly outperforming well-known DRL methods and a multi-stage stochastic MIP model.

[1] https://github.com/OptimalPursuit/navigating_uncertainty_in_mpp

- **Decision Support for Stowage Planning:** Our decision-support policy transcends deterministic models, enabling dynamic and uncertainty-informed planning in a critical part of the global supply chain.

7.2 Definitions and Notation

First, we clarify that $\otimes$ represents the outer product and $\odot$ denotes the element-wise Hadamard product, with broadcasting applied when shapes are compatible.

Voyage. A voyage can be described as a directed path graph $G_P = (P, E_P)$ with nodes being $P = \{1, 2, \ldots, N_P\}$, and edges being legs between ports $E_P = \{(p, p+1) \mid p \in \{1, 2, \ldots, N_P - 1\}\}$ with N_P being the last port. We also define sub-voyages by set $P^{\text{end}}_{\text{start}} = \{p \in P \,|\, \text{start} \leq p \leq \text{end}\}$.

Cargo. Containers have a port of load (*pol*) and a port of discharge (*pod*), also named transports or origin-destination pairs. Consider $pol \in P_1^{N_P-1}$ and $pod \in P_2^{N_P}$ with $P^2 = P \times P$, which can be represented by transport $tr = (pol, pod)$. We can define a set of transports $TR = \{(i, j) \in P^2 \mid i < j\}$. Containers are often grouped into cargo classes by characteristics, such as $K = \{20ft, 40ft\} \times \{Light, Medium, Heavy\} \times \{Spot, Long\}$.

Vessel. Bays are defined by an ordered set $B = \{1, 2, \ldots, N_B\}$, ranging from fore to aft with N_B being the last bay. Bays are horizontally separated by hatch covers into above and below deck sections, introducing the set of decks $D = \{d^{above}, d^{below}\}$.

Utilization. Vessel utilization $u_p \in \mathbb{Z}^{|B|\times|D|\times|K|\times|TR|}$ represents the cargo placement across bays B, decks D, cargo types K, and transports TR. Load operations are defined by $u_p^+ \in \mathbb{Z}_{>0}^{|B|\times|D|\times|K|\times|TR|}$, whereas discharging is denoted by $u_p^- \in \mathbb{Z}_{>0}^{|B|\times|D|\times|K|\times|TR|}$. The utilization at port p is defined as $u_p = u_{p-1} + u_p^+ - u_p^- \; \forall p \in P$, where u_0 is the vessel's arrival condition at the first port. We also define the vessel's pre-loading utilization $u'_p = u_{p-1} - u_p^-$.

Stability. The longitudinal (*lcg*) and vertical centers of gravity (*vcg*), which are associated with trim and metacentric height, of the load u_p prior to leaving port p must remain within specified bounds ($\underline{lcg}, \overline{lcg} \in \mathbb{R}_{>0}$), as shown in Constraints (7.1) and (7.2). To compute the stability, we define the longitudinal moment $lm = ld \otimes w$ and vertical moment $vm = vd \otimes w$, where $w \in \mathbb{R}_{>0}^{|K|\times|TR|}$ represents container weights, and $ld, vd \in \mathbb{R}^{|B|\times|D|}$ denote the longitudinal and vertical distances of positions from the vessel's center of gravity.

$$\underline{lcg} \leq \frac{\mathbf{1}^\top (lm \odot u_p)}{\mathbf{1}^\top (w \odot u_p)} \leq \overline{lcg}, \qquad \forall p \in P \tag{7.1}$$

$$\underline{vcg} \leq \frac{\mathbf{1}^\top (vm \odot u_p)}{\mathbf{1}^\top (w \odot u_p)} \leq \overline{vcg}, \qquad \forall p \in P \tag{7.2}$$

7.3 Related Work

This section positions our work in the literature on stowage planning, stochastic programming and ML in optimization.

Stowage Planning. To address the complexity of stowage planning, various solution approaches have been applied to different problem formulations, including exact methods [177], linearly relaxed MIP [160], matheuristics [169], population-based metaheuristics [42], neighbourhood-based metaheuristics [159], hybrid frameworks [31]. Despite these techniques, a recent survey highlights that scalable solutions for representative stowage and MPP problems remain unsolved [221]. Research has largely focused on deterministic problems, overlooking real-world needs for profit maximization in the face of demand uncertainty.

Stochastic Programming. Decision-making under uncertainty is traditionally approached through stochastic programming, where uncertainty is explicitly represented as a set of discrete scenarios evolving over multiple stages in time, approximating the underlying probability distribution [32]. This process results in a scenario tree, whose computational complexity grows exponentially as $\mathcal{O}(b^T)$, where b is the branching factor and T is the number of stages. Hence, multi-stage problems with $T > 2$ and a reasonably large b are often intractable. Common techniques to overcome this complication are scenario reduction to reduce problem size [179], decomposition techniques like Benders decomposition or Lagrangian relaxation to divide the problem [175], progressive hedging to enforce non-anticipativity iteratively [34], or approximation methods, such as stochastic dual dynamic programming [193] and sample average approximation [44], to enhance tractability while preserving solution quality.

Learning with Hard Constraints. In learning solution heuristics, a challenge arises to ensure feasibility in complex action spaces. This challenge is exacerbated when dealing with the dependence on state variables to define the feasible region. Several deep learning approaches have dealt with learning constraints by backpropagation through (in)equality completion [60], or differentiable projection layers, such as mapping interior points to boundaries [136], convex programming layers [1], or problem-specific repair layers [46]. Moreover, safe reinforcement learning has dealt with constraints by, e.g., constrained MDPs with a primal-dual approach [59], soft barriers function [225], and safety shields [6].

7.4 Markov Decision Processes

We present a formal MDP and a decomposed MDP to solve using DRL approaches. Technical details about both MDPs are provided in Appendix B.1.

7.4.1 Formal MDP

Let $\mathcal{M} = (S, X, \mathcal{T}, \mathcal{R}, P_1^{N_P-1}, \gamma)$ define an episodic discounted MDP representing the MPP, where S is a set of states, X is a set of actions, $\mathcal{T} : S \times X \to \Delta(S)$ is the transition function, $\mathcal{R} : S \times X \times P_1^{N_P-1} \to \mathbb{R}$ is the reward function, $P_1^{N_P-1}$ is the finite horizon of load ports with $N_P - 1$ being the last port, and $\gamma \in (0, 1)$ is the discounting factor.

State. The state is given by $s_p \in S$, defined as $s_p = (u_p, q_p, \zeta)$. This includes vessel utilization $u_p \in \mathbb{R}_{\geq 0}^{n_u}$ and realized demand $q_p \in \mathbb{R}_{\geq 0}^{n_q}$, where $n_u = |B| \times |D| \times |K| \times |TR|$, $n_c = |B| \times |D|$ and $n_q = |K| \times |TR|$ are the shapes of utilization, location and demand, respectively.

Additionally, environment parameters ζ contain the expected value $\mu \in \mathbb{R}^{n_q}_{\geq 0}$ and standard deviation $\sigma \in \mathbb{R}^{n_q}_{>0}$ of demand, load ports i, discharge ports j and cargo types k as $(i,j,k) \in TR \times K$, TEU per container $teu \in \{1,2\}^{n_q}$, cargo weight $w \in \mathbb{R}^{n_q}_{>0}$, cargo revenue $rev \in \mathbb{R}^{n_q}_{>0}$.

The initial state $s_0 = (u_0, q_0, \zeta)$ consists of an empty vessel $u_0 = \mathbf{0}^{n_u}$, realized demand q_0 at initial port, and ζ is initialized randomly for each episode.

Action. An action $x_p \in X$ assigns a real number of containers to utilization u_p, where $x_p \in \mathbb{R}^{n_u}_{>0}$ and thus similar to u^+_p. Each action x_p is subject to a feasible region, defined by polyhedron $PH(s_p) = \{x_p \in \mathbb{R}^{n_u}_{>0} : A(s_p)x_p \leq b(s_p)\}$. Here, $A(s_p) \in \mathbb{R}^{m_u \times n_u}$ is the constraint matrix, $b(s_p) \in \mathbb{R}^{m_u}$ is the bound vector, and m_u is the number of constraints.

Previous work demonstrated that linearly relaxed MPPs outperform exact MPPs, as the subsequent SPP effectively discretizes the solution [160]. Accordingly, we can adopt real-valued actions instead of integer-valued actions. Given a traditional MPP formulation with utilization u_p [220], we define constraints of $PH(s_p)$ in terms of actions x_p and pre-loading utilization u'_p.

$$x_p^\top \mathbf{1}_{n_c} \leq q_p \tag{7.3}$$

$$x_p teu \leq c - u'_p teu \tag{7.4}$$

$$\text{-}\mathbf{1}^\top\big((lm - \underline{lcg}w) \odot x_p\big) \leq \text{-}\mathbf{1}^\top\big((\underline{lcg}w - lm) \odot u'_p\big) \tag{7.5}$$

$$\mathbf{1}^\top\big((lm - \overline{lcg}w) \odot x_p\big) \leq \mathbf{1}^\top\big((\overline{lcg}w - lm) \odot u'_p\big) \tag{7.6}$$

$$\text{-}\mathbf{1}^\top\big((vm - \underline{vcg}w) \odot x_p\big) \leq \text{-}\mathbf{1}^\top\big((\underline{vcg}w - vm) \odot u'_p\big) \tag{7.7}$$

$$\mathbf{1}^\top\big((vm - \overline{vcg}w) \odot x_p\big) \leq \mathbf{1}^\top\big((\overline{vcg}w - vm) \odot u'_p\big) \tag{7.8}$$

Constraint (7.3) limits load x_p to the available demand q_p, while Constraint (7.4) ensures x_p does not exceed the residual TEU capacity. Constraints (7.5)–(7.6) enforce lcg limits, while Constraints(7.7)–(7.8) impose similar bounds on the vcg. These stability bounds are derived from Constraints (7.1) and (7.2).

Transition. We use a stochastic transition function $\mathcal{T}(s_{p+1}|s_p, x_p) \in \Delta(S)$. Within an episode, the transition consists of multiple components:

- Upon port arrival, port demand q_p is revealed.
- Subsequently, onboard cargo is discharged $u_{p+1} = u_p \odot (1 - \mathbf{e}^-_p)$, where $\mathbf{e}^-_p \in \{0,1\}^{n_q}$ is a binary mask indicating the cargo type and transport to nullify in u_p.
- Finally, cargo is loaded onboard $u_{p+1} = u_p + x_p$. Action x_p is based on the current utilization u_p and revealed demand q_p of port p. Future port demand stays unknown.

Reward. Equation (7.9) defines our deterministic reward function, computing profit as the difference between revenue and costs. Revenue is computed as the sum of x_p corresponding to elements of q_p. Since revenue cannot be obtained from containers exceeding demand, the summation of x_p is restricted to the elements within q_p.

While costs are computed via the state-dependent auxiliary variables hatch overstows $ho(s_p, p) \in \mathbb{R}_{>0}^{|B|}$ and excess crane moves $cm(s_p, p) \in \mathbb{R}_{>0}^{|B|-1}$. These costs are weighted by $ct^{ho} \in \mathbb{R}_{>0}$ and $ct^{cm} \in \mathbb{R}_{>0}$, respectively.

$$\mathcal{R}(s, x, p) = rev \min\left(x^\top \mathbf{1}_{n_c}, q\right) - ct^{ho}\mathbf{1}^\top ho(s, p) + ct^{cm}\mathbf{1}^\top cm(s, p) \tag{7.9}$$

7.4.2 Decomposed MDP

The formal MDP defines an effective action space of size $|X| \propto |B| \cdot |D| \cdot |K| \cdot |P|$, where each action x_p determines how cargo types and transport options are placed on the vessel per port. However, this action space is large, which can hinder learning efficiency [108].

To address this, we decompose the formal MDP into granular, sequential steps based on an index (i, j, k) $\forall (i, j) \in TR, k \in K$. Instead of placing all transport and cargo types simultaneously, we take a decomposed action for all transports (p, j) and cargo types k at port p, then departing to a new port. This reduces the action space to $|X| = |B| \cdot |D|$, while unfolding transports and cargo types over an extended time horizon $t \in H = \{0, 1, \ldots, T_{seq}\}$ with $T_{seq} = |K| \cdot |TR|$.

State. The state $s_t = (u_t, q_t, \zeta)$ depends on time step t, where $u_t \in \mathbb{R}_{\geq 0}^{n_u}$ is vessel utilization, and $q_t \in \mathbb{R}_{\geq 0}^{n_q}$ is realized demand. The environment parameter ζ remains unchanged. Given the time t, however, we can extract relevant parameters from ζ, such as (pol_t, pol_t, k_t), $rev^{(pol_t, pod_t, k_t)}$.

Action. Action $x_t \in \mathbb{R}_{\geq 0}^{n_c}$ assigns real number of containers to utilization u_t for step t. Each action is subject to $PH(s_t) = \{x_t \in \mathbb{R}_{>0}^{n_c} : A(s_t)x_t \leq b(s_t)\}$. Here, $A(s_t) \in \mathbb{R}^{m_c \times n_c}$ is the constraint matrix, $b(s_t) \in \mathbb{R}^{m_c}$ is the bound vector, and m_c is the number of constraints. It is also worth noting that Constraints (7.3)-(7.8) are reformulated to fit feasible region $PH(s_t)$.

Transition. At each time step t, the transition includes loading, where x_t is added to u_t. However, discharging and demand realization occur only when arriving at a new port, indicated by $t \in T_{\text{new port}}$.

Reward. The revenue at step t is computed as $rev(pol_t, pod_t, k_t) \min(\mathbf{1}^\top x_t, q_t^{(pol_t, pod_t, k_t)})$. However, costs depend on knowing all loading operations at port p, which is aggregated in utilization u_t at the last step of the port $t \in T_{\text{leave port}}$. As a result, the cost signal is sparse, being evaluated only once per port p rather than at each step.

7.5 Proposed Architecture

Our approach consists of several components: an encoder-decoder model parameterized by θ, an actor-critic DRL method, and a feasibility projection layer. The encoder-decoder model shown in Figure 7.1 employs a look-ahead policy $\pi_\theta(x|s_t)$ conditioned on state s_t and parameterized by mean $\mu_\theta(s_t)$ and standard deviation $\sigma_\theta(s_t)$. By training in the decomposed MDP and optionally projecting the policy, we iteratively generate actions x_t to construct solutions.

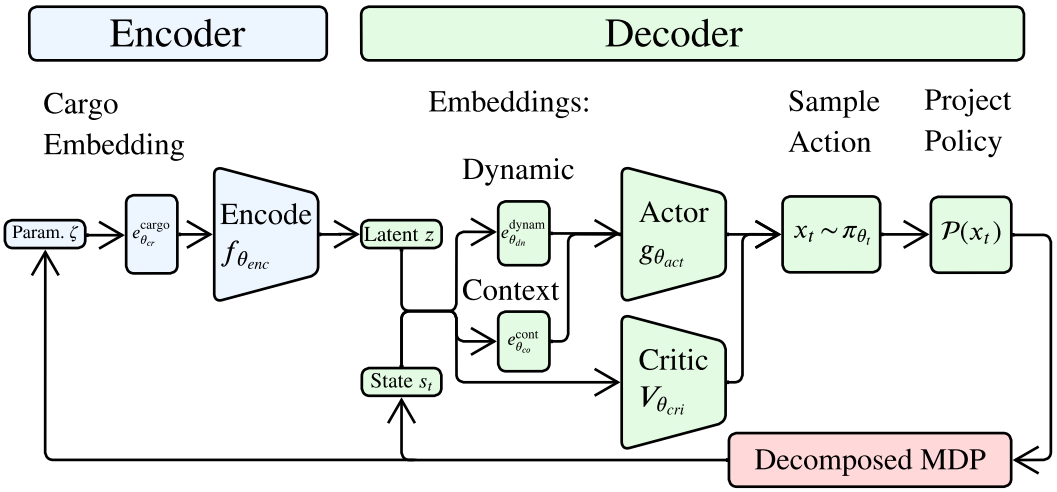


FIGURE 7.1: Deep reinforcement learning architecture with feasibility projection for actor-critic methods

7.5.1 Encoder-Decoder Model

Figure 7.2 presents our encoder-decoder model, based on that of [123], with modifications highlighted below.

Cargo Embedding. The cargo embedding $e^{\text{cargo}}(\zeta)$ parameterized by θ_{cr} maps cargo-related episode information in ζ into a feature representation for the encoder. To enhance positional awareness, we subsequently apply sinusoidal positional encoding [222].

Encoding Layers. An attention encoder $f(e^{\text{cargo}}(\zeta))$ parameterized by θ_{enc} maps embedding $e^{\text{cargo}}(\zeta)$ to latent variable z using multi-head attention (MHA) to identify revalant features dynamically [222]. Then, we use a feed-forward network (FFN) with ReLU activation, layer normalization, residual connections, and dropout.

Context Embedding. The context embedding $e^{\text{cont}}(u_t, z)$, parameterized by θ_{co}, focuses on time t, extracting time-specific features from the utilization u_t and latent variable z. These features provide the MHA query with a representation of the vessel's current condition, enabling the policy to make decisions based on the present context.

Dynamic Embedding. The dynamic embedding $e^{\text{dynam}}(q_t, z)$, parameterized by θ_{dn}, extracts features from demand q_t in all steps of horizon H, capturing patterns across the entire episode. It combines real demand and latent information to produce keys and values for the MHA layer. As such, it accounts for global temporal trends, allowing it to anticipate future conditions and make long-term decisions.

Actor Layers. As proposed in [123], our actor decoder $g(e^{\text{cont}}(u_t, z), e^{\text{dynam}}(q_t, z))$ is an attention model parameterized by θ_{act}. However, our actor takes input from the context and dynamic embedding. An MHA layer with a forward-looking mask bases decisions on steps $t, t+1, \ldots, T_{seq}$ to anticipate future events. Subsequently, an FFN extracts features, after which a pointer mechanism performs a soft selection of relevant steps using key-value inputs [224]. A softplus activation outputs positive logits $(\mu_\theta(s_t), \sigma_\theta(s_t))$.

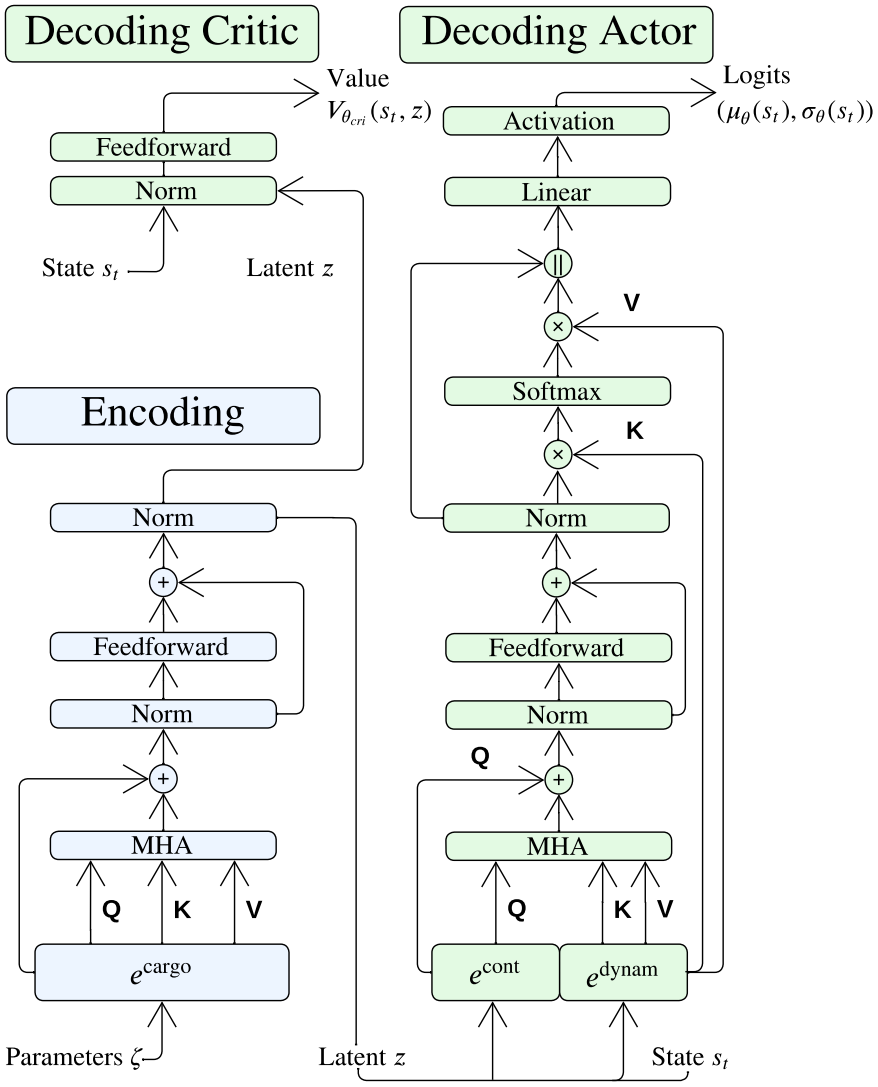


FIGURE 7.2: Layers of the encoder and the actor-critic decoder

Critic Layers. Our critic model $V(s_t, z)$, parameterized by θ_{cri}, estimates the value of state s_t and latent variable z through an FFN outputting $V_{\theta_{cri}}(s_t, z) \in \mathbb{R}$.

Action Policy. The actor logits parameterize a stochastic policy $\pi_\theta(x|s_t) = \mathcal{N}((\mu_\theta(s_t), \sigma_\theta(s_t))$, which allows action sampling $x_t \sim \pi_\theta(x|s_t)$ to construct solutions.

7.5.2 Feasibility Regularization in Actor-Critic Loss

Standard actor-critic methods can efficiently and stably learn policies that require implicit feasibility in the action space [188, 78], while explicit constraints are often integrated by feasibility regularization (FR) in the actor loss [59, 40]. Equation (7.11) defines a composite loss that combines the actor loss $\mathcal{L}_{\text{actor}}(\theta)$ with a soft regularization term $\mathcal{L}_{\text{feas}}(\theta)$. FR is defined in Equation (7.11), where λ_f controls the regularization strength, and policy samples $x_\theta(s_t)$ are assumed to allow gradient flow to update parameters θ, as is the case in SAC [78]. For algorithms that do not include actions in the computation graph, e.g., PPO [188], we substitute $x_\theta(s_t)$ with $\mu_\theta(s_t)$.

Determining Lagrangian multipliers (λ_f) for multiple constraints with varying scales is already challenging for static feasible regions. It requires costly hyperparameter tuning to balance a trade-off between objectives and feasibility effectively. As a result, FR is a naive approach when applied to dynamic, state-dependent feasible regions $PH(s_t)$ [40, 60]. However, FR remains useful as a comparative baseline.

$$\mathcal{L}(\theta) = -\mathcal{L}_{\text{actor}}(\theta) + \lambda_f \mathcal{L}_{\text{feas}}(\theta) \tag{7.10}$$

$$\mathcal{L}_{\text{feas}}(\theta) = \mathbb{E}_t \left[(A(s_t)x_\theta(s_t) - b(s_t))_{>0} \right] \tag{7.11}$$

7.5.3 Feasibility Layers

To extend beyond FR, we leverage differentiable projection layers for constraint minimization. We incorporate two constraint-specific and one general layer(s). Note that a single action may be infeasible, while combining such actions can achieve feasibility. Therefore, we prioritize violation minimization over strict enforcement to allow for flexibility.

Furthermore, non-linear transformations of continuous distribution samples change their probability density. We account for this transformation using Jacobian adjustments to the policy's log probabilities [33].

Technical details on feasibility layers are provided in Appendix B.2 of supplementary material.

Weighted Scaling Projection. Function (7.12) defines the weighted scaling layer (WS), which normalizes vector x if the sum of x exceeds scalar y. This preserves relative proportions of elements in x while enforcing x to sum to scalar y.

$$\mathcal{W}(\mathbf{x}, y) = \begin{cases} y\frac{\mathbf{x}}{\mathbf{1}^\top \mathbf{x}} & \text{if } \mathbf{1}^\top \mathbf{x} > y \\ \mathbf{x} & \text{otherwise} \end{cases} \tag{7.12}$$

Policy Clipping. We can apply function $\mathcal{C}(x, lb_{pc}, ub_{pc}) = \max(\min(x, ub_{pc}), lb_{pc})$ to enforce element-wise limits on vector x, thereby performing policy clipping (PC) on actions. However, PC is only applicable to box constraints.

Violation Projection. In Algorithm 4, we define a violation projection (VP) layer that reduces inequality constraint violations by shifting point x closer to the feasible region of some convex polyhedron $PH = \{x \in \mathbb{R}^n_{>0} : Ax \leq b\}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The inequality $Ax \leq b$ ensures PH is convex, enabling gradient-based reduction of violation $\mathcal{V}(x)$ [36]. To measure feasibility, we compute the element-wise violation $\mathcal{V}(x) = (Ax - b)_{>0}$, where each $\mathcal{V}(x)_{m_i} > 0$ indicates the violation of constraint $_{m_i}$, and $\mathcal{V}(x)_{m_i} = 0$ indicates satisfaction of that constraint. To minimize violations, we iteratively update x by applying gradient descent to reduce the violation term $\|\mathcal{V}(x)\|_2^2$, representing the squared distance from x to PH. Equation (7.13) updates x with a step size η_v, which is rewritten to the update function found in Algorithm 4.

$$x' = x - \eta_v \nabla_x \|\mathcal{V}(x)\|_2^2 \tag{7.13}$$

During training, the VP layer continues for a fixed number of epochs. However,

Algorithm 4 Violation projection layer

Require: $x \in \mathbb{R}^n_{>0}$; parameters (A, b, η_v, δ_v)
Initialize $x' \leftarrow x$
Define $\mathcal{V}(x) \leftarrow (Ax - b)_{>0}$
for $i = 1$ **to** epochs **do**
    $x \leftarrow x'$
    $x' \leftarrow x - \eta_v A^\top \mathcal{V}(x)$
    if $\mathbf{1}^\top \mathcal{V}(x') - \mathbf{1}^\top \mathcal{V}(x) \leq \delta_v$ **then**
        break
    end if
end forreturn x'

we include a stopping criteria during inference, which continues until the change in total violation $\mathbf{1}^\top \mathcal{V}(x') - \mathbf{1}^\top \mathcal{V}(x)$ is below a threshold δ_v. As a result, we x's distance to the feasible region and incorporate constraint awareness into initially unconstrained policies.

7.6 Experimental Results

We compare our policies against baselines on in-distribution and out-of-distribution instances, evaluating the objective value, computational cost and feasibility. Additionally, an ablation study analyzes the impact of feasibility mechanisms and gradient flow from actions, while managerial insights discuss the value of information, computational cost, and adaptiveness to varying levels of uncertainty.

7.6.1 Experimental Setup

Instance Generation. Training instances are sampled from a Gaussian distribution $\mathcal{N}(\mu^{(i,j,k)}, \sigma^{(i,j,k)})\ \forall (i,j) \in TR, k \in K$, where $\mu^{(i,j,k)}$ is randomly generated and $\sigma^{(i,j,k)} = CV\mu^{(i,j,k)}$ controls spread via the coefficient of variation CV. A continuous uniform distribution generates out-of-distribution instances $\mathcal{U}(lb^{(i,j,k)}, ub^{(i,j,k)})\ \forall (i,j) \in TR, k \in K$ to test generalization. Details are in Appendix B.3.

Feasibility Implementation. We evaluate feasibility by integrating multiple mechanisms to address all constraints in $PH(s_t)$. Starting with a baseline using FR, we introduce two alternatives: a constraint-specific (FR/WS/PC) and a general projection approach (FR/VP). We then remove FR from both to assess its impact. WS and PC are not analyzed separately, as it leaves many constraints unaddressed.

SMIP Baseline. We compare our approach against two common techniques in stochastic programming: stochastic MIP without anticipation (SMIP-NA) and stochastic MIP with perfect foresight (SMIP-PI), serving as an upper bound. The scenario trees and MIP models are shown in Appendix B.4.

Runs. Training occurs offline on simulated instances for a 1,000 TEU vessel over a 4-port voyage. GPU-based experiments use an NVIDIA RTX A6000, and CPU-based runs use an AMD EPYC 9454 48-core processor. Implementation details and additional experiments are provided in Appendix B.5.

7.6.2 Policy Performance

Table 7.1 compares the performance of different solution approaches. Our projected attention models (AM-P) outperform SMIP-NA, achieving around 30-35% higher objective values and 500× faster computations. The AM with FR misleadingly attains higher profits than SMIP-NA due to infeasibility. If we recover feasibility, then profits are significantly reduced to the level of SMIP-NA. In contrast, each AM-P finds feasible solutions with a higher profit than AM with FR, highlighting FR's limitations. All AM-P policies generalize well, with SAC and PPO showing an average profit reduction of 2% and 3% on unseen uniform instances, similar to SMIP-NA and SMIP-PI. These results suggest that AM-P can leverage imperfect information to anticipate future uncertainty and ensure feasibility by projection layers beyond FR.

Ablation Study. Table 7.1 analyzes the impact of feasibility mechanisms. Removing FR from FR/VP increases profit for SAC and slightly decreases profit for PPO, without diminishing feasibility. When FR is removed from FR/WS/PC, profit increases, and both DRL approaches continue to maintain feasibility. This outcome is surprising, given the absence of a direct mechanism to account for stability. However, from our domain understanding, we know container vessels generally become more stable as utilization increases in all locations. Replacing SAC with PPO, which eliminates action gradient flow, reduces feasibility in FR, though projection layers mitigate this effect. These findings highlight the impact of feasibility mechanisms on the objective value and feasibility.

TABLE 7.1: Experimental results comparing DRL methods (SAC/PPO) with a vanilla (AM) or projected attention model (AM-P) and feasibility mechanisms (F.M.): feasibility regularization (FR), weighted scaling (WS), policy clipping (PC) and violation projection (VP). We compare SMIP-NA (non-anticipation) as a baseline, and SMIP-PI (* assumes perfect information which is unrealistic) as an expected upper bound, both solved with CPLEX (CPL). Average performance metrics on N instances include objective value in profit (Ob.), inference time in seconds (Time), percentage of feasible instances (F.I.), and objective value with feasibility recovery (F.O.). Note that † indicates infeasible objectives. Generalization performance is evaluated on unseen, out-of-distribution instances based on a uniform distribution.

Methods			**Testing ($N = 30$)**				**Generalization ($N = 30$)**			
Alg.	**Model**	**F.M.**	**Ob. ($)**	**Time (s)**	**F.I. (%)**	**F.O. ($)**	**Ob. ($)**	**Time (s)**	**F.I.(%)**	**F.O. ($)**
SAC	AM	FR	1113.03†	12.63	0.00	1065.80	1211.87†	15.35	0.00	1041.42
SAC	AM-P	FR/VP	1318.05	15.20	100.00	-	1288.68	14.18	100.00	-
SAC	AM-P	VP	1447.69	15.03	100.00	-	1411.34	13.33	100.00	-
SAC	AM-P	FR/WS/PC	1373.24	13.34	100.00	-	1340.12	14.42	100.00	-
SAC	AM-P	WS/PC	1494.22	13.12	100.00	-	1482.40	12.91	100.00	-
PPO	AM	FR	1842.46†	11.74	0.00	1063.24	1830.95†	12.60	0.00	1066.69
PPO	AM-P	FR/VP	1355.45	15.60	100.00	-	1321.90	14.91	100.00	-
PPO	AM-P	VP	1318.10	14.51	100.00	-	1282.10	14.64	100.00	-
PPO	AM-P	FR/WS/PC	1369.03	14.29	100.00	-	1330.10	14.17	100.00	-
PPO	AM-P	WS/PC	1471.40	13.75	100.00	-	1455.98	13.21	100.00	-
CPL	SMIP-NA	-	1053.02	8434.58	100.00	-	1023.04	8434.94	100.00	-
CPL	SMIP-PI*	-	1713.73	40.75	100.00	-	1680.51	38.93	100.00	-

7.6.3 Managerial Insights

Figure 7.3 performs a sensitivity analysis to examine the effects of SMIP scenario size and demand uncertainty, reporting with an average and 95% confidence interval (CI).

Value of Information. Figure 7.3a compares performance under non-anticipation, imperfect, and perfect information. The predictive accuracy of SMIP models improves with the number of scenarios, as shown by the stabilizing yet decreasing objective values. With 28 scenarios, perfect information enhances profit by 60-65%, while imperfect information via DRL with VP achieves a 25-40% improvement, emphasizing the role of information availability for decision quality.

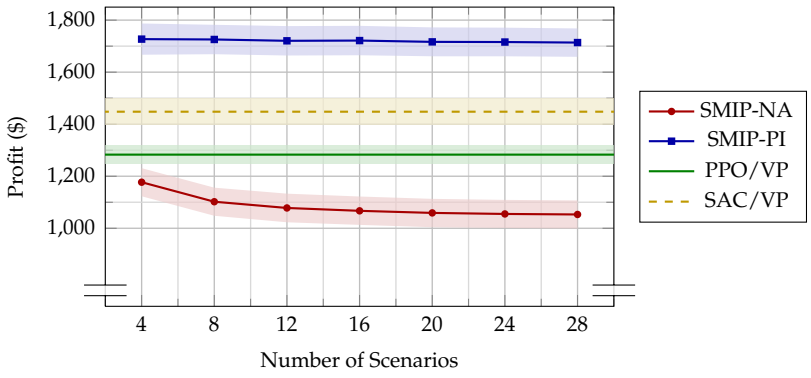
Computational Time. While inference time is critical, training time remains relevant. Figure 7.3b illustrates the total computational time of 30 instances, with training time included for DRL. Computational cost increases significantly with the number of scenarios, with SMIP-NA exhibiting exponential growth and SMIP-PI showing steady growth. Solving the most accurate SMIP-NA takes approximately 2 hours per instance, which is intractable for stowage planners. In contrast, the DRL approach requires offline training of about 11 hours, after which solutions can be constructed in seconds.

Adaptiveness. Figure 7.3c illustrates how policies adapt to variations in the spread of the demand distribution of unseen instances. As variability and uncertainty increase, both SAC and PPO policies achieve higher profits, demonstrating their ability to leverage uncertainty. Despite fluctuations in variability, both policies consistently generate feasible solutions.

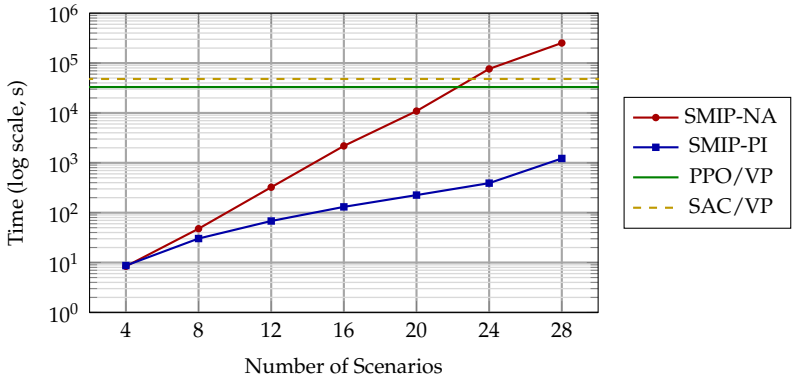
Implications for Practical Deployment. Our findings are based on simulations, limited by the lack of real-world data. To ensure effective deployment, practitioners should focus on collecting and generating representative cargo demand data.

7.7 Conclusion and Future Directions

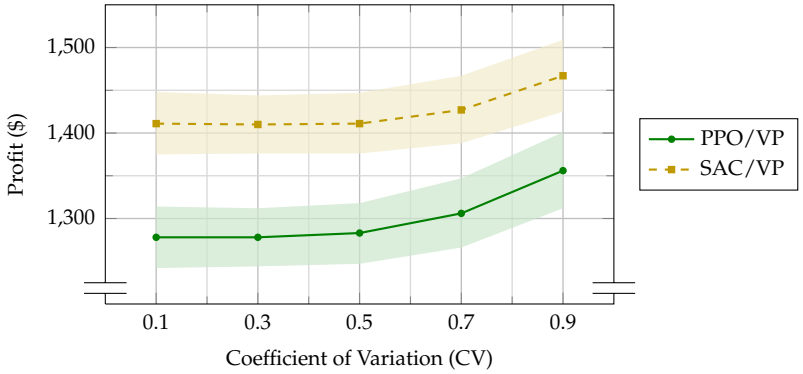
This work introduces a novel MDP formulation for the MPP under demand uncertainty, incorporating realistic inequality constraints. We train an AM policy using actor-critic DRL methods with differentiable feasibility projections to construct MPP solutions. Experimental results demonstrate that our policy efficiently generates adaptive and feasible solutions, significantly outperforming baseline DRL methods and the SMIP-NA. This approach establishes an AI-driven decision-support policy for planning under uncertainty in a critical part of the global supply chain. Future work will extend the MPP formulation and scale to larger vessels and longer voyages, further enhancing the representativeness.



(A) Profit with 95% CI across scenario sizes on 30 instances



(B) Total computational time across scenario sizes of 30 instances



(C) Profit with 95% CI across CV levels on 30 unseen instances

FIGURE 7.3: Sensitivity analysis of scenario size and demand spread

Chapter 8

Deep Reinforcement Learning under Uncertainty at Scale

This chapter will discuss the article: *"AI2STOW: End-to-End Deep Reinforcement Learning to Construct Master Stowage Plans under Demand Uncertainty"* that is currently under review [218]. This study addresses sub-objectives 2 and 3 of the thesis.

In Chapter 4, the need for scalable heuristic frameworks and the incorporation of industrial challenges is highlighted. This chapter builds on the MDP formulation introduced in Chapter 7, extending it with the block dimension and paired block stowage patterns (PBS). The policy incorporates action masking to enforce the non-convex PBS constraint, followed by feasibility projection to satisfy the remaining convex constraints. Experimental evaluations on industrial-scale instances benchmark the proposed architecture against baseline methods from DRL and stochastic optimization, along with various configurations of projection layers.

This chapter mirrors the content of the article [218], with each section corresponding directly to a section in the original work, except a redundant and omitted background section. The remainder of this chapter is structured as follows: Section 8.1 introduces the article, and Section 8.2 outlines related work in stowage planning, stochastic programming and ML for optimization problems. In Section 8.3, the MPP under demand uncertainty is defined as a MIP, while the DRL framework of AI2STOW is defined in Section 8.4. Section 8.5 provides a computational evaluation of AI2STOW with baseline methods, and Section 8.6 concludes the main findings of this study. The supporting material of this chapter can be found in Appendix C.

8.1 Introduction

Product availability and efficient deliveries depend on the smooth operation of complex supply chains to meet the demands of our dynamic and global societies. During the last century, maritime transport has emerged as the cornerstone of global trade and modern consumerism. About 45% of annual transported goods, valued at $8.1 trillion in global trade [215], are transported by container vessels. This substantial economic impact is accompanied by a significant environmental consequence, contributing to more than 400 million tonnes of CO2 emissions yearly [138]. To put this in perspective, a typical passenger vehicle emits about 4.6 tonnes of CO2 per year. However, it is worth noting that container vessels emit significantly less CO2 per cargo tonne-kilometer than other modes of transportation [104].

The maritime transportation business is highly competitive; therefore, liner shipping companies offer global shipments at low prices. Consequently, profit margins are often slim and improving operational efficiency is of paramount importance. As a result, liner shipping companies create stowage plans to allocate containers to vessel capacity [104]. Stowage planning is acknowledged as a complex combinatorial optimization (CO) problem [35, 130, 211], characterized by numerous interdependent objectives and constraints, some of which are NP-hard. This includes, but is not limited to, vessel capacity limitations, seaworthiness regulations, revenue maximization, and operational cost minimization, all of which must be addressed under conditions of cargo demand uncertainty.

Despite its substantial economic and environmental significance, stowage planning remains underexplored, particularly when compared to more established domains such as vehicle routing [104, 221]. The scarcity of publications suggests a relatively immature field, with opportunities for advancement in problem modeling, standardized benchmark datasets, and algorithmic evaluation [221]. Furthermore, many combinatorial aspects essential to solving representative real-world instances are often overlooked in existing research. The same holds for the problem's inherent stochasticity, which remains insufficiently addressed in the current literature [221].

In light of these limitations, prior research has often relied on simplifications to make the problem more tractable. Specifically, it can be observed that most contributions are unable to solve representative versions of the full stowage problem (e.g., [35, 130]). A common strategy to address this complexity is hierarchical decomposition into two sequential subproblems: the master planning problem (MPP) and the slot planning problem (SPP) (e.g., [160, 226]). Despite various attempts, the search for scalable algorithms capable of solving these decomposed yet representative problems remains an open challenge.

In recent years, machine learning (ML), and in particular deep reinforcement learning (DRL), has shown significant potential to complement or enhance traditional CO techniques [27, 147]. A wide range of CO problems has been effectively addressed using ML-based methods, which often excel in scalability, robustness to uncertainty, and adaptability to dynamic conditions [123, 217, 184]. These ML4CO approaches enable the derivation of flexible, data-driven heuristics that are often infeasible to design manually. Nevertheless, the application of ML to stowage planning remains relatively limited. Existing approaches frequently fall short in capturing the full complexity of real-world stowage scenarios [221]. This highlights the importance of a more in-depth investigation into ML-driven approaches specifically designed to address the unique challenges of stowage planning.

This article builds upon our previous work [217], in which we proposed a DRL-based framework for solving the MPP under demand uncertainty. The problem was formulated as a Markov decision process (MDP) that captures key combinatorial aspects, including demand uncertainty, vessel capacity, stability requirements, and the optimization of cargo revenue, hatch overstowage, and excess quay crane moves. To solve this formulation, we introduced an action policy with attention, also called attention model (AM) [123, 222], with a feasibility projection layer, trained using DRL methods to produce adaptive and feasible solutions.

Building on this foundation, we propose AI2STOW, an end-to-end DRL policy for master stowage planning, which offers the following contributions:

- **Extended MDP with Blocks**: We extend the original MDP with blocks to include paired block stowage patterns: an industrially relevant planning strategy often overlooked in existing stowage planning research [221]. The extended implementation is released in the open-source repository[1].
- **Action Mask to Enforce Paired Block Stowage**: We integrate an action-masking mechanism to enforce non-convex paired block stowage constraints in combination with projection layers to minimize convex feasibility violations in the DRL framework.
- **Efficient and Adaptive Solutions**: Experiments show that AI2STOW learns adaptive and feasible policies, outperforming baselines from stochastic programming and DRL in both objective quality and computational efficiency. The evaluation also includes a comparison of different projection layer configurations.
- **Decision Support for Realistic-Sized Instances**: AI2STOW can generalize well to larger problem instances, offering decision support for a realistic-sized vessel and operational planning horizons. These findings underscore the potential of DRL-based approaches in developing scalable algorithms for stowage planning.

8.2 Related Work

This section covers relevant contributions in container stowage optimization, as well as the use of machine learning in the context of CO.

8.2.1 Container Stowage Planning

The complexity of container vessel stowage planning arises from its inherent multi-port nature. Consequently, algorithms are required to balance a myriad of combinatorial aspects at each port. The field can be broadly categorized into single-port work aimed at creating light-weight operational stowage plans (e.g., [12, 7, 55, 130]), and multi-port initiatives dedicated to creating realistic stowage plans (e.g., [35, 18, 226, 160, 177, 169]). As analyzed in [221], a definitive and optimal solution to either problem is yet to be found.

To address the inherent complexity of stowage planning, a recommended strategy involves hierarchical decomposition, which systematically divides the problem into sequential subproblems. One widely recognized version, shown in Figure 8.1, decomposes the problem into the master planning problem (MPP) and the slot planning problem (SPP) (e.g., [226, 160]). During the MPP, groups of containers are allocated to locations on the vessel while aiming to satisfy global objectives and constraints (e.g., [158, 31, 43]). Subsequently, the focus shifts to the SPP, where individual containers are assigned to specific slots within the designated locations that aim to satisfy local objectives and constraints (e.g., [162, 112, 124]). For a comprehensive understanding of hierarchical decomposition, readers are referred to [104]. Furthermore, most stowage planning research assumes deterministic cargo demand, whereas the more realistic stochastic variant has received limited attention, with only a single study addressing it [49]. Recently, heuristic frameworks have gained

[1] https://github.com/OptimalPursuit/navigating_uncertainty_in_mpp

traction as alternatives to hierarchical decomposition, offering more ways to handle the complexity of stowage planning (e.g., [159, 130]).

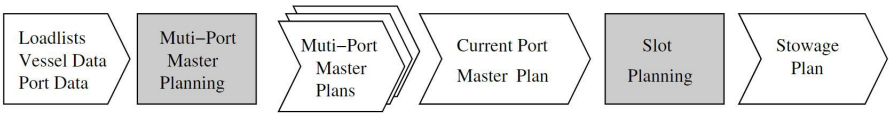


FIGURE 8.1: Hierarchical decomposition of stowage planning [160]

Regardless, several solution methods have been proposed to solve different stowage planning problems, for instance, exact methods (e.g., [177, 237]), greedy heuristics (e.g., [18, 58]), population-based (e.g., [63, 42]) or neighborhood-based metaheuristics (e.g., [7, 159]), matheuristics (e.g., [124, 169]), tree-based methods (e.g., [22]), or hybrid frameworks (e.g., [226, 31]).

To the best of our knowledge, the number of contributions related to RL in stowage planning is limited. Most contributions have focused on the placement of individual containers in single-port problems that omit key combinatorial aspects by methods such as deep Q-learning [194] and Monte Carlo tree search [236]. Given the scale of modern vessels, applying RL at the container level would require placing up to 20,000 containers per voyage, obtaining very long episodes that complicate learning. Therefore, focusing on the MPP, which abstracts the problem to a higher level, may offer a more practical and scalable alternative. To date, the only research applying RL to the MPP has been conducted by us [219, 217]. In these studies, we model the MPP as an MDP and use a DRL approach to derive solutions. While promising, this work requires further improvements in representativeness. To address this, we aim to extend the MPP formulation by incorporating paired block stowage patterns (PBS) and accounting for demand uncertainty while ensuring scalability to industrial vessel sizes and realistic voyage lengths. Despite the growing algorithmic diversity, the search continues for scalable, generalizable methods capable of solving representative stowage planning problems.

8.2.2 Stochastic Programming

A classical approach for optimization under uncertainty is stochastic programming, where uncertainty is modeled explicitly through a discrete set of scenarios that evolve over stages of time [32]. This results in a scenario tree representation, whose size grows exponentially with the number of stages, $\mathcal{O}(b^T)$, where b is the branching factor and T is the number of stages. As a result, multi-stage formulations become computationally challenging beyond a few stages.

Numerous techniques have been developed to address these challenges. Scenario reduction methods [64, 179] aim to approximate the full scenario tree by selecting a representative subset of scenarios that preserves key probabilistic characteristics. Decomposition techniques, including Benders decomposition and variants [175] or Lagrangian relaxation [41], divide the original problem into smaller, more manageable subproblems that are solved iteratively while coordinating shared constraints. The progressive hedging algorithm, introduced by [178] and extended in later work [34], solves scenario subproblems independently while enforcing non-anticipativity through augmented Lagrangian penalties. Stochastic dual dynamic programming[170, 193] approximates value functions in multi-stage problems by iteratively generating Benders cuts through backward passes and simulating decision

paths in forward passes. The sample average approximation method [192, 44] replaces expectations with sample averages over finite scenario sets, enabling tractable optimization and offering convergence guarantees as the sample size increases.

Applications span domains such as supply chain optimization [2, 210], unit commitment in power systems [165], and telecommunications network design [16]. Recent trends include data-driven stochastic programming, which leverages historical data or machine learning for scenario generation and decision policy learning [24, 29]. Another emerging direction is distributionally robust optimization, where solutions are sought under worst-case distributions within an ambiguity set [151, 73].

Despite substantial progress, scaling stochastic programming to high-dimensional, multi-stage, and real-time decision-making environments remains challenging, motivating further research into hybrid models, learning-based approximations, and scalable decomposition frameworks.

8.2.3 Machine Learning for Optimization Problems

In recent years, ML has proven to be an effective tool for solving CO problems, at times outperforming or enriching conventional solution methods [27]. Specifically, DRL has emerged as a promising method to deal with challenging CO problems [147]. The usage of ML can be classified into learning categories, each of which will be discussed in the following paragraphs.

End-to-end learning leverages ML to directly output solutions for input problem instances, thereby circumventing the need for handcrafted heuristics or manually designed search procedures. One particular example can be found in the chip design process, which can also be decomposed hierarchically, with chip floor planning being one sequential subproblem. In this context, a graph convolutional encoder with actor-critic decoders generates an approximate chip floor plan that dictates chip quality, leading to a significant acceleration of the overall process with chips of equal or superior quality compared to human-made designs [148]. Additionally, actor-critic methods have demonstrated excellent performance on various control tasks [188, 148, 156]. Other approaches to solving well-known combinatorial optimization problems include pointer networks that integrate recurrent neural networks (RNNs) with attention mechanisms [224], a graph-based encoder combined with an RNN decoder using multi-head attention and optimized via an actor-critic REINFORCE algorithm [153], and graph attention models trained with REINFORCE using a greedy rollout baseline [123] and multiple rollouts [127].

Within learning solution heuristics, a key challenge is ensuring feasibility in complex and explicit action spaces, particularly when feasible regions are dynamic and state-dependent. Various deep learning methods have addressed constraint learning through techniques such as backpropagation over (in)equality completions [60], differentiable projection layers that map interior points to boundary regions [136], convex programming layers [1], and problem-specific repair mechanisms [46]. In addition, safe reinforcement learning has approached feasibility through constrained MDPs with primal-dual techniques [59], soft barrier functions [225], and safety shields [6].

Even though end-to-end learning can be advantageous, learning to configure algorithmic components can also be effective in guiding the search process. For example,

efficient active search updates the weights of solution-constructing models via DRL for problems such as capacitated vehicle routing and job shop scheduling [86]. Another approach uses an attention model trained with REINFORCE to reconstruct neighborhoods within a large neighborhood search (LNS) framework, outperforming methods with handcrafted heuristics on capacitated and split delivery vehicle routing problems [88].

Another way is to use ML and CO algorithms in parallel. A literature survey explores traditional and learning methods for variable and node selection in branch-and-bound frameworks [139]. For example, deep learning can learn solution strategies and lower bounds by analyzing existing (near-)optimal solutions to instances, which are integrated into a tree search framework to assist branching and pruning decisions [87].

8.3 Problem Formulation

Our problem formulation models the key combinatorial aspects of the MPP, aiming to generate an approximate plan that meets global objectives and constraints under demand uncertainty. These include maximizing cargo revenue, minimizing hatch-overstowage and excess crane move costs, applying valid PBS patterns, and ensuring acceptable values for long crane length, metacentric height, and trim. Like the early stages of stowage planning, where primary objectives take precedence, this problem focuses on global objectives and constraints during the voyage. It intentionally abstracts away individual containers and slots, allowing the interchangeable placement of similar containers in various slots. The solution to the MPP guides the subsequent container placement, considering specific local objectives and constraints in the SPP.

Table 8.1 defines the sets used in this MPP. Let us define a voyage denoted by the ordered set of ports P with N_P being the last port. Given two ports in P, we can derive a subset of the voyage P^{end}_{start}. Let TR be the set of transport pairs of all possible POLs and PODs. Given port p, various subsets of TR can be defined, namely, transports on board before continuing the voyage ($TR^{OB}(p)$), the transports remaining on board (ROB) after discharging but before loading ($TR^{ROB}(p)$), and all transports that are either loaded or discharged ($TR^{M}(p)$). Furthermore, containers are classified into distinct cargo classes in the set K, which include container size (1 or 2 TEU), weight classes (light, medium and heavy) and customer types (spot market or long-term contracts). Consider that vessel locations are defined by bays, decks and paired blocks. Let B be the ordered set of bays with N_B being the number of bays, indicating the longitudinal position extending from fore to aft. For any adjacent pair of bays, we define the set of bay pairs B'. Let D represent the set of decks, differentiating between on-deck d_o and in the hold d_h, thereby determining the vertical positioning. Let BL be the set of paired blocks with N_{BL} being the number of paired blocks in a bay. If $N_{BL} = 1$, there is only a single block. For $N_{BL} > 1$, the first paired blocks in a bay correspond to the wing blocks, while the remaining blocks are center blocks. Note that modern vessels have $N_{BL} \leq 3$.

Table 8.2 defines the MPP parameters. The transport matrix q represents realized demand in containers, rev is the revenue per transport and container type, ct^{ho} refers to the hatch overstowage cost parameter, and ct^{cm} is the excess crane moves cost parameter. Additionally, c denotes vessel location capacity in TEU, while container sizes

TABLE 8.1: Sets of the MPP

Ports	$p \in P = \{1, 2, \ldots, N_P\}$
Port range	$p \in P^{end}_{start} = \{p \in P \mid start \leq p \leq end\}$
Transport pairs	$tr \in TR = \{(i,j) \in P^2 \mid i < j\}$
Onboard transp.	$tr \in TR^{OB}(p) = \{(i,j) \in P^2 \mid i \leq p, j > p\}$
ROB transports	$tr \in TR^{ROB}(p) = \{(i,j) \in P^2 \mid i < p, j > p\}$
Discharge moves	$tr \in TR^{-}(p) = \{(i,p) \in P^2 \mid i < p\}$
Load moves	$tr \in TR^{+}(p) = \{(p,j) \in P^2 \mid j > p\}$
Port moves	$tr \in TR^{M}(p) = TR^{+}(p) \cup TR^{-}(p)$
Cargo classes	$k \in K = \{20ft, 40ft\} \times \{Light, Medium, Heavy\} \times \{Spot, Long\}$
Bays	$b \in B = \{1, 2, \ldots, N_B\}$
Adjacent bays	$b' \in B' = \{(1,2), (2,3), \ldots, (N_B - 1, N_B)\}$
Decks	$d \in D = \{d_o, d_h\}$
Paired blocks	$bl \in BL = \{1, 2, \ldots, N_{BL}\}$

and cargo weights are given by teu and w, respectively. The longitudinal distance ld is measured from fore to aft, and the vertical distance vd is measured from keel upwards, with the center normalized to 1. Stability constraints include the acceptable bounds for the longitudinal and vertical center of gravity ($\underline{lcg}, \overline{lcg}$ and $\underline{vcg}, \overline{vcg}$), as well as the permissible error in excess crane moves (δ^{cm}). We also define big M as a large constant to enforce logical conditions.

TABLE 8.2: Parameters of the MPP

Transport matrix of realized demand	$q \in \mathbb{Z}_{\geq 0}^{\lvert TR \rvert \times \lvert K \rvert}$
Revenue per container	$rev \in \mathbb{R}_{>0}^{\lvert TR \rvert \times \lvert K \rvert}$
Cost per overstowed container	$ct^{ho} \in \mathbb{R}_{>0}$
Cost per excess crane move	$ct^{cm} \in \mathbb{R}_{>0}$
Location capacity in TEU	$c \in \mathbb{Z}_{\geq 0}^{\lvert B \rvert \times \lvert D \rvert \times \lvert BL \rvert}$
Container size in TEU	$teu \in \mathbb{Z}_{\geq 0}^{\lvert K \rvert}$
Cargo weight class in tonnes	$w \in \mathbb{R}_{\geq 0}^{\lvert K \rvert}$
Longitudinal distance of bays	$ld_b = \frac{2b-1}{\lvert B \rvert}\ \forall b \in B$
Vertical distance of deck	$vd_d = \frac{2d-1}{\lvert D \rvert}\ \forall d \in D$
Acceptable LCG bounds	$\underline{lcg}, \overline{lcg} \in \mathbb{R}_{\geq 0}$
Acceptable VCG bounds	$\underline{vcg}, \overline{vcg} \in \mathbb{R}_{\geq 0}$
Acceptable error excess crane moves	$\overline{\delta^{cm}} \in \mathbb{R}_{\geq 0}$
Big M	M

Let us specify some parameters based on the following definitions. The realized demand values $q_{(i,j),k}$ are sampled from a generator $\mathcal{G}$ parameterized by $\vartheta_{i,j,k}$:

$$q_{(i,j),k} \sim \mathcal{G}(\vartheta_{i,j,k}) \quad \forall (i,j) \in TR,\ k \in K, \tag{8.1}$$

The revenue of containers with type k on transport (i,j) is defined by the length of the transport $(j-i)$ plus some standard revenue SR. Note that long-term contracts have their transport revenue $(j-i)$ reduced by the parameter LR.

$$rev_{(i,j),k} = \begin{cases} (j-i)(1-LR)+SR, & \text{if } Long \in k \\ (j-i)+SR, & \text{if } Spot \in k \end{cases} \quad \forall (i,j) \in TR, k \in K. \tag{8.2}$$

The TEU per container depends on type k as:

$$teu_k = \begin{cases} 1, & \text{if } 20ft \in k \\ 2, & \text{if } 40ft \in k \end{cases} \quad \forall k \in K. \tag{8.3}$$

Similarly, container weight is determined by the weight class:

$$w_k = \begin{cases} 1, & \text{if } Light \in k \\ 2, & \text{if } Medium \in k \\ 3, & \text{if } Heavy \in k \end{cases} \quad \forall k \in K. \tag{8.4}$$

To simplify the problem, we make the following assumptions:

- Each voyage starts with an empty vessel, which is box-shaped with equal capacity in each bay.
- The voyages include ports with balanced loading and discharging operations rather than having ports that specialize in either operation.
- Demand at the current port is fully observable and deterministic, whereas demand at subsequent ports is uncertain.
- The uncertainty is exogenous, arising from external factors beyond the system's control and independent of model decisions.
- Special container types (e.g., reefers and IMDG) are not considered. We assume that each location has sufficient capacity for reefers, while other specials with local constraints can be considered in an SPP phase.
- Loading and discharge times are equal for all ports and cargo classes.
- Revenue of cargo uptake $rev_{tr,k}$ is larger than costs of overstowage ct^{ho} or excess crane moves ct^{cm}.

Stochastic programming approaches typically represent uncertainty through an explicit scenario tree. This is a directed tree $T_{ST} = (V_{ST}, E_{ST})$, where V_{ST} is the set of nodes, each corresponding to a decision or uncertainty realization at a given stage. $E_{ST} \subseteq V_{ST} \times V_{ST}$ is the set of directed edges representing transitions between nodes over time. The tree T_{ST} consists of:

1. A root node $v_1 \in V_{ST}$, representing the initial state at the first port.
2. Stages $p = 1, 2, \ldots, N_P - 1$, where each node v belongs to a stage $p(v)$. We denote stages by p, as stages are equivalent to ports in a voyage.
3. Branching structure, where each node has S_{ST} child nodes representing possible future realizations.
4. A measure $P_{ST} : V_{ST} \to [0, 1]$ assigning probabilities to nodes, ensuring:

$$\sum_{v' \in child(v)} \mathbb{P}(v') = \mathbb{P}(v), \quad \forall v \in V_{ST}. \tag{8.5}$$

5. Scenario paths $\phi \in \mathcal{Z}$, which are root-to-leaf paths representing possible realizations of uncertainty over time. The number of scenario paths grows exponentially as $|\mathcal{Z}| = S_{ST}^{N_P - 1}$.

Based on the scenario tree, we define the MPP under demand uncertainty as a multi-stage stochastic MIP. Table 8.3 defines all decision variables under scenario path $\phi \in \mathcal{Z}$ provided by the scenario tree. Vessel utilization $\tilde{u}$ assigns cargo of type k and transport $tr = (i, j)$, with *POL* i and *POD* j, to locations defined by bay b, deck d and block bl. Hatch overstowage $\tilde{ho}$ represents the amount of containers that need to be restowed in bay b and block bl at port p, while excess crane moves $\tilde{cm}$ defines the amount of additional crane moves outside moves allocated to the vessel by a terminal at port p. Hatch movement $\tilde{hm}$ indicates the need to access below-deck locations at bay b, block bl and port p, whereas the destination indicator $\tilde{di}$ signifies that POD j is assigned to bay b and block bl at port p.

TABLE 8.3: Decision variables of the MPP

Vessel utilization	$\tilde{u}_{tr,k}^{b,d,bl,\phi} \in \mathbb{Z}_{\geq 0}\ \forall b \in B, d \in D, bl \in BL, tr \in TR, k \in K, \phi \in \mathcal{Z}$
Hatch overstowage	$\tilde{ho}_{p}^{b,bl,\phi} \in \mathbb{Z}_{\geq 0}\ \forall b \in B, bl \in BL, p \in P, \phi \in \mathcal{Z}$
Excess crane moves	$\tilde{cm}_{p}^{\phi} \in \mathbb{Z}_{\geq 0}\ \forall p \in P, \phi \in \mathcal{Z}$
Hatch movement	$\tilde{hm}_{p}^{b,bl,\phi} \in \{0,1\}\ \forall b \in B, bl \in BL, p \in P, \phi \in \mathcal{Z}$
Destination indicator	$\tilde{di}_{p,j}^{b,bl,\phi} \in \{0,1\}\ \forall b \in B, bl \in BL, p \in P, j \in P_p^{N_P}, \phi \in \mathcal{Z}$

The objective function (8.6) maximizes the expected revenue with parameter $rev_{tr,k} \in \mathbb{R}_{>0}$ and minimizes the expected hatch-overstowage with parameter $ct^{ho} \in \mathbb{R}_{>0}$ and the expected crane moves costs with parameter $ct^{cm} \in \mathbb{R}_{>0}$ over scenario paths $\phi \in \mathcal{Z}$ with probability $\mathbb{P}_\phi$. We assume equal probability for each scenario path.

Constraint (8.7) limits onboard utilization to the cargo demand q, whereas Constraint (8.8) limits each vessel location to the TEU capacity c for each location. Constraint (8.9) links utilization to the destination indicator, while Constraint (8.10) only allows a single POD for each bay and block, thereby enforcing PBS. In Constraint (8.11), we indicate that hatches need to be opened if below deck cargo needs to be loaded or discharged. Based on these movements, Constraint (8.12) models the amount of hatch overstowage in containers. Subsequently, we compute the target of crane moves $\bar{z}$ in Constraint (8.13), after which Constraint (8.14) computes the excess number of crane moves $\tilde{cm}$. Additionally, we model the longitudinal and vertical stability in Constraints (8.16) until (8.19). First, we compute the longitudinal moment, vertical moment and total weight in Constraints (8.16), (8.17) and (8.15), respectively. Second, Constraint (8.18) bounds lcg between $\underline{lcg}$ and $\overline{lcg}$. Third, Constraint (8.19) bounds vcg between $\underline{vcg}$ and $\overline{vcg}$. Finally, we include non-anticipation in Constraint (8.20) to prevent leveraging future demand realizations.

$$\max \sum_{\phi \in \mathcal{Z}} \mathbb{P}_\phi \sum_{p \in P} \sum_{b \in B} \sum_{d \in D} \sum_{bl \in BL} \sum_{k \in K} \sum_{tr \in TR^+(p)} rev_{tr,k} \tilde{u}_{tr,k}^{b,d,bl,\phi} - ct^{ho} \tilde{ho}_{p}^{b,bl,\phi} - ct^{cm} \tilde{cm}_{p}^{\phi} \tag{8.6}$$

$$\text{s.t.} \sum_{b \in B} \sum_{d \in D} \sum_{bl \in BL} \tilde{u}_{tr,k}^{b,d,bl,\phi} \leq q_{tr,k}^{\phi} \quad \forall p \in P, tr \in TR^{OB}(p), k \in K, \phi \in \mathcal{Z} \tag{8.7}$$

$$\sum_{k\in K}\sum_{tr\in TR^{OB}(p)} teu_k \tilde{u}_{tr,k}^{b,d,bl,\phi} \leq c_{b,d,bl}$$
$$\forall p \in P, b \in B, d \in D, bl \in BL, \phi \in \mathcal{Z} \tag{8.8}$$

$$\sum_{k\in K} \tilde{u}_{(p,j),k}^{b,d,bl,\phi} \leq M\tilde{di}_{p,j}^{b,bl,\phi}$$
$$\forall p \in P, j \in P_p^{N_P}, b \in B, d \in D, bl \in BL, \phi \in \mathcal{Z} \tag{8.9}$$

$$\sum_{j\in P_p^{N_P}} \tilde{di}_{p,j}^{b,bl,\phi} \leq 1$$
$$\forall p \in P, b \in B, bl \in BL, \phi \in \mathcal{Z} \tag{8.10}$$

$$\sum_{k\in K}\sum_{tr\in TR^{M}(p)} \tilde{u}_{tr,k}^{b,d_h,bl,\phi} \leq M\tilde{hm}_p^{b,bl,\phi}$$
$$\forall p \in P, b \in B, bl \in BL, \phi \in \mathcal{Z} \tag{8.11}$$

$$\sum_{k\in K}\sum_{tr\in TR^{ROB}(p)} \tilde{u}_{tr,k}^{b,d_o,bl,\phi} - M(1-\tilde{hm}_p^{b,bl,\phi}) \leq \tilde{ho}_p^{b,bl,\phi}$$
$$\forall p \in P, b \in B, bl \in BL, \phi \in \mathcal{Z} \tag{8.12}$$

$$\overline{z}_p^{\phi} = (1+\delta^{cm})\frac{2}{|B|}\sum_{tr\in TR^{M}(p)}\sum_{k\in K} q_{tr,k}^{\phi}$$
$$\forall p \in P, \phi \in \mathcal{Z} \tag{8.13}$$

$$\sum_{b\in b'}\sum_{d\in D}\sum_{bl\in BL}\sum_{k\in K}\sum_{tr\in TR^{M}(p)} \tilde{u}_{tr,k}^{b,d,bl,\phi} - \overline{z}_p^{\phi} \leq c\tilde{m}_p^{\phi}$$
$$\forall p \in P, b' \in B', \phi \in \mathcal{Z} \tag{8.14}$$

$$tw_p^{\phi} = \sum_{k\in K} w_k \sum_{tr\in TR^{OB}(p)}\sum_{bl\in BL}\sum_{d\in D}\sum_{b\in B} \tilde{u}_{tr,k}^{b,d,bl,\phi}$$
$$\forall p \in P, \phi \in \mathcal{Z} \tag{8.15}$$

$$lm_p^{\phi} = \sum_{b\in B} ld_b \sum_{k\in K} w_k \sum_{tr\in TR^{OB}(p)}\sum_{bl\in BL}\sum_{d\in D} \tilde{u}_{tr,k}^{b,d,bl,\phi}$$
$$\forall p \in P, \phi \in \mathcal{Z} \tag{8.16}$$

$$vm_p^{\phi} = \sum_{d\in D} vd_d \sum_{k\in K} w_k \sum_{tr\in TR^{OB}(p)}\sum_{bl\in BL}\sum_{b\in B} \tilde{u}_{tr,k}^{b,d,bl,\phi}$$
$$\forall p \in P, \phi \in \mathcal{Z} \tag{8.17}$$

$$\underline{lcg} \cdot tw_p^{\phi} \leq lm_p^{\phi} \leq \overline{lcg} \cdot tw_p^{\phi}$$
$$\forall p \in P, \phi \in \mathcal{Z} \tag{8.18}$$

$$\underline{vcg} \cdot tw_p^{\phi} \leq vm_p^{\phi} \leq \overline{vcg} \cdot tw_p^{\phi}$$
$$\forall p \in P, \phi \in \mathcal{Z} \tag{8.19}$$

$$\tilde{u}_{tr,k}^{b,d,bl,\phi'} = \tilde{u}_{tr,k}^{b,d,bl,\phi}$$
$$\forall p \in P, tr \in TR^{+}(p), k \in K, b \in B, d \in D, bl \in BL,$$
$$\phi, \phi' \in \mathcal{Z} \mid q_{[p-1]}^{\phi} = q_{[p-1]}^{\phi'} \tag{8.20}$$

8.4 Deep Reinforcement Learning Framework

In the previous section, an explicit MIP model is provided that grows rapidly with problem size due to time-indexed variables and constraints. As an alternative, we propose an MDP formulation of the MPP, modeling it as a sequential decision process. This structure scales with episode length and the sizes of the state and action spaces, avoiding the combinatorial growth inherent in MIPs. In this section, we extend the DRL framework from [217] by refining the MDP, the policy model and feasibility mechanisms. For completeness, we restate the formal and decomposed MDP formulations for the MPP under demand uncertainty, incorporating extensions for blocks and PBS patterns. We then introduce the architecture of AI2STOW, integrating updated self-attention (SA) mechanisms into the dynamic embeddings, before defining the action mask to enforce PBS.

8.4.1 Formal Markov Decision Process

We define an episodic discounted MDP to model the MPP under demand uncertainty. The MDP is represented as $\mathcal{M} = (S, X, \mathcal{T}, \mathcal{R}, P_1^{N_P-1}, \gamma)$, where $T_{epi} = N_1^{N_P-1}$ is a finite horizon equal to the number of load ports. Let us also define the shapes of the utilization, vessel location, and cargo demand as $n_u = |B| \times |D| \times |BL| \times |K| \times |TR|$, $n_c = |B| \times |D| \times |BL|$, and $n_q = |K| \times |TR|$, respectively.

Before formally introducing the aspects of $\mathcal{M}$, we first provide some intuition on this sequential decision-making problem. Each step of the MDP corresponds to a port call, where the state captures the vessel's utilization u_p and the cargo demand at each port q_p. An action x_p represents the placement of specific cargo at vessel locations in the current port. Transitions related to vessel utilization involve adding loaded cargo and removing discharged cargo, while transitions related to demand correspond to the realization of demand upon port arrival. The reward at each step is the cargo revenue minus the hatch overstowage and excess crane move costs.

Formally, the current state $s_p \in S$ is defined as $s_p = (u_p, q_p, \zeta)$, where vessel utilization $u_p \in \mathbb{R}_{\geq 0}^{n_u}$, and realized demand $q_p \in \mathbb{R}_{\geq 0}^{n_q}$. The environment parameters ζ include the expected demand $\mu \in \mathbb{R}_{\geq 0}^{n_q}$ and its standard deviation $\sigma \in \mathbb{R}_{>0}^{n_q}$. The demand is characterized by load ports i, discharge ports j, and cargo types k, where $(i, j, k) \in TR \times K$. Further, the TEU per container is given by $teu \in \{1, 2\}^{n_q}$, cargo weight by $w \in \mathbb{R}_{>0}^{n_q}$, and cargo revenue by $rev \in \mathbb{R}_{>0}^{n_q}$. For consistency in dimensionality, teu and w, previously defined with shape $|K|$ in Section 8.3, are expanded to shape $n_q = |K| \times |TR|$ in the MDP. The initial state is given by $s_0 = (u_0, q_0, \zeta)$, where the vessel starts empty with $u_0 = \mathbf{0}^{n_u}$, and the realized demand q_0 at the initial port. The parameters ζ are randomly initialized at the start of each episode.

The action $x_p \in X$ represents the placement of containers on the vessel, updating the utilization u_p, where $x_p \in \mathbb{R}_{>0}^{n_u}$ and is analogous to $u_{tr,k}^{b,d,bl}$ $\forall tr \in TR_p^+, k \in K, b \in B, d \in D, bl \in BL$. Actions x_p are subject to a feasible region, defined by polyhedron $PH(s_p) = \{x_p \in \mathbb{R}_{>0}^{n_u} : A(s_p)x_p \leq b(s_p)\}$. Here, $A(s_p) \in \mathbb{R}^{m_u \times n_u}$ is the constraint matrix, $b(s_p) \in \mathbb{R}^{m_u}$ is the bound vector, and m_u is the number of constraints. As shown in [160], linearly relaxed MPPs outperform integer MPPs, as the subsequent SPP discretizes the solution. Consequently, we adopt real-valued actions instead of integer-valued actions.

Given a traditional MPP with utilization u_p [220], the constraints of $PH(s_p)$ are defined in terms of actions x_p and pre-loading utilization $u'_p = u^{b,d,bl}_{tr,k}\ \forall tr \in TR^{ROB}_p, k \in K, b \in B, d \in D, bl \in BL$. We also introduce the Hadamard product $\odot$ for element-wise multiplication.

$$x_p^\top \mathbf{1}_{n_c} \leq q_p \tag{8.21}$$
$$x_p teu \leq c - u'_p teu \tag{8.22}$$
$$\text{-}\mathbf{1}^\top\big((lm - \underline{lcg}w)\odot x_p\big) \leq \text{-}\mathbf{1}^\top\big((\underline{lcg}w - lm)\odot u'_p\big) \tag{8.23}$$
$$\mathbf{1}^\top\big((lm - \overline{lcg}w)\odot x_p\big) \leq \mathbf{1}^\top\big((\overline{lcg}w - lm)\odot u'_p\big) \tag{8.24}$$
$$\text{-}\mathbf{1}^\top\big((vm - \underline{vcg}w)\odot x_p\big) \leq \text{-}\mathbf{1}^\top\big((\underline{vcg}w - vm)\odot u'_p\big) \tag{8.25}$$
$$\mathbf{1}^\top\big((vm - \overline{vcg}w)\odot x_p\big) \leq \mathbf{1}^\top\big((\overline{vcg}w - vm)\odot u'_p\big) \tag{8.26}$$
$$x_p \leq M(di(u'_p) + 1 - eu(u'_p)) \tag{8.27}$$

Constraint (8.21) restricts the load x_p to the available demand q_p, while Constraint (8.22) ensures that x_p does not exceed the residual TEU capacity. Constraints (8.23)–(8.24) impose limits on the longitudinal center of gravity (LCG), whereas Constraints (8.25)–(8.26) enforce similar bounds on the vertical center of gravity (VCG). These stability constraints are derived from Constraints (8.15)–(8.19), as presented in [217]. Finally, Constraint (8.27) enforces PBS patterns on the action variable x_p, where $di(u'_p) \in \{0,1\}^{n_u}$ indicates of available locations based on used PODs in u'_p, and $eu(u'_p) \in \{0,1\}^{n_u}$ denotes currently empty locations based on u'_p.

The stochastic transition function $\mathcal{T}(s_{p+1}|s_p, x_p) \in \Delta(S)$ updates the current state. During episodes, the transition comprises multiple components:

- Upon port arrival, port demand q_p is revealed.
- Subsequently, onboard cargo is discharged $u_{p+1} = u_p \odot (1 - \mathbf{e}^-_p)$, where $\mathbf{e}^-_p \in \{0,1\}^{n_q}$ is a binary mask indicating the cargo type and transport to nullify in u_p.
- Finally, cargo is loaded onboard $u_{p+1} = u_p + x_p$. Action x_p is based on the current utilization u_p and revealed demand q_p of port p. Future port demand stays unknown.

The deterministic reward function is defined in Equation (8.28) and computes profit as the difference between revenue and costs. Revenue is given by $\min(q, x^\top \mathbf{1}_{n_c})^\top rev$, where $rev \in \mathbb{R}^{n_q}_{>0}$ is the per-unit revenue, $x \in \mathbb{R}^{n_q \times n_c}_{\geq 0}$ the cargo allocation, and $q \in \mathbb{Z}^{n_q}_{>0}$ the demand vector. The pointwise minimum ensures revenue is only collected up to demand. Costs are determined using state-dependent auxiliary variables: hatch overstows $ho(s_p, p) \in \mathbb{R}^{|B|}_{>0}$ and excess crane moves $cm(s_p, p) \in \mathbb{R}^{|B|-1}_{>0}$. These costs are weighted by $ct^{ho} \in \mathbb{R}_{>0}$ and $ct^{cm} \in \mathbb{R}_{>0}$, respectively.

$$\begin{aligned}\mathcal{R}(s, x, p) = rev \min\left(x^\top \mathbf{1}_{n_c}, q\right) \\ - ct^{ho}\mathbf{1}^\top ho(s, p) + ct^{cm}\mathbf{1}^\top cm(s, p).\end{aligned} \tag{8.28}$$

8.4.2 Decomposed Markov Decision Process

The formal MDP defines an action space of size $|X| \propto |B| \cdot |D| \cdot |BL| \cdot |K| \cdot |P|$. An action simultaneously places all container types and transports on the vessel, which obtains an action space with high dimensionality that may impede learning efficiency [108]. An alternative is to decompose the formal MDP as a sequence of granular decisions indexed by (i, j, k), where $(i, j) \in TR$ represents transport routes and $k \in K$ denotes cargo types. Consequently, actions sequentially handle each transport (p, j) and cargo type k at port p, followed by sailing to the next port. This decomposition reduces the action space to $|X| = |B| \cdot |D| \cdot |BL|$ while distributing decisions over an extended time horizon $t \in H = \{0, 1, \ldots, T_{seq}\}$ with $T_{seq} = |K| \cdot |TR|$. Due to its reduced size, we prefer the decomposed MDP.

The state $s_t = (u_t, q_t, \zeta)$ depends on time step t, including vessel utilization $u_t \in \mathbb{R}^{n_u}_{\geq 0}$ and realized demand $q_t \in \mathbb{R}^{n_q}_{\geq 0}$. The environment parameter ζ remains unchanged. Given the time t, however, we can extract relevant parameters from ζ, such as $pol_t, pol_t, k_t, rev^{(pol_t, pod_t, k_t)}$.

At each step t, the action $x_t \in \mathbb{R}^{n_c}_{\geq 0}$ assigns containers to the utilization u_t. Each action is constrained by the feasible region $PH(s_t)$, defined as $PH(s_t) = \{x_t \in \mathbb{R}^{n_c}_{>0} : A(s_t)x_t \leq b(s_t)\}$. Here, $A(s_t) \in \mathbb{R}^{m_c \times n_c}$ is the constraint matrix, $b(s_t) \in \mathbb{R}^{m_c}$ is the bound vector, and m_c represents the number of constraints. Note that Constraints (8.21)-(8.27) are reformulated to align with $PH(s_t)$.

The stochastic transition consists of loading at each time step t, where x_t is added to u_t, while discharging and demand realization occur only upon arrival at a new port, indicated by $t \in T_{new\ port}$:

$$T_{new\ port} = \Big\{ t \in H \mid \exists p \in P_1^{N_P - 1}, t = |K| \Big((p-1)(N_P - 1) - \frac{p(p-1)}{2} \Big) \Big\}. \tag{8.29}$$

The revenue at step t is computed as $rev(pol_t, pod_t, k_t)\mathbf{1}^\top x_t$. However, costs depend on knowing all loading operations at port p, which is aggregated in utilization u_t at the last step of the port $t \in T_{leave\ port}$:

$$T_{leave\ port} = \Big\{ t \in H \mid \exists p \in P_1^{N_P - 1}, t = |K| \Big(p(N_P - 1) - \frac{p(p-1)}{2} \Big) - 1 \Big\}. \tag{8.30}$$

As a result, the cost signal is sparse, being evaluated only once per port p rather than at each step.

8.4.3 Proposed Architecture

The AI2STOW architecture comprises three key components: an encoder-decoder model parameterized by θ, a feasibility projection layer and the DRL implementation. As illustrated in Figure 8.2, the encoder-decoder model employs a look-ahead policy $\pi_\theta(x|s_t)$ that considers an estimate of future states, which is conditioned on the state s_t and parameterized by a mean $\mu_\theta(s_t)$ and standard deviation $\sigma_\theta(s_t)$. By training within the decomposed MDP and applying policy projection, the framework iteratively generates actions x_t to construct feasible solutions.

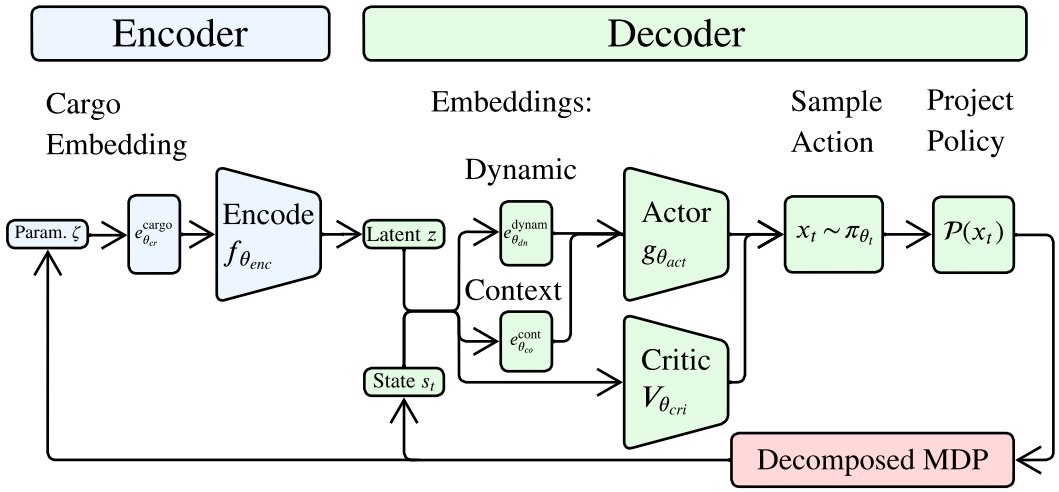


FIGURE 8.2: Deep reinforcement learning architecture with feasibility projection for actor-critic methods [217]

Encoder-Decoder Model

Figure 8.3 presents the encoder-decoder model, originally based on the work of [123], with architectural adjustments highlighted. Note that $+, \times, \|$ denote summation, multiplication and concatenation operations in Figure 8.3.

The core idea behind encoder-decoder models is to convert the input into a compact, low-dimensional representation that allows subsequent models to efficiently extract features and incorporate new information. For instance, the encoder uses embedding $e^{cargo} : 8 \times n_q \to E$, mapping input ζ to a representation of shape E. The encoder $f : E \to Z$ maps the embedding output to latent variable $z \in Z$. The context embeddings takes z and utilization u_t as input by $e^{cont} : Z \times n_u \to E$, whereas the dynamic embedding takes z and q_t as input by $e^{dynam} : Z \times n_q \to E$. In turn, the actor decoder uses both context and dynamic representations to map $g : 2 \times E \to X$. The embeddings transform different parts of the input data into representations, enabling the model to process specific groups of features and learn meaningful patterns from them. This will be discussed in more detail in the text below.

The cargo embedding $e^{cargo}(\zeta)$ parameterized by θ_{cr} maps cargo-related episode information in ζ into a feature representation for the encoder. To incorporate positional information, we apply sinusoidal positional encoding [222], which augments the embeddings with continuous signals that enable the model to distinguish element order and capture position-dependent structure.

An attention encoder $f(e^{cargo}(\zeta))$ parameterized by θ_{enc} maps $e^{cargo}(\zeta)$ to latent variable z using multi-head attention (MHA) to identify relevant features dynamically [222]. In MHA, query (Q), key (K), and value (V) are learned projections of the input that enable the model to selectively weigh and aggregate input elements based on their contextual relevance, where attention weights are computed by comparing queries with keys to determine how much focus to place on each value. Then, we use a feed-forward network (FFN) with ReLU activation to introduce non-linearity, layer normalization to stabilize training, residual connections to sum the MHA output with its input to facilitate gradient flow and preserve input information, and dropout to prevent overfitting [76].

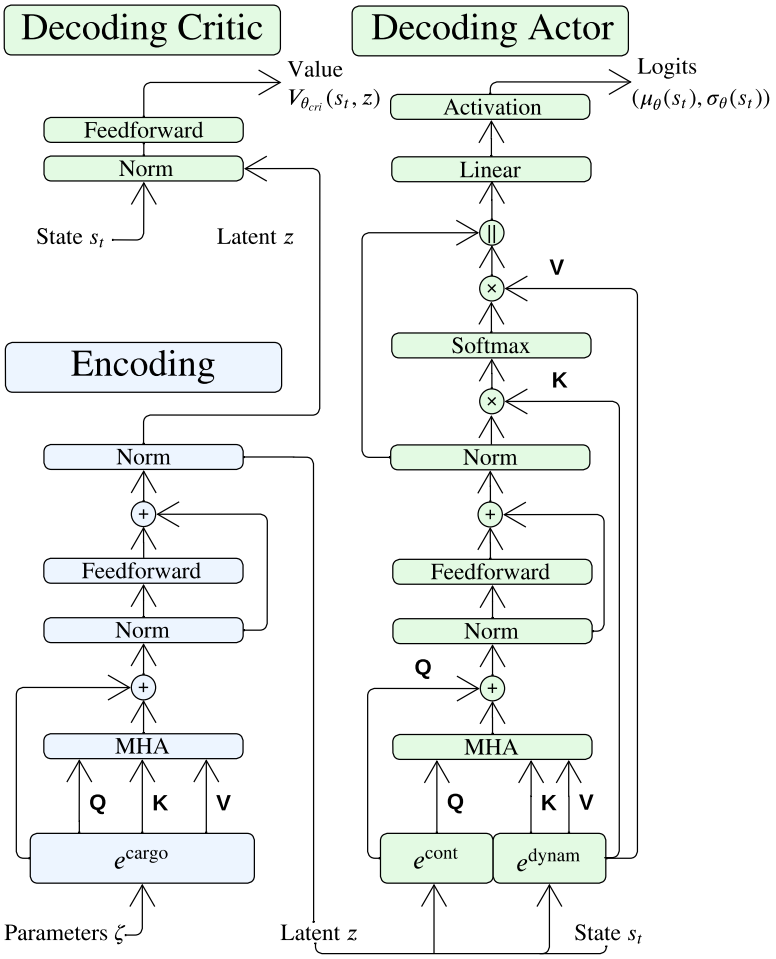


FIGURE 8.3: Layers of the encoder and the actor-critic decoder [217]

The context embedding $e^{cont}(u_t, z)$, parameterized by θ_{co}, focuses on time t, extracting time-specific features from the utilization u_t and latent variable z. These features provide the MHA query with a representation of the vessel's current condition, enabling the policy to make decisions based on the present.

The dynamic embedding $e^{dynam}(q_t, z)$, parameterized by θ_{dn}, leverages SA to capture temporal dependencies in demand q_t over horizon H. By dynamically weighing past and present observations, SA identifies trends and long-term patterns, enhancing decision-making [222]. This embedding combines real demand and latent factors to produce keys and values for an MHA layer. By emphasizing relevant historical patterns while filtering noise, it enables the model to anticipate future conditions and optimize long-term strategies.

As proposed in [123], our actor decoder $g(e^{cont}(u_t, z), e^{dynam}(q_t, z))$ is also an attention model, parameterized by θ_{act}. However, our actor takes input from the context and dynamic embedding. An MHA layer with a forward-looking mask bases decisions on steps $t, t+1, \ldots, T_{seq}$ to anticipate future events. Subsequently, an FFN extracts features, after which a pointer mechanism performs a soft selection of relevant steps using key-value inputs [224]. A softplus activation outputs positive logits $(\mu_\theta(s_t), \sigma_\theta(s_t))$.

Our critic model $V_{\theta_{cri}}(s_t, z)$, parameterized by θ_{cri}, estimates the value of state s_t and latent variable z through an FFN outputting $V_{\theta_{cri}}(s_t, z) \in \mathbb{R}$, providing a low-variance baseline to guide and stabilize policy updates.

The actor logits parameterize a stochastic policy $\pi_\theta(x \mid s_t) = \mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t))$, which enables sampling actions $x_t \sim \pi_\theta(x \mid s_t)$ to construct solutions, while promoting exploration and supporting gradient-based optimization.

Feasibility Mechanisms

A variety of feasibility mechanisms can be used to ensure policy π_θ adheres to feasible region $PH(s_t)$. A straightforward approach is to include feasibility regularization (FR) in the loss function, as shown in Equations (8.31) and (8.32). In previous work, we argued that FR faces significant challenges when applied to dynamic, state-dependent feasible regions [217]. The primary difficulty lies in determining the Lagrangian multiplier λ_f, as balancing multiple constraints with varying scales requires expensive hyperparameter optimization. Despite these limitations, FR remains a useful baseline for comparison.

$$\mathcal{L}(\theta) = -\mathcal{L}_{actor}(\theta) + \lambda_f \mathcal{L}_{feas}(\theta) \tag{8.31}$$

$$\mathcal{L}_{feas}(\theta) = \mathbb{E}_t \left[(A(s_t)x_\theta(s_t) - b(s_t))_{>0} \right] \tag{8.32}$$

Projection layers $\mathcal{P} : X \to X$ enforce feasibility by mapping policy samples onto feasible sets by non-linear transformations, which distort the original action density and require Jacobian-based log-probability corrections [33]. However, such corrections can be undefined, leading to biased gradients, particularly when using closed-form (convex) solvers like [1]. Moreover, these solvers are often too slow for practical use during training. We instead favor efficient, differentiable projection layers with well-defined Jacobians, such as the violation projection (VP) layer in [220]. Regardless of the limitations, solvers remain valuable at inference, where differentiability and speed are less critical, providing a reliable post-processing step to enforce feasibility.

Furthermore, $PH(s_t)$ transforms from a convex to a non-convex polyhedron if PBS is included, as the introduction of binary values creates a discontinuity in the polyhedron. To preserve convexity, we propose omitting Constraint (8.27) from $PH(s_t)$ to allow VP to handle the remaining convex constraints, whereas the non-convex constraint can be handled by action masking. Masking is a differentiable operation that preserves the linear structure, allowing constraint enforcement without output distortion [203]. We define $xm(s_t) \in \{0,1\}^{n_q}$ as a state-dependent action mask. However, since the policy essentially says not to execute an action for some elements in x_t, we want the probability of this element occurring to be 0. Hence, Equations (8.33) and (8.34) define the action masking and handling of log probabilities.

$$x'_t = xm(s_t) \odot x_t \tag{8.33}$$

$$\log \pi(a|s) = xm(s_t) \odot \log \pi(a|s) + (1 - xm(s_t)) \cdot (-\infty) \tag{8.34}$$

Subsequently, we define a state-dependent action mask $xm(s_t)$ to enforce PBS patterns. Algorithm 5 defines the PBS action mask, which ensures the grouping of cargo

with POD j into blocks while maximizing the available space. Recall that blocks include both above-deck and below-deck locations. The algorithm works as follows:

1. The algorithm starts with computing POD demand, residual capacity and POD locations indicator based on the state s_t. Then, it identifies locations that are either empty or already used by POD j.

2. The algorithm calculates the amount of demand required to be placed in empty locations, given the residual capacity.

3. To maintain stability, a random score is assigned to half of the bay-block pairs and reflected symmetrically to the other half. Only empty bay-block pairs are considered.

4. The scored bay-block pairs are sorted in descending order, and a top-k subset is selected such that the cumulative capacity of those empty locations is sufficient to meet the remaining demand.

5. Finally, the mask is constructed by marking the selected top-k empty locations as valid, while preserving locations already used by POD j.

Algorithm 5 Action Mask for PBS

Require: POD $j \in P$, realized demand $q \in \mathbb{Z}_{\geq 0}^{n_q}$, TEU capacity $c \in \mathbb{Z}_{>0}^{n_c}$, utilization $u \in \mathbb{R}_{\geq 0}^{n_c \times n_q}$

Ensure: Valid location mask $xm \in \{0,1\}^{n_c}$

Compute Utilization Status:
POD demand: $q_{pod} = \sum_{i \in P} \sum_{k \in K} q^{(i,j,k)}$
Residual capacity: $c_{res} = c - u \cdot teu$
POD indicator: $il_{b,d,bl,j} = \mathbb{I}\left(\sum_{i \in P} \sum_{k \in K} u^{(b,d,bl,i,j,k)} > 0\right) \forall b \in B,\ d \in D,\ bl \in BL,\ j \in P$
Empty locations: $el = \{(b,d,bl) \mid il_{b,d,bl,j} = 0, \forall j \in P\}$
Used locations with POD j: $ul_j = \{(b,d,bl) \mid il_{b,d,bl,j} = 1\}$

Compute Remaining Demand to Fulfill:
$rq \leftarrow \max\left(0,\ q_{pod} - \sum_{(b,d,bl) \in ul_j} c_{res}^{b,d,bl}\right)$
$fq \leftarrow \min\left(\mathbf{1}^\top c_{res},\ rq\right)$

Symmetric Random Scoring of Empty Bay Blocks:
Sample random scores: $sc_{b,bl} \sim \mathcal{U}(0,1)$
Reflect symmetrically: $\tilde{sc}_{b,bl} = sc_{\min(b,\ |B|-b+1),\ bl}$
Bay-block empty mask: $ms_{b,bl} = \bigwedge_{d \in D} \mathbb{I}\left[(b,d,bl) \in el\right]$
Apply mask: $\tilde{sc} \leftarrow \tilde{sc} \odot ms$
Sort indices: $\Pi \leftarrow \text{argsort}(\text{vec}(\tilde{sc}),\ \text{descending})$

Select Top-k Empty Locations Meeting Capacity Requirement:
$c_\Pi \leftarrow \text{gather}(c \text{ by } \Pi)$
$\hat{c} \leftarrow \text{cumsum}(c_\Pi)$
$k \leftarrow \min\{i \mid \hat{c}_i \geq fq\}$
$\tilde{xm}[i] \leftarrow \mathbb{I}[i \in \{\Pi_1, \ldots, \Pi_k\}]$

Merge Mask with Used Locations for POD:
$xm \leftarrow \tilde{xm} \wedge el \vee ul_j$
return xm

Deep Reinforcement Learning Implementation

Several DRL algorithms can be employed to train a policy in conjunction with a feasibility projection layer. To this end, we provide a general pseudocode outlining the core steps of the training process in Algorithm 6. This pseudocode abstracts the common structure of DRL training loops. The training involves iterative interactions between the agent and the environment. Within the interactions, the policy produces an action based on the current state, which is first element-wise multiplied with state-dependent mask $xm(s_t)$, and then passed through a projection layer for convex constraints. The resulting projected actions are used to collect transitions, which are stored in an experience buffer. This buffer is subsequently used to update value estimates, compute the loss function that guides policy improvement, and update the model parameters θ. The pseudocode is algorithm-agnostic and can be instantiated as either on-policy (e.g., PPO) or off-policy (e.g., SAC), depending on how the experience buffer $\mathcal{D}$ is populated and sampled.

Algorithm 6 General DRL Training with Feasibility Projection

Require: MDP $\mathcal{M}$, initial policy π_θ, projection layer $\mathcal{P}(\cdot)$, mask layer $xm(\cdot)$, buffer $\mathcal{D}$
Ensure: Trained policy π_θ
for each training iteration **do**
  Interact with environment $\mathcal{M}$ using masked and projected policy $\mathcal{P}(xm(s_t) \odot \pi_\theta(s_t))$
  Store collected transitions in experience buffer $\mathcal{D}$
  Update performance metrics (e.g., $G_t,\ V(s_t),\ Q(s_t, x_t)$) from buffer $\mathcal{D}$
  Compute loss $\mathcal{L}(\theta)$ based on buffer $\mathcal{D}$
  Update parameters: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
end for
return π_θ

We also present a general pseudocode for inference in Algorithm 7. During inference, the trained policy generates actions based on the current state. A raw action is sampled from the learned distribution and then passed through the projection layer to enforce feasibility. The projected action is used to compute the reward and transition the environment to the next state. This process is repeated for a fixed time horizon, and the trajectory of visited states and raw actions is recorded for analysis.

Algorithm 7 General DRL Inference with Feasibility Projection

Require: MDP $\mathcal{M}$, trained stochastic policy π_θ, projection layer $\mathcal{P}(\cdot)$, mask layer $xm(\cdot)$
Ensure: Trajectory $\{(s_t, \tilde{x}_t)\}_{t=0}^{T_{seq}}$, episodic reward R_{seq}
Sample initial state s_0 from environment $\mathcal{M}$
Initialize episodic reward $R_{seq} \leftarrow 0$
for $t = 0$ to T_{seq} **do**
  Sample raw action $\tilde{x}_t \sim \pi_\theta(s_t)$
  Apply masking: $x'_t \leftarrow xm(s_t) \odot \tilde{x}_t$
  Apply convex projection: $x_t \leftarrow \mathcal{P}(x'_t)$
  Observe reward $r_t = \mathcal{R}(s_t, x_t)$
  Observe next state $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, x_t)$
  Accumulate reward: $R_{seq} \leftarrow R_{seq} + r_t$
  Update state: $s_t \leftarrow s_{t+1}$
end for
return $\{(s_t, \tilde{x}_t)\}_{t=0}^{T_{seq}}$, R_{seq}

8.5 Experimental Results

In this section, we examine the performance of AI2STOW on the MPP under demand uncertainty based on a realistic problem to assess its scalability. We start by discussing the experimental setup, after which we analyze the performance of the AI2STOW policy. Subsequently, an ablation study is performed, after which managerial insights are provided.

8.5.1 Experimental Setup

The largest container vessels range between 14,500 and 24,000 TEU [221]. To adequately reflect such vessels, we assume the use of a vessel of 20,0000 TEU with 20 bays and 3 hatch covers in each bay. Typically, stowage planners use a horizon of 5 ports to mitigate the complexity of planning entire voyages. This provides a focused and manageable scope for optimization. These and additional MPP parameters are provided in Appendix C.1

In Appendix C.2, we define the instance generator used for the training and evaluation of AI2STOW, including a description of the container and TEU demand per port. An upper bound $ub \in \mathbb{R}^{n_q}$ is computed to align the expected demand with the vessel's maximum capacity, such that $\mathbf{1}^\top ub \propto \mathbf{1}^\top c$. During training, ub is perturbed by a small random factor to encourage instance diversity. Demand instances are then sampled from a uniform distribution $\mathcal{U}(0, ub^{(i,j,k)})$, for all $(i,j) \in TR,\ k \in K$. To test generalization, the voyage length N_P is varied during evaluation.

Several feasibility mechanisms are used in these experiments to address the constraints in $PH(s_t)$. The baseline approach is training DRL with FR (DRL-FR), whereas AI2STOW trains with FR, the general VP layer for convex constraints [217], and the action mask to enforce PBS. Accordingly, the training projection layer, $\mathcal{P}_{train}$, can be denoted as PBS/VP or similar variants. During inference, a convex programming (CP) layer can be used [1], possibly combined with PBS and policy clamping on TEU capacity (PC) [217]. The resulting inference-time projection layer, $\mathcal{P}_{test}$, can thus be expressed as PBS/CP/PC or a corresponding variant. Implementation details on projection layers are provided in Appendix C.3.

We compare AI2STOW against two stochastic programming approaches: a stochastic mixed integer program without anticipation (SMIP-NA), as described in Section 8.3, and a stochastic MIP with perfect information (SMIP-PI), which relaxes the non-anticipativity constraint. The SMIP-NA serves as a baseline for AI2STOW, while SMIP-PI represents an expected upper bound. Since AI2STOW operates with continuous actions, we construct the SMIP models using a linear relaxation of the decision variables $\tilde{u}, \tilde{ho}$ and $\tilde{cm}$.

Training occurs offline on simulated instances provided by the instance generator. Inference (or testing) occurs online on seeded simulated instances: AI2STOW models perform multiple inference rollouts to greedily select the best-performing solution. GPU-based experiments use an NVIDIA RTX A6000, and CPU-based runs use an AMD EPYC 9454 48-Core Processor. Additionally, CPU experiments are given a 1-hour runtime limit, after which the best solution is returned. Implementation details on the DRL algorithms and hyperparameters are also provided in Appendix C.3.

8.5.2 Policy Performance

Table 8.4 presents a comparative evaluation of AI2STOW against baseline methods. Across all test instances, each version of AI2STOW consistently outperforms SMIP-NA in both objective value, achieving a 20 to 25 % improvement, and computation time, with reductions of approximately 20 times and 150 times for the CP and VP variants, respectively. With regard to feasibility, only PBS/VP does not fully guarantee feasibility. However, this limitation can be resolved either by fine tuning the parameters in PBS/VP* or by applying PBS/VP/PC to clip capacity violations. While these results demonstrate significant performance gains, the SMIP-PI model suggests that further improvements may be attainable if more accurate or complete information were available. Nonetheless, AI2STOW stands out for its efficiency and ability to make high-quality decisions under uncertainty. It also significantly outperforms DRL-FR, achieving a 70 % higher objective value while maintaining consistent feasibility, whereas DRL-FR failed to produce any feasible solutions. This highlights the practical advantage of projection layers over feasibility regularization.

Among the AI2STOW variants, PBS/CP demonstrates robustness by ensuring feasibility across all instance sizes. However, this comes at the cost of computational efficiency, with a 7 times increase in runtime compared to PBS/VP variants. In contrast, the feasibility performance of PBS/VP and PBS/VP* degrades when generalizing to larger instances. This degradation can be attributed to the hyperparameter sensitivity of the VP layer. Larger instances change the episode length and the magnitude of actions, which requires fine-tuning of hyperparameters. To address this, clipping on TEU capacity in PBS/VP/PC achieves feasible solutions with only a slight reduction in objective value compared to PBS/CP. This shows that VP, unlike CP, requires targeted fine-tuning across voyage lengths to be generally effective, while adding PC offers a simple and effective way to increase robustness.

In Table 8.4, there is a lack of results for SMIP models if $N_P \in \{5, 6\}$. The exponential growth of the scenario tree with increasing $|\mathcal{Z}| \propto N_P$ significantly impacts computational cost and memory usage. While solving SMIP-NA and SMIP-PI for $N_P = 4$ is possible with a computational time of around 30 minutes, longer voyages become impractical on our hardware due to excessive memory demands, as shown in Appendix C.4. Even if we could hold the scenario tree in memory, we would also face intractable computational times for SMIP-NA. This underscores the scalability of DRL models, which can handle uncertainty implicitly. In contrast, all AI2STOW models across all voyage lengths remain well within the practical tractability threshold of 10 minutes for decision support in stowage planning [160].

TABLE 8.4: Experimental results comparing the AI2STOW framework with various inference-time projection layers ($\mathcal{P}_{test}$) against baseline and upper-bound models. The DRL models are trained on voyages with 4 ports ($N_P = 4$), while both DRL and SMIP models are evaluated on $N = 30$ test instances. SMIP-NA (non-anticipative) serves as a baseline, while SMIP-PI (assumes perfect information) represents an expected upper bound; both SMIPs use $S_{ST} = 20$ and are solved using CPLEX. Reported metrics include average objective value in profit (O.), inference time in seconds (T.), and the percentage of feasible instances (F.). Generalization performance is assessed on longer voyages with $N_P \in 5, 6$, beyond the training distribution.

Methods		**Testing ($N_P = 4$)**			**Gen. ($N_P = 5$)**			**Gen. ($N_P = 6$)**		
Model	$\mathcal{P}_{test}(\cdot)$	**O. ($)**	**T. (s)**	**F. (%)**	**O. ($)**	**T. (s)**	**F. (%)**	**O. ($)**	**T. (s)**	**F. (%)**
AI2STOW	PBS/CP	25637.52	76.37	100.00	34840.63	105.19	100.00	43496.30	182.63	100.00
	PBS/VP	25352.06	10.59	86.67	34326.75	23.90	0.00	42515.64	38.79	0.00
	PBS/VP*	25404.68	11.54	100.00	34326.75	23.51	0.00	42515.64	38.34	0.00
	PBS/VP/PC	25346.14	10.33	100.00	34052.90	22.83	100.00	41979.11	38.09	100.00
DRL-FR	-	14959.52	3.19	0.00	19417.46	5.36	0.00	23471.99	7.96	0.00
SMIP-NA	-	20647.22	1576.77	100.00	-	-	-	-	-	-
SMIP-PI*	-	33787.63	40.91	100.00	-	-	-	-	-	-

8.5.3 Ablation Study

Table 8.5 presents the results of an ablation study, which replaces or removes model components of the model to assess their impact on performance. Replacing SA with FF layers in the dynamic embedding, and substituting the AM policy with an MLP, led to only marginal performance degradation when using PBS/CP as postprocessing step in $\mathcal{P}_{test}$. However, under PBS/VP projection, as used during training, replacing SA with FF layers resulted in more feasibility violations. Interestingly, replacing the AM with an MLP under PBS/VP improved feasibility performance. This suggests that the problem instances are sufficiently regular for simpler architectures to learn (e.g., MLPs). However, AMs are likely to outperform MLPs in less structured and more challenging instances, where long-range dependencies and complex interactions are present [222].

Regarding the projection layers, training with VP works well when using PBS/CP during inference, even improving the objective at the cost of longer runtimes. However, when applying PBS/VP* or VP* during inference, feasibility drops significantly, as PBS patterns are not enforced. This indicates that PBS/CP can effectively resolve feasibility issues found during training. Still, we prefer solutions to remain closer to the feasible region and rely less on post-processing through $\mathcal{P}_{test}(\cdot)$. Notably, training solely on PBS leads to significant reductions in the objective value and depends on post-processing for feasibility.

TABLE 8.5: Ablation study results for variations of AI2STOW components on $N = 30$ instances. The table specifies the DRL algorithm used for training (Alg.), the policy model as an attention model (AM) or a multilayer perceptron (MLP), combined with a dynamic embedding layer using either self-attention (SA) or a feedforward layer (FF), along with the training projection $\mathcal{P}_{train}$ and testing projection $\mathcal{P}_{test}$. Reported metrics include the average objective value in profit (O.), inference time in seconds (T.), and the percentage of feasible instances (F.).

Methods				**Testing ($N_P = 4$)**		
Alg.	**Model**	$\mathcal{P}_{train}(\cdot)$	$\mathcal{P}_{test}(\cdot)$	**O. ($)**	**T. (s)**	**F. (%)**
SAC	SA-AM	PBS/VP	PBS/CP	25637.52	76.37	100.00
SAC	SA-AM	PBS/VP	PBS/VP	25352.06	10.59	86.67
SAC	FF-AM	PBS/VP	PBS/CP	25558.75	79.86	100.00
SAC	FF-AM	PBS/VP	PBS/VP	25224.73	10.51	76.67
SAC	FF-MLP	PBS/VP	PBS/CP	25463.42	71.74	100.00
SAC	FF-MLP	PBS/VP	PBS/VP	25222.44	9.89	96.67
SAC	SA-AM	VP	PBS/CP	26722.24	93.51	100.00
SAC	SA-AM	VP	PBS/VP*	25755.81	16.64	0.00
SAC	SA-AM	VP	VP*	30070.51	6.36	0.00
SAC	SA-AM	PBS	PBS/CP	10501.61	9.76	100.00
SAC	SA-AM	PBS	PBS/VP*	10501.57	4.10	100.00
SAC	SA-AM	PBS	PBS	10367.80	3.19	0.00

8.5.4 Managerial Insights

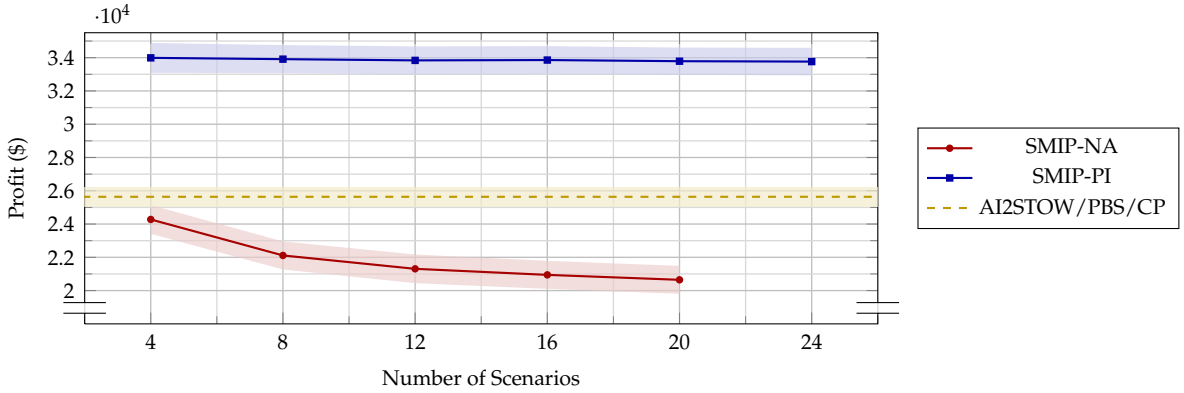
In Figure 8.4a, we assess the value of information by comparing profit under non-anticipation (SMIP-NA), imperfect information (AI2STOW), and perfect information (SMIP-PI). Access to additional information leads to substantial improvements in profit relative to SMIP-NA, approximately 25% for AI2STOW and 60% for SMIP-PI.

While AI2STOW performs significantly better than the non-anticipative baseline, a notable gap remains to the perfect-information upper bound, largely due to the high uncertainty in uniform instances, which are inherently difficult to predict. As the number of scenarios increases, the profit achieved by SMIP-NA gradually decreases and begins to stabilize, indicating diminishing returns from additional information. This trend is less evident in the perfect-information setting. However, convergence is not fully observed within the experimental range: at $S_{ST} = 24$, SMIP-NA returned only empty solutions within the 1-hour time limit, hence, the point is excluded from Figure 8.4a. This suggests that beyond a certain scenario size, the scenario tree becomes computationally intractable, regardless of the tractability limit of practical stowage planning.

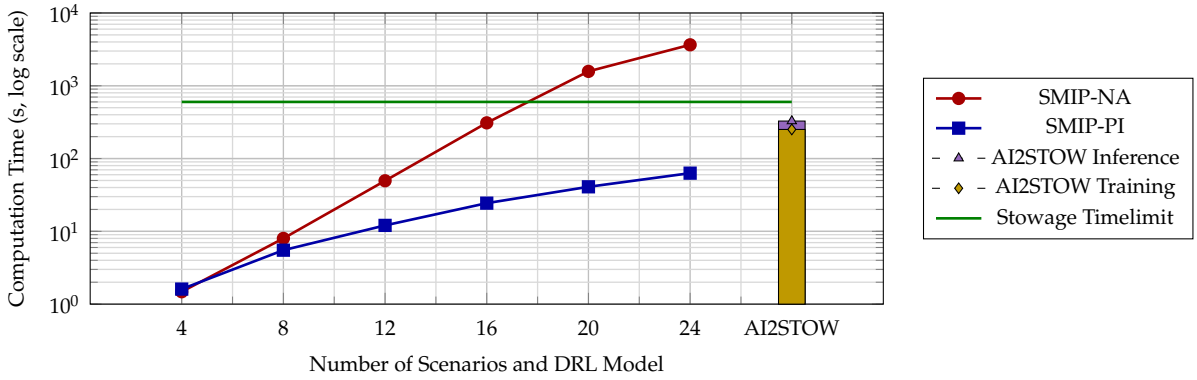
Figure 8.4b illustrates the computational cost of SMIP-NA and SMIP-PI alongside the training and inference time of AI2STOW. Expanding the scenario tree is computationally expensive, with SMIP-NA exhibiting an exponential increase in runtime. The only deviation from this trend occurs at 24 scenarios, where SMIP-NA exceeds the 1-hour time limit. SMIP-PI also shows a steady increase in computational time as scenario size grows. Additionally, the estimated runtime for solving a single instance with AI2STOW is lower than that of SMIP-NA at 20 and 24 scenarios. This suggests that, even with just 30 test instances, the one-time offline training cost of AI2STOW is already justified. As the number of instances increases, this training time is further amortized, improving the cost-effectiveness in larger deployments. If the training budget is sufficient, then the resulting inference time is significantly lower compared to the runtime of SMIP models.

In Figure 8.4c, the performance of AI2STOW with SA-AM, FF-AM, and FF-MLP is evaluated under varying levels of distributional shift. Note that AI2STOW is trained with $UR = 1.1$. The results indicate that all three models perform similarly across different distribution shifts with consistent variability, while SA-AM consistently achieves slightly higher profits in shifting scenarios. This suggests that reducing model complexity does not compromise performance. A closer examination of the distribution shift reveals an asymmetric response. When the distribution is shifted toward lower *UR* values, performance decreases in a manner that scales approximately with the reduced demand, which can be expected as these scenarios fall within the support of the original uniform demand distribution. Conversely, increasing *UR* does not yield improved performance. This indicates that none of the models effectively handle substantial shifts beyond the original distribution, highlighting their limited extrapolation capability under significant demand increases.

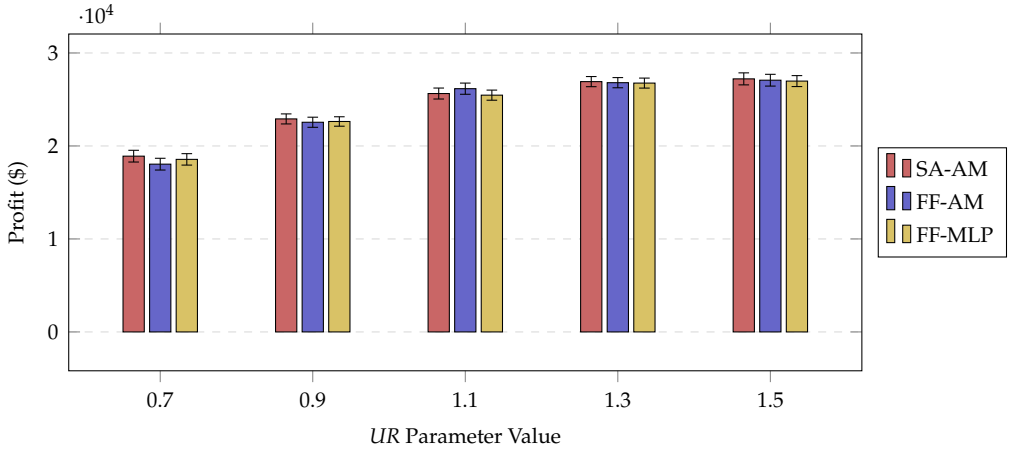
From a practical perspective, we show that including stochastic cargo demand results in a more realistic and complex optimization problem. Particularly challenging to solve using explicit scenario tree formulations, which quickly become intractable. We demonstrate that AI2STOW efficiently generates solutions that implicitly handle uncertainty and consistently identifies master plans with higher expected profit. Even when accounting for training time, our approach is computationally superior to SMIP models. To demonstrate generalization, AI2STOW adapts well to previously unseen and larger voyage lengths. However, under significant distribution shifts, its performance can stagnate. Moreover, its success depends heavily on the quality of training data; when this data is representative of realistic instances, the method performs effectively in practice. This underscores the importance of realistic demand simulators to support the training of DRL-based methods.



(A) Profit with 95% CI across scenario sizes.



(B) Average computational time across scenario sizes for SMIP models and AI2STOW. For AI2STOW, the reported time includes both training and inference averaged over 30 instances. Additional instances would further smooth the training time estimate.



(C) Profit with 95% CI under distributional shift.

FIGURE 8.4: Sensitivity analysis across scenario size and distributional shift

8.6 Conclusion

This article introduces AI2STOW, an end-to-end deep reinforcement learning model designed to solve the MPP under demand uncertainty, specifically tailored for realistic vessel sizes and operational planning horizons. AI2STOW builds on previous work [217] by extending the MDP to include paired block stowage patterns alongside existing global objectives, such as revenue maximization and the minimization of hatch overstowage and excess crane moves. It also respects vessel capacity and ensures both longitudinal and vertical stability.

Experimental results show that AI2STOW generates feasible and adaptive stowage plans for simulated instances on a realistic-sized vessel and operational voyage lengths. AI2STOW also outperforms baseline methods from both deep reinforcement learning and stochastic programming. These findings provide strong evidence that DRL is a promising approach for scalable stowage planning.

Looking ahead, we aim to improve the MDP's representativeness by incorporating local objectives and constraints. We also plan to integrate this framework with the SPP for more efficient end-to-end stowage plan construction. Finally, we encourage exploration into hybrid approaches that combine ML with CO frameworks as alternatives to purely end-to-end learning.

Chapter 9

Conclusion and Future Directions

This chapter summarizes the key findings of this dissertation, reflects on their broader implications, discusses the ethical considerations of the study, and outlines directions for future research.

9.1 Conclusion

This thesis set out to investigate scalable solution methods in ML and CO to efficiently generate solutions to (subproblems of) the CSPP, thereby supporting decision-making in stowage planning. The findings indicate that while advances in CO and ML can significantly enhance the computational efficiency of solution methods, each presents distinct strengths and limitations. Neither CO nor ML provides a universally superior, or *silver bullet*, solution to the challenging CSPP or its subproblems.

With respect to sub-objective 1, the findings indicate that a wide range of CO approaches and only a limited number of ML methods have been applied to non-standardized versions of the CSPP and its subproblems, often relying on non-public problem instances. These observations highlight the need for a more unified approach to the CSPP with a clear problem formulation and publicly available benchmark datasets. To support this, the minimal requirements for a representative CSPP are defined in this thesis, along with a set of benchmark instances. The issue of representativeness is mostly relevant to the CSPP and MPP, while the SPP is close to being representative for industrial problems. Several research gaps are identified for representative operational constraints in the CSPP and MPP, including paired block stowage patterns, valid restows, lashing requirements, and demand uncertainty. In terms of solution methods, there is a clear need for scalable and efficient approaches capable of handling the complexity of these constraints.

To address sub-objective 2, this thesis presents novel formulations of MPP. First, a novel 0-1 IP model is proposed, which searches within the space of valid block stowage patterns. By abstracting away direct container-to-capacity assignments, this formulation significantly reduces the number of decision variables compared to traditional MIP approaches. Second, the thesis proposes MDP formulations of the MPP, modeling it as a sequential decision process. MDPs scale with episode length for a given state and action space, while MIPs scale with the number of indices and constraints on variables, often growing rapidly with the problem size. Third, the integration of paired block stowage patterns is introduced in both the 0-1 IP model and an MDP formulation, representing a novel inclusion of this operational constraint.

Finally, this thesis incorporates demand uncertainty directly into the MPP, which has not yet been addressed within the MPP.

As part of sub-objective 3, the findings indicate that the 0-1 IP model outperforms the traditional MIP formulation in terms of both optimality and runtime while providing a sufficiently accurate representation of the MPP on real-world instances. However, for larger problem instances, the runtime of the 0-1 IP model exceeds the 10-minute tractability threshold. In parallel, DRL methods are used to generate solutions for the MDPs. These methods typically produce fast solutions, often within 2 minutes per instance. Nevertheless, DRL requires extensive training on generated instances to learn policies that yield profitable solutions without a guarantee of optimality. Moreover, the MDP formulations do not inherently ensure feasibility, and feasibility has proven difficult to learn through reward shaping or feasibility regularization alone. To address this challenge, the thesis emphasizes the use of differentiable projection layers, which have shown promising results in improving feasibility while retaining the efficiency of DRL-based approaches.

In response to sub-objective 4, two key theoretical contributions are established. First, this thesis demonstrated that searching within the space of valid block stowage patterns is NP-hard, confirming the inherent combinatorial complexity of the task. Second, our proposed differentiable projection layer based on violation gradient descent is shown to minimize the violation of convex inequality constraints within a continuous optimization framework, allowing integration with gradient-based learning models.

9.2 Discussion

The CSPP is essential to a reliable and efficient global supply chain, which is the backbone of international trade. Given that container shipping has relatively low emissions per cargo tonne-kilometer, effective stowage planning supports both operational performance and the broader goals of sustainable logistics and the global green transition. The CSPP is inherently challenging due to the wide variety of objectives and constraints involved in realistic settings. Capturing its full complexity requires additional efforts to incorporate practical considerations into solving representative problem formulations. Despite its operational relevance and complexity, the CSPP remains underrepresented in academic research. This planning problem requires greater attention, not due to a lack of relevance, but because its significance calls for deeper and more rigorous investigation.

From a decision-support perspective, the CO- and ML-based approaches reduce computational costs, enhance feasibility in ML models, and improve adaptability under uncertainty. They enable rapid what-if analysis, can integrate with automated planning systems, and bridge the gap between theoretical and operational planning. These scalable decision-support systems pave the way for more resilient and efficient container shipping.

A key contribution of this thesis is addressing uncertainty in future port calls, which is often overlooked in prior work, as discussed in Chapter 4. One of the few exceptions is [49], which uses a rolling horizon matheuristic to manage scenario tree complexity. In contrast, this thesis offers a scalable DRL model that implicitly captures uncertainty without relying on explicit scenario trees. Practically, the model

can serve as a warm-start solution within existing decision-support tools, accelerating convergence and improving robustness. It can also function as an exploratory tool, helping planners understand the effects of uncertainty on decision quality and manage operational risk more effectively.

Given the minimal requirements of a representative problem and benchmark data, this thesis establishes a foundation for reproducible evaluation, meaningful comparison, and accelerated development of future stowage optimization methods. Similarly, the open-source MDP environments can benchmark algorithms and facilitate research into sequential decision-making approaches, allowing researchers to flexibly test, compare, and iterate on policies.

This dissertation investigated exact CO methods and ML heuristics as two alternative solution methods to better understand their respective strengths and limitations. The findings indicate that exact CO methods offer guarantees of optimality and feasibility, but they often struggle to scale and require explicit modeling of reality. In contrast, ML heuristics scale efficiently and learn implicit representations of real-world complexity, yet they lack reliable performance guarantees. These insights support the argument in favor of hybrid approaches that combine the strengths of both paradigms while mitigating their shortcomings.

While recent efforts (e.g., [169, 130]), including this work, make progress toward more representative formulations of the CSPP by incorporating more realistic constraints, the studied problems remain simplified and do not yet capture the full complexity of real-world objectives and operational requirements. The formulation proposed by [198] offers a more comprehensive standard for the CSPP but also highlights the difficulty of solving such instances. In general, as problem complexity increases, exact CO methods become computationally intractable, and heuristic frameworks struggle to balance objectives and feasibility. In particular, ML-based heuristics are challenged by explicit, non-convex feasible regions. Bridging this gap remains an open challenge, particularly the integration of combinatorial optimization into end-to-end differentiable frameworks that offer performance guarantees.

Empirical evaluation of the CSPP is hindered by the limited availability of publicly accessible data. Although instance generators exist [18, 53, 58, 177], they frequently lack sufficient realism for widespread applicability. Real-world stowage planning data intrinsically reflects complex factors such as internal organizational policies, global trade dynamics, and external events, resembling a similar complexity found in financial markets. As a result, the simulators integrated into our MDP share these same limitations. Consequently, developing instance generators capable of accurately representing realistic scenarios is particularly challenging but also very useful. The development of such generators requires deep domain insights and extensive publicly available data.

Furthermore, I would like to share some personal reflections on the thesis process. Given the complexity of the CSPP, the steep learning curve required to understand the domain-specific background should be acknowledged. In retrospect, prior familiarity with container shipping could have significantly accelerated the early stages of the research, particularly within the constraints of a limited timeframe. In the project's time constraints, I have committed to specific problem formulations that aligned with the research scope and goals. Nevertheless, I fully acknowledge the possible existence of alternative formulations that may offer different perspectives

on the problem. Moreover, my perspective on both the CSPP and the research process itself has evolved significantly. What began as a mathematical optimization problem gradually revealed itself as a human-in-the-loop decision-making challenge, shaped by operational constraints and safety-critical considerations. At the same time, I came to see AI-based research not merely as analysis and experimenting but as a craft that demands intuition, patience, and iterative refinement. These experiences have enriched my understanding of research and enhanced my capacity to navigate complex, real-world problems.

9.3 Ethical Considerations

Decision-support tools have significant potential to enhance decision-making processes. However, maintaining human oversight is essential to ensure fairness, accountability, and transparency. While these models can substantially augment human judgment, they must never entirely replace it, especially in high-stakes scenarios. To mitigate automation bias effectively, it is crucial to implement safeguards that allow users to question, challenge, or override AI-generated outputs. Additionally, continuous validation and regular auditing of these systems are necessary to maintain trust and uphold fairness.

The use of AI models in decision-making incurs significant costs, both economically and environmentally. Energy consumption is a key factor, as it is directly proportional to the runtime and size of AI models. As such, it is essential to align the model size and runtime budget with the specific task. Assuming unlimited resources is unrealistic and irresponsible as it neglects the importance of resource management and sustainable AI practices.

9.4 Future Directions

Building upon the work presented in this thesis, future research could focus on developing hybrid solution methods that integrate ML into CO frameworks. These hybrid approaches aim to mitigate the scalability challenges faced by many CO methods while achieving feasible solutions with near-optimal objectives. One potential avenue is the development of ML-driven LNS, where an ML policy selects relevant neighborhoods during the search process, as opposed to relying on randomized selection [172]. Another promising direction is to extend the work in this thesis by proposing a three-phase matheuristic framework. The first phase could involve a variant of AI2STOW [218], followed by the second phase, which would incorporate a representative MPP model as described in [221], and the third phase could involve solving the representative SPP model [166]. This approach would allow AI2STOW to warm-start with an MIP model, which in turn generates input variables for the SPP model.

An alternative direction of research could address the challenge posed by the limited availability of benchmark data, which may hinder the ability of ML-based approaches to learn meaningful representations. To overcome this issue, further work could focus on developing a representative instance generator that simulates market dynamics and economic factors. In parallel, the inherent structure of container vessels, characterized by pre-planned voyages and a cellular layout, suggests that graph representation learning could be an effective method to leverage the structure

and inform ML policies. Additionally, container vessels deal with hydrodynamics and stress forces due to weather conditions and vessel motion. Future research could explore the use of physics-informed ML to model and predict these physical phenomena more accurately, offering deeper insights into the behavior of container vessels under varying conditions.

Bibliography

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, and Stephen Boyd. 2019. Differentiable Convex Optimization Layers. In *Advances in Neural Information Processing Systems*. DOI: 10.5555/3454287.3455145.

[2] Shabbir Ahmed. 2006. Convexity and decomposition of mean-risk stochastic programs. *Mathematical Programming*, 106, 3, (May 2006), 433–446. DOI: 10.1007/s10107-005-0638-8.

[3] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1993. *Network flows: Theory, algorithms, and applications*. Prentice Hall. ISBN: 978-1-292-04270-1.

[4] Mai L. Ajspur, Rune M. Jensen, and Kent H. Andersen. 2019. *A decomposed fourier-motzkin elimination framework to derive vessel capacity models*. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11756 LNCS. Pages: 100. Springer International Publishing. ISBN: 978-3-030-31139-1. DOI: 10.1007/978-3-030-31140-7_6.

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (KDD '19). event-place: Anchorage, AK, USA. Association for Computing Machinery, New York, NY, USA, 2623–2631. ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330701.

[6] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. en. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. (Apr. 2018). DOI: 10.1609/aaai.v32i1.11797.

[7] Daniela Ambrosino, Davide Anghinolfi, Massimo Paolucci, and Anna Sciomachen. 2010. An Experimental Comparison of Different Heuristics for the Master Bay Plan Problem. In *Experimental Algorithms*. Vol. 6049. Springer Berlin Heidelberg, 314–325. DOI: 10.1007/978-3-642-13193-6_27.

[8] Daniela Ambrosino, Massimo Paolucci, and Anna Sciomachen. 2015. A MIP Heuristic for Multi Port Stowage Planning. *Transportation Research Procedia*, 10, July, 725–734. Publisher: Elsevier B.V. DOI: 10.1016/j.trpro.2015.09.026.

[9] Daniela Ambrosino, Massimo Paolucci, and Anna Sciomachen. 2015. Computational evaluation of a MIP model for multi-port stowage planning problems. *Soft Computing*. Publisher: Springer Berlin Heidelberg. DOI: 10.1007/s00500-015-1879-y.

[10] Daniela Ambrosino, Massimo Paolucci, and Anna Sciomachen. 2015. Experimental evaluation of mixed integer programming models for the multi-port master bay plan problem. *Flexible Services and Manufacturing Journal*, 27, 2-3, 263–284. Publisher: Springer US. DOI: 10.1007/s10696-013-9185-4.

[11] Daniela Ambrosino, Massimo Paolucci, and Anna Sciomachen. 2018. Shipping Liner Company Stowage Plans: An Optimization Approach. In *Advances in Intelligent Systems and Computing*. Vol. 572. Jacek Żak, Yuval Hadas, and Riccardo Rossi, editors. Springer International Publishing, Cham, 405–420. DOI: 10.1007/978-3-319-57105-8_20.

[12] Daniela Ambrosino, Anna Sciomachen, and Elena Tanfani. 2004. Stowing a containership: the master bay plan problem. *Transportation Research Part A*, 38, 2, (Feb. 2004), 81–99. DOI: 10.1016/j.tra.2003.09.002.

[13] Daniela Ambrosino, Anna Sciomachen, and Elena Tanfani. 2006. A decomposition heuristics for the container ship stowage problem. *Journal of Heuristics*, 12, 3, (May 2006), 211–233. DOI: 10.1007/s10732-006-5905-1.

[14] S. Arora and B. Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press. ISBN: 978-0-521-42426-4.

[15] Anastasios Haralampos Aslidis. 1989. *Combinatorial algorithms for stacking problems*. PhD thesis. Massachusetts Institute of Technology, USA. https://dspace.mit.edu/handle/1721.1/33478.

[16] Alper Atamtürk and Muhong Zhang. 2007. Two-Stage Robust Network Flow and Design Under Demand Uncertainty. *Operations Research*, 55, 4, 662–673. DOI: 10.1287/opre.1070.0428.

[17] Mordecai Avriel, Michal Penn, and Naomi Shpirer. 2000. Container ship stowage problem: complexity and connection to the coloring of circle graphs. *Discrete Applied Mathematics*, 103, 1-3, (July 2000), 271–279. DOI: 10.1016/S0166-218X(99)00245-0.

[18] Mordecai Avriel, Michal Penn, Naomi Shpirer, and Smadar Witteboon. 1998. Stowage planning for container ships to reduce the number of shifts. *Annals of Operations Research*, 76, 1-4, 55–71. DOI: 10.1023/A:1018956823693.

[19] Mordecai; Avriel and Michal Penn. 1993. Exact and approximate solutions of the container ship stowage problem. *Computers & Industrial Engineering*, 25, 271–274. DOI: 10.1016/0360-8352(93)90273-Z.

[20] Jose Roberto Ayala Solares et al. 2020. Deep learning for electronic health records: A comparative review of multiple deep neural architectures. *Journal of Biomedical Informatics*, 101, 103337. DOI: https://doi.org/10.1016/j.jbi.2019.103337.

[21] Win Cho Aye, Malcolm Yoke Hean Low, Huang Shell Ying, Hsu Wen Jing, Liu Fan, and Zeng Min. 2010. Visualization and simulation tool for automated stowage plan generation system. *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010, IMECS 2010*, II, 1013–1019. https://www.iaeng.org/publication/IMECS2010/IMECS2010_pp1013-1019.pdf.

[22] Anibal Azevedo, Ribeiro Cassilda Maria, Galeno José de Sena, Antônio Augusto Chaves, Luis Leduino Salles Neto, and Antônio Carlos Moretti. 2014. Solving the 3D container ship loading planning problem by representation by rules and meta-heuristics. *International Journal of Data Analysis Techniques and Strategies*, 6, 3, 228–260. DOI: 10.1504/IJDATS.2014.063060.

[23] Anibal Tavares Azevedo, Luiz Leduino de Salles Neto, Antônio Augusto Chaves, and Antônio Carlos Moretti. 2018. Solving the 3D stowage planning problem integrated with the quay crane scheduling problem by representation by rules and genetic algorithm. *Applied Soft Computing Journal*, 65, (Apr. 2018), 495–516. Publisher: Elsevier Ltd. DOI: 10.1016/j.asoc.2018.01.006.

[24] Gah-Yi Ban and Cynthia Rudin. 2019. The Big Data Newsvendor: Practical Insights from Machine Learning. *Operations Research*, 67, 1, 90–108. DOI: 10.1287/opre.2018.1757.

[25] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W.P. Savelsbergh, and Pamela H. Vance. 1998. *Branch-and-price: Column generation for solving huge integer programs*. Number 3. Vol. 46. Publication Title: Operations Research. INFORMS.

[26] Richard Bellman. 1957. *Dynamic Programming*. Princeton University Press. ISBN: 978-0-691-07951-6.

[27] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. 2021. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290, 2, (Apr. 2021), 405–421. DOI: 10.1016/j.ejor.2020.07.063.

[28] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*. J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors. Vol. 24. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[29] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. 2018. Data-driven robust optimization. en. *Mathematical Programming*, 167, 2, (Feb. 2018), 235–292. DOI: 10.1007/s10107-017-1125-8.

[30] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. *Evolution Strategies: A Comprehensive Introduction*. Vol. 1. Publication Title: Natural Computing. Springer. DOI: 10.1023/A:1015059928466.

[31] Mevlut Savas Bilican, Ramazan Evren, and Mumtaz Karatas. 2020. A Mathematical Model and Two-Stage Heuristic for the Container Stowage Planning Problem with Stability Parameters. *IEEE Access*, 8, 113392–113413. DOI: 10.1109/ACCESS.2020.3003557.

[32] J.R. Birge and F. Louveaux. 2011. *Introduction to Stochastic Programming*. *Springer Series in Operations Research and Financial Engineering*. Springer New York. ISBN: 978-1-4614-0237-4.

[33] Christopher M. Bishop. 2006. *Pattern recognition and machine learning*. en. *Information science and statistics*. Springer, New York. ISBN: 978-0-387-31073-2.

[34] Natashia Boland, Jeffrey Christiansen, Brian Dandurand, Andrew Eberhard, Jeff Linderoth, James Luedtke, and Fabricio Oliveira. 2018. Combining Progressive Hedging with a Frank–Wolfe Method to Compute Lagrangian Dual Bounds in Stochastic Mixed-Integer Programming. en. *SIAM Journal on Optimization*, 28, 2, (Jan. 2018), 1312–1336. DOI: 10.1137/16M1076290.

[35] R.C. Botter and M.A. Brinati. 1992. Stowage container planning: a model for getting an optimal solution. *Computer Applications in Automation of Shipyard Operation and Ship Design*, VII, C, 217–228. https://trid.trb.org/View/443380.

[36] Stephen P. Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. en. (Version 29 ed.). Cambridge University Press, Cambridge New York Melbourne New Delhi Singapore. ISBN: 978-0-521-83378-3.

[37] Berit Brouer, Fernando Alvarez, Christian Plum, David Pisinger, and Mikkel Sigurd. 2013. A Base Integer Programming Model and Benchmark Suite for Liner-Shipping Network Design. *Transportation Science*, 48, (Jan. 2013). DOI: 10.1287/trsc.2013.0471.

[38] E. Burke, K. Jackson, J. H. Kingston, and R. Weare. 1997. Automated university timetabling: The state of the art. *The Computer Journal*, 40, 9, (Jan. 1997), 565–571. DOI: 10.1093/comjnl/40.9.565.

[39] Edmund K. Burke, Graham Kendall, John Newall, Emma Hart, Peter Ross, and Stefan Schulenburg. 2003. Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of Metaheuristics*. Springer, 457–474. ISBN: 978-1-4419-1663-1.

[40] Miguel Calvo-Fullana, Santiago Paternain, Luiz F. O. Chamon, and Alejandro Ribeiro. 2021. State Augmented Constrained Reinforcement Learning: Overcoming the Limitations of Learning With Rewards. *IEEE Transactions on Automatic Control*, 69, 4275–4290. https://api.semanticscholar.org/CorpusID:232035640.

[41] Claus C. Carøe and Rüdiger Schultz. 1999. Dual decomposition in stochastic integer programming. en. *Operations Research Letters*, 24, 1-2, (Feb. 1999), 37–45. DOI: 10.1016/S0167-6377(98)00050-9.

[42] Yimei Chang, Masoud Hamedi, and Ali Haghani. 2022. Solving integrated problem of stowage planning with crane split by an improved genetic algorithm based on novel encoding mode. *Measurement and Control*, (Sept. 2022). DOI: 10.1177/00202940221097981.

[43] Shih Liang Chao and Pi Hung Lin. 2021. Minimizing overstowage in master bay plans of large container ships. *Maritime Economics and Logistics*, 23, 1, 71–93. Publisher: Palgrave Macmillan UK. DOI: 10.1057/s41278-019-00126-6.

[44] Rui Chen and James Luedtke. 2022. On sample average approximation for two-stage stochastic programs without relatively complete recourse. en. *Mathematical Programming*, 196, 1-2, (Nov. 2022), 719–754. DOI: 10.1007/s10107-021-01753-9.

[45] Shilin Chen, Jaya Krishnan, Jing Zhang, Sudipta Seal, Ian McGough, Wenyi Wang, and Kun Chen. 2021. A deep learning model for predicting next-generation sequencing depth from DNA probe sequences. *Nature Communications*, 12, 1, 1–10. Publisher: Nature Publishing Group. DOI: 10.1038/s41467-021-24497-8.

[46] Wenbo Chen, Mathieu Tanneau, and Pascal Van Hentenryck. 2024. End-to-End Feasible Optimization Proxies for Large-Scale Economic Dispatch. en. *IEEE Transactions on Power Systems*, 39, 2, 4723–4734. DOI: 10.1109/TPWRS.2023.3317352.

[47] DW Cho. 1981. *Development of a methodology for containership load planning*. PhD. Oregon State University. https://ir.library.oregonstate.edu/downloads/g158bm271.

[48] Chien Chang Chou and Pao Yi Fang. 2021. Applying expert knowledge to containership stowage planning: an empirical study. *Maritime Economics and Logistics*, 23, 1, 4–27. Publisher: Palgrave Macmillan UK. DOI: 10.1057/s41278-018-0113-0.

[49] J. Christensen, A. Erera, and D. Pacino. 2019. A rolling horizon heuristic for the stochastic cargo mix problem. *Transportation Research Part E: Logistics and Transportation Review*, 123. DOI: 10.1016/j.tre.2018.10.010.

[50] J. Christensen and D. Pacino. 2017. A matheuristic for the Cargo Mix Problem with Block Stowage. *Transportation Research Part E: Logistics and Transportation Review*, 97. DOI: 10.1016/j.tre.2016.10.005.

[51] European Commission. 2007. Freight transport logistics action plan. Tech. rep. COM(2007) 607. European Union, (Oct. 2007). https://eur-lex.europa.eu/EN/legal-content/summary/freight-transport-logistics-action-plan.html.

[52] Andrea Conca, Angela Di Febbraro, Davide Giglio, and Francesco Rebora. 2018. Automation in freight port call process: Real time data sharing to improve the stowage planning. In vol. 30. DOI: 10.1016/j.trpro.2018.09.009.

[53] Laura Cruz-reyes, Paula Hernández H, Patricia Melin, Héctor J Fraire H, and Julio Mar O. 2013. Constructive Algorithm for a Benchmark in Ship Stowage Planning, 393–408. DOI: 10.1007/978-3-642-33021-6_31.

[54] Laura Cruz-Reyes et al. 2015. Lower and Upper Bounds for the Master Bay Planning Problem. *International Journal of Combinatorial Optimization Problems and Informatics*, 6, 1, 42–52. https://ijcopi.org/ojs/article/view/61.

[55] Alberto Delgado, Rune Møller Jensen, and Nicolas Guilbert. 2012. A placement heuristic for a commercial decision support system for container vessel stowage. *38th Latin America Conference on Informatics, CLEI 2012 - Conference Proceedings*. DOI: 10.1109/CLEI.2012.6427181.

[56] Alberto Delgado, Rune Møller Jensen, Kira Janstrup, Trine Høyer Rose, and Kent Høj Andersen. 2012. A Constraint Programming model for fast optimal stowage of container vessel bays. *European Journal of Operational Research*, 220, 1, (July 2012), 251–261. DOI: 10.1016/j.ejor.2012.01.028.

[57] Alberto Delgado, Rune Møller Jensen, and Christian Schulte. 2009. Generating optimal stowage plans for container vessel bays. In vol. 5732 LNCS, 6–20. DOI: 10.1007/978-3-642-04244-7_4.

[58] Ding Ding and Mabel C. Chou. 2015. Stowage Planning for Container Ships: A Heuristic Algorithm to Reduce the Number of Shifts. *European Journal of Operational Research*. Publisher: Elsevier Ltd. DOI: 10.1016/j.ejor.2015.03.044.

[59] Dongsheng Ding, Kaiqing Zhang, Tamer Basar, and Mihailo Jovanovic. 2020. Natural Policy Gradient Primal-Dual Method for Constrained Markov Decision Processes. In *Advances in Neural Information Processing Systems*. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors. Vol. 33. Curran Associates, Inc., 8378–8390. https://proceedings.neurips.cc/paper_files/paper/2020/file/5f7695debd8cde8db5abcb9f161b49ea-Paper.pdf.

[60] Priya L. Donti, David Rolnick, and J. Zico Kolter. 2021. DC3: A learning method for optimization with hard constraints. en. In *Proceedings of the 9th International Conference on Learning Representations*. (Apr. 2021). DOI: 10.48550/arXiv.2104.12225.

[61] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. 1996. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26, 1, 29–41. DOI: 10.1109/3477.484436.

[62] Jay Doshi, Kunal Parmar, Raj Sanghavi, and Narendra Shekokar. 2023. A comprehensive dual-layer architecture for phishing and spam email detection. *Computers & Security*, 133, 103378. DOI: https://doi.org/10.1016/j.cose.2023.103378.

[63] Opher Dubrovsky, Gregory Levitin, and Michal Penn. 2002. A genetic algorithm with a compact solution encoding for the container ship stowage problem. *Journal of Heuristics*, 8, 6, 585–599. DOI: 10.1023/A:1020373709350.

[64] J. Dupacova, N. Gröwe-Kuska, and W. Römisch. 2003. Scenario reduction in stochastic programming. en. *Mathematical Programming*, 95, 3, (Mar. 2003), 493–511. DOI: 10.1007/s10107-002-0331-0.

[65] Amina El Yaagoubi, Mohamed Charhbili, Jaouad Boukachour, and Ahmed El Hilali Alaoui. 2022. Multi-objective optimization of the 3D container stowage planning problem in a barge convoy system. *Computers & Operations Research*, (Mar. 2022), 105796–105796. Publisher: Pergamon. DOI: 10.1016/J.COR.2022.105796.

[66] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. 2020. Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. In *Proceedings of the International Conference on Learning Representations*. (May 2020). https://openreview.net/forum?id=r1etN1rtPB.

[67] European Commission. 2023. Reducing Emissions from the Shipping Sector. (2023) https://climate.ec.europa.eu/eu-action/transport/reducing-emissions-shipping-sector_en#events.

[68] Veronika Eyring et al. 2024. Pushing the frontiers in climate modelling and analysis with machine learning. *Nature Climate Change*, 14, 9, (Sept. 2024), 916–928. DOI: 10.1038/s41558-024-02095-y.

[69] Simone Foa, Corrado Coppola, Giorgio Grani, and Laura Palagi. 2022. Solving the vehicle routing problem with deep reinforcement learning. *Computing Research Repository*, (July 2022). http://arxiv.org/abs/2208.00202.

[70] Janna Franzkeit, Anne Schwientek, and Carlos Jahn. 2020. Stowage Planning for Inland Container Vessels: A Literature Review. In Issue: September, 247–280. ISBN: 978-3-7531-2347-9.

[71] Rodolphe M. Freville. 2004. Hybrid Metaheuristics: An Emerging Approach to Optimization. In *Handbook of Metaheuristics*. Springer, 457–474. ISBN: 978-1-4419-1663-1.

[72] Yasuhiro Fujita and Shin-ichi Maeda. 2018. Clipped Action Policy Gradient. In *Proceedings of the 35th International Conference on Machine Learning*. (Feb. 2018). https://proceedings.mlr.press/v80/fujita18a/fujita18a.pdf.

[73] Rui Gao and Anton Kleywegt. 2023. Distributionally Robust Stochastic Optimization with Wasserstein Distance. *Mathematics of Operations Research*, 48, 2, 603–655. DOI: 10.1287/moor.2022.1275.

[74] Gendreau, M. and Potvin, J. Y. 2019. *Handbook of Metaheuristics*. (2nd ed.). Vol. 146. Springer. ISBN: 978-1-4419-1663-1.

[75] Fred Glover. 1989. Tabu Search—Part I. *ORSA Journal on Computing*, 1, 3, 190–206. Publisher: INFORMS. DOI: 10.1287/ijoc.1.3.190.

[76] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. ISBN: 978-0-262-03561-3. http://www.deeplearningbook.org.

[77] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2672–2680. https://papers.nips.cc/paper_files/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

[78] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, (Aug. 2018). DOI: 10.48550/arXiv.1801.01290.

[79] Ibrahim Y. Hafez, Ahmed Y. Hafez, Ahmed Saleh, Amr A. Abd El-Mageed, and Amr A. Abohany. 2025. A systematic review of AI-enhanced techniques in credit card fraud detection. *Journal of Big Data*, 12, 1, (Jan. 2025), 6. DOI: 10.1186/s40537-024-01048-8.

[80] Milad Haghani, Frances Sprei, Khashayar Kazemzadeh, Zahra Shahhoseini, and Jamshid Aghaei. 2023. Trends in electric vehicles research. *Transportation Research Part D: Transport and Environment*, 123, 103881. DOI: https://doi.org/10.1016/j.trd.2023.103881.

[81] Ben Hambly, Renyuan Xu, and Huining Yang. 2023. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33, 3, 437–503. DOI: https://doi.org/10.1111/mafi.12382.

[82] Masoud Hamedi. 2011. *CONTAINERSHIP LOAD PLANNING WITH CRANE OPERATIONS*. PhD. University of Maryland, USA.

[83] Mathias Offerlin Herup, Gustav Christian Wichmann Thiesgaard, Jaike Van Twiller, and Rune Møller Jensen. 2022. A Linear Time Algorithm for Optimal Quay Crane Scheduling. In *Computational Logistics*. Vol. 13557. Springer Nature Switzerland, Barcelona, Spain, 60–73. DOI: 10.1007/978-3-031-16579-5.

[84] John H. Holland. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press. ISBN: 978-0-262-27555-2.

[85] John N. Hooker. 2007. Planning and Scheduling to Minimize Tardiness. In *Principles and Practice of Constraint Programming* (Lecture Notes in Computer Science). Vol. 4741. Springer, 314–328. DOI: 10.1007/11564751_25.

[86] André Hottung, Yeong-Dae Kwon, and Kevin Tierney. 2022. Efficient Active Search for Combinatorial Optimization Problems. In *Proceedings of the International Conference on Learning Representations*. http://arxiv.org/abs/2106.05126.

[87] André Hottung, Shunji Tanaka, and Kevin Tierney. 2020. Deep learning assisted heuristic tree search for the container pre-marshalling problem. *Computers & Operations Research*, 113, 104781. DOI: https://doi.org/10.1016/j.cor.2019.104781.

[88] André Hottung and Kevin Tierney. 2019. Neural Large Neighborhood Search for the Capacitated Vehicle Routing Problem. In *Proceedings of the European Conference on Artificial Intelligence*. (Nov. 2019). DOI: 10.3233/FAIA200124.

[89] Hsien Pin Hsu, Chia Nan Wang, Hsin Pin Fu, and Thanh Tuan Dang. 2021. Joint scheduling of yard crane, yard truck, and quay crane for container terminal considering vessel stowage plan: An integrated simulation-based optimization approach. *Mathematics*, 9, 18. DOI: 10.3390/math9182236.

[90] Min Hu and Wei Cai. 2017. Multi-objective optimization based on improved genetic algorithm for containership stowage on full route. *2017 4th International Conference on Industrial Engineering and Applications, ICIEA 2017*, 224–228. DOI: 10.1109/IEA.2017.7939211.

[91] Wenbin Hu, Zhengbing Hu, Lei Shi, Peng Luo, and Wei Song. 2012. Combinatorial optimization and strategy for ship stowage and loading schedule of container terminal. *Journal of Computers*, 7, 8, 2078–2092. DOI: 10.4304/jcp.7.8.2078-2092.

[92] International Chamber of Shipping. 2023. Environmental Performance: Comparison of CO2 Emissions by Different Modes of Transport. https://www.ics-shipping.org/shipping-fact/environmental-performance-environmental-performance/.

[93] International Maritime Organization. 2004. International Convention for the Control and Management of Ships' Ballast Water and Sediments (BWM Convention). (2004). https://www.imo.org/en/OurWork/Environment/Pages/BallastWaterManagement.aspx.

[94] International Maritime Organization. 2023. Emission Control Areas (ECAs) designated under MARPOL Annex VI. (2023). https://www.imo.org/en/OurWork/Environment/Pages/Emission-Control-Areas-(ECAs)-designated-under-regulation-13-of-MARPOL-Annex-VI-(NOx-emission-control).aspx.

[95] International Maritime Organization. 2023. Ships' Routeing: Traffic Separation Schemes (TSS). (2023). https://www.imo.org/en/OurWork/Safety/Pages/ShipsRouteing.aspx.

[96] International Maritime Organization. 2024. *International Maritime Dangerous Goods Code (IMDG Code)*. (42nd ed.). International Maritime Organization, London, UK. ISBN: 978-92-801-1797-4.

[97] International Organization for Standardization. 2020. ISO 668:2020 - Series 1 freight containers — Classification, dimensions and ratings. (2020). https://www.iso.org/standard/80539.html.

[98] Cagatay Iris, Jonas Christensen, Dario Pacino, and Stefan Ropke. 2018. Flexible ship loading problem with transfer vehicle assignment and scheduling. *Transportation Research Part B: Methodological*, 111, (May 2018), 113–134. Publisher: Elsevier Ltd. DOI: 10.1016/j.trb.2018.03.009.

[99] Isabel Correia, Isabel Correia, Francisco Saldanha-da-Gama, and Francisco Saldanha da Gama. 2015. Facility Location Under Uncertainty, (Jan. 2015), 177–203. MAG ID: 412237700. DOI: 10.1007/978-3-319-13111-5_8.

[100] Kamal Jain and Vijay V. Vazirani. 2001. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48, 2, 274–296. Publisher: ACM. DOI: 10.1145/375827.375845.

[101] Rune Møller Jensen and Mai Lise Ajspur. 2018. The Standard Capacity Model: Towards a Polyhedron Representation of Container Vessel Capacity. In *Computational Logistics*. Vol. 8197. Springer International Publishing, 175–190. ISBN: 978-3-642-41018-5. DOI: 10.1007/978-3-030-00898-7.

[102] Rune Møller Jensen and Mai Lise Ajspur. 2022. Revenue management in liner shipping: Addressing the vessel capacity challenge. *Maritime Transport Research*, 3, (Jan. 2022). Publisher: Elsevier Ltd. DOI: 10.1016/j.martra.2022.100069.

[103] Rune Moller Jensen, Eilif Leknes, and Tom Bebbington. 2012. Fast interactive decision support for modifying stowage plans using binary decision diagrams. In vol. 2196, 1555–1561. ISBN: 978-988-19251-9-0.

[104] Rune Møller Jensen, Dario Pacino, Mai Lise Ajspur, and Claus Vesterdal. 2018. *Container Vessel Stowage Planning*. Weilbach. ISBN: 978-87-7790-311-3.

[105] Jian Jin and Weijian Mi. 2019. An AIMMS-based decision-making model for optimizing the intelligent stowage of export containers in a single bay. *Discrete and Continuous Dynamical Systems - Series S*, 12, 4-5, 1101–1115. DOI: 10.3934/dcdss.2019076.

[106] John Jumper et al. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596, 583–589. DOI: 10.1038/s41586-021-03819-2.

[107] Evangelos I. Kaisar. 2006. A STOWAGE PLANNING MODEL FOR MULTIPORT CONTAINER TRANSPORTATION. *University of Maryland*. http://drum.lib.umd.edu/handle/1903/9139.

[108] Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. 2020. Action Space Shaping in Deep Reinforcement Learning. en. In *2020 IEEE Conference on Games (CoG)*. (May 2020). https://api.semanticscholar.org/CorpusID:214775114.

[109] J-G Kang and Y-D Kim. 2002. Stowage planning in maritime container transportation. *Journal of the Operational Research Society*, 53, 4, 415–426. http://www.ingentaconnect.com/content/pal/01605682/2002/00000053/00000004/2601322.

[110] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. 2023. Champion-level drone racing using deep reinforcement learning. *Nature*, 620, 123–129. DOI: 10.1038/s41586-023-06419-4.

[111] Kiros Gebrearegawi Kebedow and Johan Oppen. 2018. Including containers with dangerous goods in the multi-port master bay planning problem. *Mendel*, 24, 2, 23–36. DOI: 10.13164/mendel.2018.2.023.

[112] Kiros Gebrearegawi Kebedow and Johan Oppen. 2019. Including containers with dangerous goods in the cargo mix problem for container vessel stowage. *Communications - Scientific Letters of the University of Zilina*, 21, 2, 100–113. DOI: 10.26552/com.c.2019.2.100-113.

[113] Kiros Gebrearegawi Kebedow and Johan Oppen. 2019. Including containers with dangerous goods in the slot planning problem. In *Proceedings of the International Conference on Industrial Engineering and Operations Management*. Vol. 2019. Number: MAR, 225–232. http://www.ieomsociety.org/ieom2019/papers/74.pdf.

[114] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 137–146. DOI: 10.1145/956750.956769.

[115] James Kennedy and Russell Eberhart. 1995. Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, 4, 1942–1948. DOI: 10.1109/ICNN.1995.488968.

[116] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 55, 4559–4604.

[117] Song-Kyoo Kim, Chan Yeob Yeun, Ernesto Damiani, and Nai-Wei Lo. 2019. A machine learning framework for biometric authentication using electrocardiogram. *IEEE access : practical innovations, open solutions*, 7, 94858–94868. DOI: 10.1109/ACCESS.2019.2927079.

[118] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*. (Dec. 2014). http://arxiv.org/abs/1412.6980.

[119] Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. In *Proceedings of the 29th International Conference on Machine Learning*. https://arxiv.org/abs/1312.6114.

[120] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1609.02907.

[121] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. 1983. Optimization by Simulated Annealing. *Science*, 220, 4598, 671–680. Publisher: AAAS. DOI: 10.1126/science.220.4598.671.

[122] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. 2022. A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. en. *Electronics*, 11, 1, (Jan. 2022), 141. DOI: 10.3390/electronics11010141.

[123] Wouter Kool, Herke van Hoof, and Max Welling. 2019. Attention, Learn to Solve Routing Problems! In *Proceedings of the International Conference on Learning Representations*. https://openreview.net/forum?id=ByxBFsRqYm.

[124] Aleksandra Korach, Berit Dangaard Brouer, and Rune Møller Jensen. 2020. Matheuristics for slot planning of container vessel bays. *European Journal of Operational Research*, 282, 3, 873–885. Publisher: Elsevier B.V. DOI: 10.1016/j.ejor.2019.09.042.

[125] Christian Kroer, Martin Kjaer Svendsen, Rune Møller Jensen, and Joseph Roland Kiniry. 2012. SAT and SMT-based Interactive Configuration for Container Vessel Stowage Planning. Tech. rep. http://www.columbia.edu/~ck2945/papers/KroerSvendsen12.pdf.

[126] Christian Kroer, Martin Kjær Svendsen, Rune M. Jensen, Joseph Kiniry, and Eilif Leknes. 2016. Symbolic configuration for interactive container ship stowage planning. *Computational Intelligence*, 32, 2, 259–283. DOI: 10.1111/coin.12051.

[127] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. 2020. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems*. https://arxiv.org/abs/2010.16011.

[128] Ailsa H. Land and Alison G. Doig. 1960. An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28. Publication Title: Econometrica. DOI: 10.1007/978-3-540-68279-0_5.

[129] Gilbert Laporte. 2009. Fifty years of vehicle routing. *Transportation Science*, 43, 4, 408–416. DOI: 10.1287/trsc.1090.0301.

[130] Rune Larsen and Dario Pacino. 2021. A heuristic and a benchmark for the stowage planning problem. *Maritime Economics and Logistics*, 23, 1, 94–122. DOI: 10.1057/s41278-020-00172-5.

[131] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 11, 2278–2324. DOI: 10.1109/5.726791.

[132] Chaemin Lee, Mun Keong Lee, and Jae Young Shin. 2020. Lashing Force Prediction Model with Multimodal Deep Learning and AutoML for Stowage Planning Automation in Containerships. *Logistics*, 5, 1, 1–1. DOI: 10.3390/logistics5010001.

[133] Marc Levinson. 2016. *The Box*. (REV - Revised, 2 ed.). Princeton University Press. ISBN: 978-0-691-17081-7. DOI: 10.2307/j.ctvcszztg.

[134] Jun Li, Yu Zhang, Sanyou Ji, and Lanbo Zheng. 2020. Solving inland container ship stowage planning problem on full route through a two-phase approach. In vol. 12. Issue: 1-2, 65–91. DOI: 10.1504/IJSTL.2020.105863.

[135] Jun Li, Yu Zhang, Jie Ma, and Sanyou Ji. 2018. Multi-Port Stowage Planning for Inland Container Liner Shipping Considering Weight Uncertainties. *IEEE Access*, 6, 66468–66480. Publisher: Institute of Electrical and Electronics Engineers Inc. DOI: 10.1109/ACCESS.2018.2878308.

[136] Meiyi Li, Soheil Kolouri, and Javad Mohammadi. 2023. Learning to Solve Optimization Problems With Hard Linear Constraints. en. *IEEE Access*, 11, 59995–60004. DOI: 10.1109/ACCESS.2023.3285199.

[137] Fan Liu, Malcolm Yoke Hean Low, Wen Jing Hsu, Shell Ying Huang, Min Zeng, and Cho Aye Win. 2011. Randomized algorithm with tabu search for multi-objective optimization of large containership stowage plans. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6971 LNCS, 256–272. DOI: 10.1007/978-3-642-24264-9_20.

[138] Lloyd's List. 2022. Shipping emissions rise 4.9% in 2021. (Jan. 2022). https://lloydslist.com/LL1139627/Shipping-emissions-rise-49-in-2021.

[139] Andrea Lodi and Giulia Zarpellon. 2017. On learning and branching: a survey. *TOP*, 25, 2, (July 2017), 207–236. DOI: 10.1007/s11750-017-0451-6.

[140] Logistics eLearning. 2023. Largest container ships by year. (2023). https://logisticselearning.com/largest-container-ships-by-year/.

[141] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2018. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13, 3, e0194889.

[142] Vittorio Maniezzo, Marco Antonio Boschetti, and Thomas St"utzle. 2021. *Matheuristics: Algorithms and Implementations*. Springer International Publishing. ISBN: 978-3-030-70276-2. DOI: 10.1007/978-3-030-70276-2.

[143] Harry Markowitz. 1952. Portfolio selection. *The Journal of Finance*, 7, 1, 77–91. DOI: 10.2307/2975974.

[144] Silvano Martello and Paolo Toth. 1990. *Knapsack problems: Algorithms and computer implementations*. Wiley. ISBN: 978-0-471-92420-3.

[145] Gifford L. Martin, Sabah U. Randhawa, and Edward D. McDowell. 1988. Computerized containership load planning: A methodology and evaluation. *Computers and Industrial Engineering*, 14, 4, 429–440. DOI: 10.1016/0360-8352(88)90045-9.

[146] Bernardo Martin-Iradi, Dario Pacino, and Stefan Ropke. 2022. The Multiport Berth Allocation Problem with Speed Optimization: Exact Methods and a Cooperative Game Analysis. *Transportation Science*, 56, 4, 972–999. DOI: 10.1287/trsc.2021.1112.

[147] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134, (Oct. 2021), 105400–105400. DOI: 10.1016/j.cor.2021.105400.

[148] Azalia Mirhoseini et al. 2021. A graph placement methodology for fast chip design. *Nature*, 594, 7862, 207–212. DOI: 10.1038/s41586-021-03544-w.

[149] Nenad Mladenović and Pierre Hansen. 1997. Variable Neighborhood Search. *Computers & Operations Research*, 24, 11, 1097–1100. Publisher: Elsevier. DOI: 10.1016/S0305-0548(97)00031-2.

[150] Volodymyr Mnih et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518, 7540, (Feb. 2015), 529–533. DOI: 10.1038/nature14236.

[151] Peyman Mohajerin Esfahani and Daniel Kuhn. 2018. Data-driven distributionally robust optimization using the Wasserstein metric: performance guarantees and tractable reformulations. en. *Mathematical Programming*, 171, 1-2, (Sept. 2018), 115–166. DOI: 10.1007/s10107-017-1172-1.

[152] Maria Flavia Monaco, Marcello Sammarra, and Gregorio Sorrentino. 2014. The Terminal-Oriented Ship Stowage Planning Problem. *European Journal of Operational Research*, (May 2014). Publisher: Elsevier B.V. DOI: 10.1016/j.ejor.2014.05.030.

[153] MohammadReza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takac. 2018. Reinforcement Learning for Solving the Vehicle Routing Problem. In *Advances in Neural Information Processing Systems*. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors. Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/9fb4651c05b2ed70fba5afe0b039a550-Paper.pdf.

[154] Nisan, Noam, Roughgarden, Tim, Tardos, Éva, and Vazirani, Vijay V. 2007. *Algorithmic game theory*. en. Cambridge university press, New York. ISBN: 978-0-521-87282-9.

[155] S Nugroho, E B Djatmiko, Murdjito, E W Ardhi, H Supomo, and I G N S Buana. 2021. Regulatory framework of a computer-based stowage planning: safety and efficiency considerations. *IOP Conference Series: Materials Science and Engineering*, 1052, 1, 012065–012065. DOI: 10.1088/1757-899x/1052/1/012065.

[156] OpenAI. 2023. GPT-4 technical report. (2023). https://arxiv.org/abs/2303.08774.

[157] David W. Otter, Julian R. Medina, and Jugal K. Kalita. 2020. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32, 2, 604–624. DOI: 10.1109/TNNLS.2020.2979670.

[158] Dario Pacino. 2013. An LNS Approach for Container Stowage Multi-port Master Planning. In *Computational Logistics*, 35–44. DOI: 10.1007/978-3-642-41019-2_3.

[159] Dario Pacino. 2018. Crane Intensity and Block Stowage Strategies in Stowage Planning. In *Computational Logistics*. Vol. 11184 LNCS. Springer, 191–206. DOI: 10.1007/978-3-030-00898-7_12.

[160] Dario Pacino, Alberto Delgado, RM Jensen, and Tom Bebbington. 2011. Fast generation of near-optimal plans for eco-efficient stowage of large container vessels. *Computational Logistics*, 286–301. http://link.springer.com/chapter/10.1007/978-3-642-24264-9_22.

[161] Dario Pacino, Alberto Delgado, Rune Møller Jensen, and Tom Bebbington. 2012. An accurate model for seaworthy container vessel stowage planning with ballast tanks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7555 LNCS, 17–32. DOI: 10.1007/978-3-642-33587-7_2.

[162] Dario Pacino and Rune Møller Jensen. 2010. A 3-Phase Randomized Constraint Based Local Search Algorithm for Stowing Under Deck Locations of Container Vessel Bays. (2010). https://pure.itu.dk/files/108203401/ITU-TR-2010-123_pdf.pdf.

[163] Dario Pacino and Rune Møller Jensen. 2013. Fast slot planning using constraint-based local search. *Lecture Notes in Electrical Engineering*, 186 LNEE, 49–63. DOI: 10.1007/978-94-007-5651-9-4.

[164] Manfred Padberg and Giovanni Rinaldi. 1991. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, 33, 1. Publication Title: SIAM Review. http://www.jstor.org/stable/2030652.

[165] Anthony Papavasiliou and Shmuel S. Oren. 2012. A stochastic unit commitment model for integrating renewable supply and demand response. In *2012 IEEE Power and Energy Society General Meeting*, 1–6. DOI: 10.1109/PESGM.2012.6344858.

[166] Francisco Parreño, Dario Pacino, and Ramon Alvarez-Valdes. 2016. A GRASP algorithm for the container stowage slot planning problem. *Transportation Research Part E: Logistics and Transportation Review*, 94, 141–157. DOI: 10.1016/j.tre.2016.07.011.

[167] C Parreño-Torres. 2020. Improving container terminal efficiency: New models and algorithms for Premarshalling and Stowage Problems. April. http://roderic.uv.es/handle/10550/75440.

[168] Consuelo Parreño-Torres, Ramon Alvarez-Valdes, and Francisco Parreño. 2019. Solution strategies for a multiport container ship stowage problem. *Mathematical Problems in Engineering*, 2019. DOI: 10.1155/2019/9029267.

[169] Consuelo Parreño-Torres, Hatice Çalık, Ramon Alvarez-Valdes, and Rubén Ruiz. 2021. Solving the generalized multi-port container stowage planning problem by a matheuristic algorithm. *Computers and Operations Research*, 133, 105383–105383. Publisher: Elsevier Ltd. DOI: 10.1016/j.cor.2021.105383.

[170] M. V. F. Pereira and L. M. V. G. Pinto. 1991. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52, 1, (May 1991), 359–375. DOI: 10.1007/BF01582895.

[171] Michael L. Pinedo. 2016. *Scheduling: Theory, algorithms, and systems*. (5th ed.). Springer. ISBN: 978-3-319-26580-3.

[172] David Pisinger and Stefan Røpke. 2010. Large Neighborhood Search. In *Handbook of Metaheuristics*. (2nd ed.). Springer, 399–420. ISBN: 978-1-4419-1663-1.

[173] Shubham Suresh Pol and Avtar Singh. 2021. Task scheduling algorithms in cloud computing: a survey. In *2021 2nd international conference on secure cyber computing and communications (ICSCCC)*, 244–249. DOI: 10.1109/ICSCCC51823.2021.9478160.

[174] Momina Qureshi, Masood Ahmad Arbab, and Sadaqat ur Rehman. 2024. Deep learning-based forecasting of electricity consumption. *Scientific Reports*, 14, 1, (Mar. 2024), 6489. DOI: 10.1038/s41598-024-56602-4.

[175] Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. 2017. The Benders decomposition algorithm: A literature review. en. *European Journal of Operational Research*, 259, 3, (June 2017), 801–817. DOI: 10.1016/j.ejor.2016.12.005.

[176] Dalia Rashed, Amr Eltawil, and Mohamed Gheith. 2021. A Fuzzy Logic-Based Algorithm to Solve the Slot Planning Problem in Container Vessels. *Logistics*, 5, 4, 67–67. DOI: 10.3390/logistics5040067.

[177] R. Roberti and D. Pacino. 2018. A decomposition method for finding optimal container stowage plans. *Transportation Science*, 52, 6, 1444–1462. DOI: 10.1287/trsc.2017.0795.

[178] R. T. Rockafellar and Roger J.-B. Wets. 1991. Scenarios and Policy Aggregation in Optimization Under Uncertainty. en. *Mathematics of Operations Research*, 16, 1, (Feb. 1991), 119–147. DOI: 10.1287/moor.16.1.119.

[179] Werner Römisch. 2009. Scenario Reduction Techniques in Stochastic Programming. en. In *Stochastic Algorithms: Foundations and Applications*. Vol. 5792. Osamu Watanabe and Thomas Zeugmann, editors. Series Title: Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–14. ISBN: 978-3-642-04943-9 978-3-642-04944-6. DOI: 10.1007/978-3-642-04944-6_1.

[180] Francesca Rossi, Peter Van Beek, and Toby Walsh. 2006. *Handbook of Constraint Programming*. Elsevier. ISBN: 978-0-444-52726-4.

[181] Michael H. Rothkopf, Aleksandar Pekec, and Ronald M. Harstad. 1998. Computationally manageable combinational auctions. *Management Science*, 44, 8, 1131–1147. https://www.jstor.org/stable/2634691.

[182] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323, 6088, 533–536. DOI: 10.1038/323533a0.

[183] S Russell and P Norvig. 2020. *Artificial Intelligence: A Modern Approach*. (4th edition ed.). Pearson. ISBN: 978-0-13-461099-3. http://aima.cs.berkeley.edu/.

[184] Utsav Sadana, Abhilash Chenreddy, Erick Delage, Alexandre Forel, Emma Frejinger, and Thibaut Vidal. 2025. A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 320, 2, 271–289. DOI: https://doi.org/10.1016/j.ejor.2024.03.020.

[185] D. J. Saginaw and A. N. Perakis. 1989. Decision support system for containership stowage planning. *Marine Technology and SNAME News*, 26, 1, 47–61. DOI: 10.5957/mt1.1989.26.1.47.

[186] Lorenzo Schena, Pedro A. Marques, Romain Poletti, Samuel Ahizi, Jan Van den Berghe, and Miguel A. Mendez. 2024. Reinforcement Twinning: From digital twins to model-based reinforcement learning. *Journal of Computational Science*, 82, 102421. DOI: https://doi.org/10.1016/j.jocs.2024.102421.

[187] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *4th International Conference on Learning Representations*. http://arxiv.org/abs/1506.02438.

[188] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347. (July 2017). DOI: doi.org/10.48550/arXiv.1707.06347.

[189] Anna Sciomachen and Elena Tanfani. 2003. The master bay plan problem: A solution method based on its connection to the three-dimensional bin packing problem. *IMA Journal of Management Mathematics*, 14, 3, 251–269. DOI: 10.1093/imaman/14.3.251.

[190] Anna Sciomachen and Elena Tanfani. 2007. A 3D-BPP approach for optimising stowage plans and terminal productivity. *European Journal of Operational Research*, 183, 1433–1446. DOI: 10.1016/j.ejor.2005.11.067.

[191] Cristina Serban and Doina Carp. 2017. A genetic algorithm for solving a container storage problem using a residence time strategy. *Studies in Informatics and Control*, 26, 1, 59–66. DOI: 10.24846/v26i1y201707.

[192] A. Shapiro. 2003. Monte Carlo sampling approach to stochasticprogramming. en. *ESAIM: Proceedings*, 13, (Dec. 2003), 65–73. J.P. Penot, editor. DOI: 10.1051/proc:2003003.

[193] Alexander Shapiro. 2011. Analysis of stochastic dual dynamic programming method. en. *European Journal of Operational Research*, 209, 1, (Feb. 2011), 63–72. DOI: 10.1016/j.ejor.2010.08.007.

[194] Yifan Shen, Ning Zhao, Mengjue Xia, and Xueqiang Du. 2017. A deep Q-learning network for ship stowage planning problem. *Polish Maritime Research*, 24, S3, 102–109. DOI: 10.1515/pomr-2017-0111.

[195] JJ Shields. 1984. Containership Stowage: A Computer-Aided Preplanning System. http://trid.trb.org/view.aspx?id=419881.

[196] David Silver et al. 2017. Mastering the game of Go without human knowledge. *Nature*, 550, 7676, (Oct. 2017), 354–359. DOI: 10.1038/nature24270.

[197] Keerthana Sivamayil, Elakkiya Rajasekar, Belqasem Aljafari, Srete Nikolovski, Subramaniyaswamy Vairavasundaram, and Indragandhi Vairavasundaram. 2023. A Systematic Study on Reinforcement Learning Based Applications. en. *Energies*, 16, 3. DOI: 10.3390/en16031512.

[198] A. Sivertsen, L. Reinhardt, and RM Jensen. 2025. A representative model and benchmark suite for the container stowage planning problem. *Under review*.

[199] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2023. *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press. ISBN: 978-0-262-04861-3.

[200] Xu-yan Song, Xiao-chen Dou, Yuan Ren, and Xiao Liu. 2010. Research on application of simulation technology in container ship stowage problem of port logistics. *Proceedings IE & EM 2010 : 2010 IEEE 17th International Conference on Industrial Engineering and Engineering Management*, 29–31. Publisher: IEEE.

[201] Statista. 2023. Global container ship CO2 emissions by month 2023. (2023). https://www.statista.com/statistics/1480859/monthly-shipping-emissions-worldwide-container-ships/.

[202] Statista. 2024. Container shipping worldwide. Tech. rep. https://www.statista.com/topics/1367/container-shipping/#topicOverview.

[203] Roland Stolz, Hanna Krasowski, Jakob Thumm, Michael Eichelbeck, Philipp Gassert, and Matthias Althoff. 2025. Excluding the Irrelevant: Focusing Reinforcement Learning through Continuous Action Masking. In *Proceedings of the 38th Conference on Neural Information Processing Systems*. DOI: 10.48550/arXiv.2406.03704.

[204] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 3104–3112. DOI: 10.5555/2969033.2969173.

[205] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press. ISBN: 978-0-262-03924-6.

[206] Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. 2024. Deep reinforcement learning for robotics: a survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*. DOI: https://doi.org/10.1146/annurev-control-030323-022510.

[207] Ran Tao, Pan Zhao, Jing Wu, Nicolas Martin, Matthew T. Harrison, Carla Ferreira, Zahra Kalantari, and Naira Hovakimyan. 2023. Optimizing Crop Management with Reinforcement Learning and Imitation Learning. en. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Macau, SAR China, (Aug. 2023), 6228–6236. ISBN: 978-1-956792-03-4. DOI: 10.24963/ijcai.2023/691.

[208] The Economist. 2013. Free exchange - The humble hero. *The Economist*, (May 2013). https://www.economist.com/finance-and-economics/2013/05/18/the-humble-hero.

[209] Theo Notteboom, Athanasios Pallis, and Jean-Paul Rodrigue. 2022. *Port Economics, Management and Policy*. English. (1st ed.). Routledge. ISBN: 978-0-429-31818-4. DOI: 10.4324/9780429318184.

[210] Simon Thevenin, Yossiri Adulyasak, and Jean-François Cordeau. 2022. Stochastic Dual Dynamic Programming for Multiechelon Lot Sizing with Component Substitution. en. *INFORMS Journal on Computing*, 34, 6, (Nov. 2022), 3151–3169. DOI: 10.1287/ijoc.2022.1215.

[211] Kevin Tierney, Dario Pacino, and Rune Møller Jensen. 2014. On the complexity of container stowage planning problems. *Discrete Applied Mathematics*, 169, 225–230. DOI: 10.1016/j.dam.2014.01.005.

[212] Massimo Tipaldi, Raffaele Iervolino, and Paolo Roberto Massenio. 2022. Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges. *Annual Reviews in Control*, 54, 1–23. DOI: https://doi.org/10.1016/j.arcontrol.2022.07.004.

[213] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. 2020. Deepfakes and beyond: A Survey of face manipulation and fake detection. *Information Fusion*, 64, 131–148. DOI: https://doi.org/10.1016/j.inffus.2020.06.014.

[214] Eric J. Topol. 2019. High-performance medicine: the convergence of human and artificial intelligence. *Nature Medicine*, 25, 44–56.

[215] United Nations Conference on Trade and Development. 2021. Review of Maritime Transport 2021. Tech. rep. United Nations. https://unctad.org/publication/review-maritime-transport-2021.

[216] United Nations Conference on Trade and Development. 2023. Container port throughput, annual. (2023). https://unctadstat.unctad.org/datacentre/dataviewer/US.ContPortThroughput.

[217] Jaike Van Twiller, Yossiri Adulyasak, Erick Delage, Djordje Grbic, and Rune Møller Jensen. 2025. Navigating Demand Uncertainty in Container Shipping: Deep Reinforcement Learning for Enabling Adaptive and Feasible Master Stowage Planning. en. arXiv:2502.12756 [cs]. (Feb. 2025). DOI: 10.48550/arXiv.2502.12756.

[218] Jaike Van Twiller, Djordje Grbic, and Rune Moller Jensen. 2025. AI2STOW: End-to-End Deep Reinforcement Learning to Construct Master Stowage Plans under Demand Uncertainty. Under Review. (2025).

[219] Jaike Van Twiller, Djordje Grbic, and Rune Møller Jensen. 2023. Towards a Deep Reinforcement Learning Model of Master Bay Stowage Planning. en. In *Computational Logistics*. Vol. 14239. Series Title: Lecture Notes in Computer Science. Springer Nature Switzerland, Berlin, Germany, 105–121. DOI: 10.1007/978-3-031-43612-3_6.

[220] Jaike Van Twiller, Agnieszka Sivertsen, Rune M. Jensen, and Kent H. Andersen. 2024. An Efficient Integer Programming Model for Solving the Master Planning Problem of Container Vessel Stowage. In *Computational Logistics*. Springer Nature Switzerland, Monterrey, Mexico, 236–253. DOI: 10.1007/978-3-031-71993-6_16.

[221] Jaike Van Twiller, Agnieszka Sivertsen, Dario Pacino, and Rune Møller Jensen. 2024. Literature survey on the container stowage planning problem. *European Journal of Operational Research*, 317, 3, 841–857. DOI: https://doi.org/10.1016/j.ejor.2023.12.018.

[222] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[223] Vijay V. Vazirani. 2001. *Approximation Algorithms*. Springer. ISBN: 978-3-642-08469-0.

[224] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Advances in Neural Information Processing Systems*. C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors. Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

[225] Yixuan Wang, Simon Sinong Zhan, Ruochen Jiao, Zhilu Wang, Wanxin Jin, Zhuoran Yang, Zhaoran Wang, Chao Huang, and Qi Zhu. 2023. Enforcing Hard Constraints with Soft Barriers: Safe Reinforcement Learning in Unknown Stochastic Environments. In *Proceedings of the 40th International Conference on Machine Learning*. DOI: 10.5555/3618408.3619930.

[226] I D Wilson and P A Roach. 2000. Container stowage planning: a methodology for generating computerised solutions. *Journal of the Operational Research Society*, 51, 11, (Nov. 2000), 1248–1255. DOI: 10.1057/palgrave.jors.2601022.

[227] ID Wilson, PA Roach, and JA Ware. 2001. Container stowage pre-planning: using search to generate solutions, a case study. *Knowledge-Based Systems*, 14, 3-4, 137–145. http://www.sciencedirect.com/science/article/pii/S0950705101000909.

[228] Wolsey, L. A. 2020. *Integer Programming*. (2nd ed.). Wiley, (Sept. 2020). ISBN: 978-1-119-60653-6.

[229] Allen J. Wood, Bruce F. Wollenberg, and Gerald B. Sheblé. 2013. *Power generation, operation, and control*. (3rd ed.). Wiley. ISBN: 978-0-471-79055-6.

[230] Qingcai Wu, Qiucheng Xia, and Maochuan Wu. 2021. Research on intelligent loading system for container ships. *IOP Conference Series: Earth and Environmental Science*, 632, 2. DOI: 10.1088/1755-1315/632/2/022074.

[231] Zhikuang Xin, Zhenghong Wu, Dong Zhu, Xiaoguang Wang, Jue Wang, and Yangang Wang. 2024. Reinforcement learning for scientific application: a survey. In *Knowledge science, engineering and management: 17th international conference, KSEM 2024, birmingham, UK, august 16–18, 2024, proceedings, part V*. Springer-Verlag, Birmingham, United Kingdom and Berlin, Heidelberg, 188–202. ISBN: 978-981-97-5488-5. DOI: 10.1007/978-981-97-5489-2_17.

[232] Shen Yifan, Zhao Ning, and Mi Weijian. 2016. Group-Bay Stowage Planning Problem for Container Ship. *Polish Maritime Research*, 23, s1, 152–159. DOI: 10.1515/pomr-2016-0060.

[233] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning-based natural language processing. *IEEE Computational Intelligence Magazine*, 13, 3, 55–75. DOI: 10.1109/MCI.2018.2840738.

[234] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. 2021. Reinforcement learning in healthcare: a survey. *Acm Computing Surveys*, 55, 1, (Nov. 2021). DOI: 10.1145/3477600.

[235] Wei Ying Zhang, Yan Lin, Zhuo Shang Ji, and Guang Fa Zhang. 2008. Review of containership stowage plans for full routes. *Journal of Marine Science and Application*, 7, 278–285. DOI: 10.1007/s11804-008-7087-8.

[236] Ning Zhao, Yuechao Guo, Tianyu Xiang, Mengjue Xia, Yifan Shen, and Chao Mi. 2018. Container Ship Stowage Based on Monte Carlo Tree Search. *Journal of Coastal Research*, 83, 540–547. DOI: 10.2112/SI83-090.1.

[237] Huiling Zhu, Mingjun Ji, and Wenwen Guo. 2020. Integer Linear Programming Models for the Containership Stowage Problem. *Mathematical Problems in Engineering*, 2020. DOI: 10.1155/2020/4382745.

[238] Sebastian Zurheide and Kathrin Fischer. 2015. Revenue management methods for the liner shipping industry. *Flexible Services and Manufacturing Journal*, 27, 2-3, (Sept. 2015), 200–223. Publisher: Springer New York LLC. DOI: 10.1007/s10696-014-9192-0.

Appendix A

Appendices of Literature Review

A.1 Experimental Results of Multi-Port Planning

TABLE A.1: Reported results from multi-port planning of [160] and [109]

TEUs	Num. ports	Solve time (sec.)
Results from [160]		
4755	4	10
9618	4	30
9984	4	21
2584	5	3
4755	5	21
4456	6	16
6545	6	5
8490	6	31
6717	7	23
7490	8	10
4478	9	5
5047	9	263
5052	9	214
4478	10	332
7344	10	252
9160	10	2079
9118	11	3711
9118	11	*timeout*
5044	14	*timeout*
9160	16	69
Results from [109]		
2500	4	32
3000	4	21
4000	4	32
2500	6	119
3000	6	121
4000	6	123
2500	8	366
3000	8	381
4000	8	399

A.2 Network-Flow Model

The model is based on a network-flow formulation. For a detailed description of the network structure, we refer the reader to the original publication [43]. Following is an extension of the common sets and parameters needed for this formulation.

Sets	
N	The set of all nodes
P	The set of ports
$\mathcal{A}$	The set of all arcs
$\mathcal{A}_i^-, \mathcal{A}_i^+$	The set of outgoing and incoming arcs of node $i \in N$
OD	The set of transports (origin/destination port pairs) $(o,d) \in P \times P$
OD^{ON}	The set of transports $t = (o,d) \in OD$ for which $o \leq p$ and $d > p$
OD^{A}	The set of transports $t = (o,d) \in OD$ for which $o = p$ or $d = p$
A_t^{TR}	The set of all arcs belonging to the transport $t \in OD$
A_t^{τ}	The set of arcs connecting the transport nodes and the container type nodes
A_{tl}^{L}	The set of arcs connecting the container type nodes and the block nodes
K_t	The number of containers in transport $t \in OD$
E_{ij}	The number of containers flowing through the type nodes
Parameters	
S	The source node
T	The sink node
q_i	Is equal to q for $i = S$, $-q$ for $i = T$ and zero for all other nodes, where q is the total number of containers to be stowed
$\tau^L(i)$	The length of container type node $i \in T$
$\tau^W(i)$	The weight of container type node $i \in T$
TEU_i	The TEU value of container type node $i \in T$

As the following is a network-flow formulation, the decision variables $x_{ij} \in \mathbb{R}^+$ represent the flow (the number) of containers from the source to the sink node, through the arcs $(i,j) \in A$.

$$\textbf{min} \sum_{p \in P} C^T y_p^T \tag{A.1}$$

s.t.

$$\sum_{(i,j) \in \mathcal{A}_i^-} x_{ij} - \sum_{(i,j) \in \mathcal{A}_i^+} x_{ij} = q_i \quad \forall i \in N \tag{A.2}$$

$$x_{ij} = K_t \quad \forall t \in OD, (i,j) \in A_t^{TR} \tag{A.3}$$

$$x_{ij} = E_{ij} \quad \forall t \in OD, (i,j) \in A_t^{\tau} \tag{A.4}$$

$$\sum_{t \in OD^{ON_p}} \sum_{(i,j) \in A_{tl}^L} TEU_i x_{ij} \leq C_l^{20} \quad \forall p \in P, l \in L \tag{A.5}$$

$$\sum_{t \in OD^{ON_p}} \sum_{(i,j) \in A_{tl}^L, \tau(i) = \alpha} x_{ij} \leq C_l^{\alpha} \quad \forall p \in P, l \in L, \alpha \in \{20, 40, R\} \tag{A.6}$$

$$\sum_{t \in OD_p^{OD}} \sum_{(i,j) \in A_{tl}^L} \tau^W(i) x_{ij} \leq W_l^{Max} \quad \forall p \in P, l \in L \tag{A.7}$$

$$x_{ij} \leq M y_{ij} \quad \forall p \in P, t \in OD_p^A, (i,j) \in A_{tl}^{LU} \tag{A.8}$$

$$\sum_{t \in OD_p^O} \sum_{(i,j) \in A_{tl}^{LO}} x_{ij} \leq M(1 - y_{ij}) \quad \forall p \in P, t \in OD_p^A, (i,j) \in A_{tl}^{LU} \tag{A.9}$$

$$\sum_{b \in N} \sum_{l \in BL_b} \sum_{A_{tl}^L} x_{ij} \leq y_p^T \quad \forall p \in P, N \in Bin \tag{A.10}$$

The model's objective is the minimization of the makespan at each port (A.1). This objective differs from the original publication, but this change was necessary for the sake of the comparison. Constraint (A.2) is the classic flow-conservation constraint between the source and the sink node. To ensure that cargo is correctly routed through the network, it is necessary to constraint containers within their respective origin-destination arcs (A.3), and the correct container type arcs (A.4). Constraints (A.5), (A.6), and (A.7) are the revised capacity and weight constraints. The absence of hatch=overstowage is ensured by constraints (A.8), which indicate the presents of container moves in below deck locations, and constraint (A.9) which imposes that no hatch-overstowage is allowed. Finally, constraint (A.10) calculates the makespan at each port.

Appendix B

Appendices of DRL under Uncertainty

B.1 MDP of Master Planning Problem

B.1.1 Sets and Parameters

Provided the sets and parameters in Section 7.2, we introduce additional subsets of the transport set TR given port $p \in P$:

- Onboard transports: $TR_p^{OB} = \{(i,j) \in P^2 \,|\, i \leq p, j > p\}$
- Arrival transports: $TR_p^{AC} = \{(i,j) \in P^2 \,|\, i < p, j > p\}$
- Load transports: $TR_p^{+} = \{(p,j) \in P^2 \,|\, j > p\}$
- Discharge transports: $TR_p^{-} = \{(i,p) \in P^2 \,|\, i < p\}$
- Transports in crane operations: $TR_p^{M} = TR_p^{+} \cup TR_p^{-}$

Considering episode parameters ζ, we define:

- Transports: $tr = (i,j) \in TR$
- Cargo types: $k = (\kappa_1, \kappa_2, \kappa_3) \in K$

For each combination (i,j,k), we associate the expected demand $\mu^{(i,j,k)}$, standard deviation $\sigma^{(i,j,k)}$, TEU per container $teu^{(i,j,k)}$, container weight $w^{(i,j,k)}$, and revenue per container $rev^{(i,j,k)}$.

The TEU per container depends on k as:

$$teu(k) = \begin{cases} 1, & \text{if } \kappa_1 = 20 \text{ ft.} \\ 2, & \text{if } \kappa_1 = 40 \text{ ft.} \end{cases}.$$

Similarly, the container weight is defined by:

$$w(k) = \begin{cases} 1, & \text{if } \kappa_2 = \text{Light} \\ 2, & \text{if } \kappa_2 = \text{Medium} \\ 3, & \text{if } \kappa_2 = \text{Heavy} \end{cases}.$$

Both *teu* and w are broadcasted to shape $n_q = |K| \times |TR|$ in the MDP formulations for consistency in dimensionality.

The revenue function is given by:

$$rev(i,j,k) = \begin{cases} (j-i)(1-LR)+0.1, & \text{if } \kappa_3 = \text{Long} \\ (j-i)+0.1, & \text{if } \kappa_3 = \text{Spot} \end{cases}.$$

The parameters μ and σ are randomly generated, as shown in Appendix B.3.

B.1.2 Formal MDP

We define the MDP by decomposing the traditional CO problem outlined in [220].

In the traditional MPP, cargo is loaded onto a vessel at each port in a voyage. Let $u \in \mathbb{R}^{n_u}$ represent the vessel's utilization over the voyage, which is defined by the set of ports $P = \{1, 2, \ldots, N_P\}$. Let us also recall the following:

$$n_u = |B| \times |D| \times |K| \times |TR|$$

$$n_c = |B| \times |D|, \quad n_q = |K| \times |TR| \quad n_u = \{n_c \times n_q\}$$

Utilization u can be decomposed into individual voyage legs, corresponding to the segments between consecutive ports. Specifically, we decompose as:

$$u = (u_0, u_1, \ldots, u_{N_{P-1}})$$

where each $u_p \in \mathbb{R}^{n_u}$ represents the vessel's utilization immediately after operations at port p.

Feasible Region

Suppose we have a feasible region for the MPP, where:

- $A' \in \mathbb{R}^{m_u \times n_u}$ is the constraint matrix,
- $b' \in \mathbb{R}^{m_u}$ is the bound vector,
- $u_p \in \mathbb{R}^{n_u}_{\geq 0}$ is the nonnegative vessel utilization.

The feasible region, denoted as PH, is given by:

$$PH(s_p) = \{u_p \in \mathbb{R}^{n_u}_{\geq 0} \mid A'u_p \leq b'\}.$$

At port p, utilization can be decomposed into load operations and pre-load utilization:

$$u_p = u'_p + u^+_p \ ,$$

where:

- $u^+_p \in \mathbb{R}^{n_u}_{\geq 0}$ represents load operations,
- $u'_p = u_{p-1} - u^-_p$ is the utilization before load operations,
- $u_{p-1} \in \mathbb{R}^{n_u}_{\geq 0}$ is the previous step's utilization,
- $u^-_p \in \mathbb{R}^{n_u}_{\geq 0}$ is the discharge operations.

Consequently, we can rewrite the feasible region as:

$$PH(s_p) = \{u^+_p \in \mathbb{R}^{n_u}_{\geq 0} \mid A'u^+_p \leq b' - A'u'_p\} \ .$$

B.1.3 Decomposed MDP

Utilization can be decomposed into sequential steps to refine temporal granularity, thereby obtaining an decomposed MDP formulation. We decompose u as:

$$u = (u_0, u_1, \ldots, u_{T_{seq}}) \ ,$$

where $u_t \in \mathbb{R}^{n_u}$ represents the utilization at time step t, and $t \in H = \{1, 2, \ldots, T_{seq}\}$ denotes the episodic horizon H.

Each step t represents a transport and cargo type, as tuple (pol_t, pod_t, k_t). Algorithm 8 illustrates an episode of the decomposed MDP. First, we reset the state s_0, initialize time t, and an empty trajectory. The episode iterates over load ports (pol_t), discharge ports (pod_t), and cargo classes (k_t). At each step, we sample action x_t from policy $\pi_\theta(x|s_t)$ conditioned on state s_t and episode parameters ζ, and transition to state s_{t+1}. Afterwards, we store the results in the trajectory and increment time t. This process continues until all combinations of pol_t, pod_t, and k_t are explored, accumulating a total of T_{seq} steps.

Transitions

We use a stochastic transition function $\mathcal{T}(s_{t+1}|s_t, x_t, \zeta) \in \Delta(S)$. The transition consists of sequential steps:

1. If $t \in T_{\text{new port}}$, port demand is revealed. This means we show $q^{(i,j,k)}_t \ \forall (i,j) \in TR^+_{pol_t}, k \in K$.
2. If $t \in T_{\text{new port}}$, onboard cargo is discharged $u_{t+1} = u_t \odot (1 - \mathbf{e}^-_t)$, where $\mathbf{e}^-_t \in \{0,1\}^{n_q}$ is a binary mask indicating the cargo type and transport to nullify in u_t.
3. Each time t, cargo is loaded onboard $u_{t+1} = u_t + x_t \odot \mathbf{e}^+_t$, where $\mathbf{e}^+_t \in \{0,1\}^{n_q}$ is a binary indicator specifying cargo types and transports to add to u_t.

Algorithm 8 Episode of Augmented MDP

Require: $\mathcal{T}$, π_θ, ζ, $\mathcal{Q}$
$q_{T_{seq}}^{(i,j,k)} \sim \mathcal{Q}(\mu^{(i,j,k)}, \sigma^{(i,j,k)})\ \forall (i,j) \in TR, k \in K$
$s_0 \leftarrow (\mathbf{0}^{n_u}, q_{T_{seq}} \odot \mathbf{e}_0^+)$, $t \leftarrow 0$, Trajectory $\leftarrow \{\}$
for $pol_t = 1$ to $N_P - 1$ **do**
  for $pod_t = pol_t + 1$ to N_P **do**
    for $k_t \in K$ **do**
      $x_t \sim \pi_\theta(x|s_t, \zeta)$
      $s_{t+1} \sim \mathcal{T}(s_t, x_t, \zeta)$
      Append (s_t, x_t, r_t, s_{t+1}) to Trajectory
      $t \leftarrow t + 1$
    end for
  end for
end for
return Trajectory

Additionally, we define the set of time steps before we leave for a new port $p+1$ is defined as follows:

$$T_{\text{leave port}} = \Big\{ t \in H \mid \exists p \in P_1^{N_P - 1} \text{ such that } t = |K| \Big(p(N_P - 1) - \frac{p(p-1)}{2} \Big) - 1 \Big\}.$$

Finally, the set of time steps at which we arrive at a new port p is defined as follows:

$$T_{\text{new port}} = \Big\{ t \in H \mid \exists p \in P_1^{N_P - 1} \text{ such that } t = |K| \Big((p-1)(N_P - 1) - \frac{p(p-1)}{2} \Big) \Big\}.$$

Feasible Region

The state-dependent feasible region for each time t is formulated as:

$$PH(s_t) = \{ u_t \in \mathbb{R}_{\geq 0}^{n_u} \mid A' u_t \leq b' \}$$

Similar to the port utilization, utilization can be decomposed into load operations and pre-load utilization:

$$u_t = u'_t + u_t^+$$

where:

- $u_t^+ \in \mathbb{R}_{\geq 0}^{n_u}$ represents load operations,
- $u'_t = u_{t-1} - u_t^-$ is the utilization before load operations,
- $u_{t-1} \in \mathbb{R}_{\geq 0}^{n_u}$ is the previous step's utilization,
- $u_t^- \in \mathbb{R}_{\geq 0}^{n_u}$ is the discharge operations.

Using the decomposition, we obtain the feasible region as:

$$PH(s_t) = \{u_t \in \mathbb{R}^{n_u}_{\geq 0} \mid A'(u_t^+ + u_t') \leq b'\}$$

Substituting Load Operations for Actions

Actions x_t correspond to transformed load operations u_t^+, given by:

$$x_t = u_t^+ M(s_t), \quad M(s_t) \in \{0,1\}^{n_u \times n_c},$$

where $M(s_t)$ is a state-dependent sparsity mask that selects relevant elements from u_t^+.

However, load operations are subject to m_u constraints, whereas actions adhere to m_c constraints. To bridge this difference, we define the state-dependent constraint matrix:

$$A(s_t) = T(s_t)^\top A' M(s_t), \quad T(s_t) \in \{0,1\}^{m_u \times m_c},$$

where:

- A' is the original constraint matrix of shape (m_u, n_u),
- $T(s_t)$ maps the constraints of u_t^+ to that of x_t
- $M(s_t)$ maps the space of u_t^+ to that of x_t,

Similarly, we introduce a state-dependent bound:

$$b''(s_t) = T(s_t)^\top b',$$

where:

- b' is the original bound of shape $(m_u, 1)$,
- $T(s_t)$ maps the constraints of u_t^+ to that of x_t

Feasible Region for Actions

Using the refined notation, we express the state-dependent feasible region in terms of actions:

$$PH(s_t) = \{x_t \in \mathbb{R}^{n_c}_{\geq 0} \mid A(s_t)x_t \leq b''(s_t) - A'u_t'\}.$$

Next, we define the updated bound as:

$$b(s_t) = b''(s_t) - A'u_t'.$$

Substituting this into the feasible region, we obtain:

$$PH(s_t) = \{x_t \in \mathbb{R}^{n_c}_{\geq 0} \mid A(s_t)x_t \leq b(s_t)\}.$$

B.1.4 MPP Constraints

Let us specify the MPP constraints of $PH(s_p)$ and $PH(s_t)$.

Demand Constraints

Let us consider the demand subset of $PH(s_p)$ as:

$$PH(s_p)_{\text{dem}} = \{x_p \in \mathbb{R}^{n_u}_{\geq 0} \mid A'_{\text{dem}} x_p \leq b'_{\text{dem}} - A'_{\text{dem}} u'_p\} \ .$$

We sum over all vessel locations to obtain an aggregated number of containers of shape n_q. Note that only current load actions x_p are relevant for q_p, hence we can omit $A'_{\text{dem}} u'_p$ as pre-loading utilization has already satisfied its demand requirements.

$$x_p^\top \mathbf{1}_{n_c} \leq q_p$$

Consider the demand subset of $PH(s_t)$ as:

$$PH(s_t)_{\text{dem}} = \{x_t \in \mathbb{R}^{n_c}_{\geq 0} \mid A(s_t)_{\text{dem}} x_t \leq b'(s_t)_{\text{dem}} - A'_{\text{dem}} u'_t\} \ .$$

Right now, we can sum the full vector x_t as it needs to sum to scalar $q_t^{(pol_t, pod_t, k_t)}$. Again, previous steps are irrelevant to demand, hence we can disregard $A'_{\text{dem}} u'_t$ to obtain:

$$\mathbf{1}^\top x_t \leq q_t^{(pol_t, pod_t, k_t)}$$

Capacity Constraints

Let us consider the constraint subset of $PH(s_p)$ as:

$$PH(s_p)_{\text{cap}} = \{x_p \in \mathbb{R}^{n_u}_{\geq 0} \mid A'_{\text{cap}} x_p \leq b'_{\text{cap}} - A'_{\text{cap}} u'_p\} \ .$$

We sum TEU of all cargo types and transports in x_p to obtain TEU use per location with shape n_c. The TEU of pre-load utilization is also considered by subtracting it from the vessel capacity, obtaining the following:

$$x_p teu \leq c - u'_p teu \ ,$$

Consider the demand subset of $PH(s_t)$ as:

$$PH(s_t)_{\text{cap}} = \{x_t \in \mathbb{R}^{n_c}_{\geq 0} \mid A(s_t)_{\text{cap}} x_t \leq b'(s_t)_{\text{cap}} - A'_{\text{cap}} u'_t\} \ .$$

Now, we can do the same trick based on a single scalar $teu^{(pol_t, pod_t, k_t)}$ multiplied with the sum of action x_t.

$$teu^{(pol_t, pod_t, k_t)} \mathbf{1}^\top x_t \leq c - u'_t teu \ ,$$

Stability Constraints

The stability constraints require some algebra to derive for $PH(s_p)$ and $PH(s_t)$.

The lcg constraint in its original form is given by:

$$\frac{\mathbf{1}^\top (lm \odot u_p)}{\mathbf{1}^\top (w \odot u_p)} \leq \overline{lcg} \, .$$

Applying the utilization decomposition, we can obtain the formulation for $PH(s_p)$:

$$\begin{aligned}
\mathbf{1}^\top (lm \odot u_p) &\leq \overline{lcg}\mathbf{1}^\top (w \odot u_p) \\
\mathbf{1}^\top (lm \odot u_p^+) + \mathbf{1}^\top (lm \odot u'_p) &\leq \overline{lcg}\mathbf{1}^\top (w \odot u_p^+) \\
&\quad + \overline{lcg}\mathbf{1}^\top (w \odot u'_p) \\
\mathbf{1}^\top (lm_p \odot x_p) - \overline{lcg}\mathbf{1}^\top (w \odot x_p) &\leq \overline{lcg}\mathbf{1}^\top (w \odot u'_p) \\
&\quad - \mathbf{1}^\top (lm \odot u'_p) \\
\mathbf{1}^\top \big((lm - \overline{lcg}w) \odot x_p\big) &\leq \mathbf{1}^\top \big((\overline{lcg}w - lm) \odot u'_p\big) \, .
\end{aligned}$$

This approach extends to both the lower and upper bounds for the lcg and vcg, ensuring that vessel stability is properly maintained at every step.

Based on the $PH(s_p)$ constraint, we can substitute load operations for actions and obtain the formulation for $PH(s_t)$:

$$\mathbf{1}^\top \big((lm(t) - \overline{lcg}w(t)) \odot x_t\big) \leq \mathbf{1}^\top \big((\overline{lcg}w - lm) \odot u'_p\big) \, .$$

where $w(t) = w^{(pol_t, pod_t, k_t)}$ and $lm(t) = lm^{(pol_t, pod_t, k_t)}$

B.1.5 Auxiliary Variables

The reward function contains two auxiliary variables derived from state s, which incur costs due to inefficient port operations. At port p, Equation (B.1) creates an indicator of hatch movements $hm(s,p) \in \{0,1\}^{|B|}$, whereas Equation (B.2) computes the number of on-deck containers during hatch movements, causing hatch overstowage $ho(s,p) \in \mathbb{R}_{\geq 0}^{|B|}$.

$$hm(s,p) = \left(\sum_{k \in K} \sum_{tr \in TR_p^M} u_t^{(b, d^{below}, k, tr)} > 0 \right) \tag{B.1}$$

$$ho(s,p) = hm(s,p) \left(\sum_{k \in K} \sum_{tr \in TR_p^{AC}} u_t^{(b, d^{above}, k, tr)} \right) \tag{B.2}$$

Equation (B.3) computes the target crane moves at port p by equally spreading the total demand per port over pairs of adjacent bays, where δ^{cm} is the allowed deviation from the equal spread set by ports. Subsequently, Equation (B.4) computes the excess crane moves $cm(s,p) \in \mathbb{R}_{\geq 0}^{|B|-1}$

$$\overline{cm}(s,p) = (1 + \delta^{cm}) \frac{2}{|B|} \sum_{tr \in TR_p^M} \sum_{k \in K} q_t^{(tr,k)} \tag{B.3}$$

$$cm(s,p) = \max\left(\sum_{d\in D}\sum_{k\in K}\sum_{tr\in TR_p^M} u_t^{(0:|B|-1,d,k,tr)} + u_t^{(1:|B|,d,k,tr)} - \overline{cm}(s,p), 0\right) \tag{B.4}$$

B.2 Feasibility Mechanisms

Table B.1 provides an overview of implemented feasibility mechanisms.

TABLE B.1: Feasibility mechanisms and relation to constraints

Type	Implementation	Constraints
FR	Composite loss	Constraints $PH(s_t)$
VP	$VP(x_t, A(s_t), b(s_t), \alpha_v, \delta_v)$	Constraints $PH(s_t)$
WS	$\mathcal{W}(x_t, q_t)$	Demand q_t
PC	$\mathcal{C}(x_t, 0, c - u_t' teu)$	TEU capacity c

B.2.1 Log Probability Adjustments

This subsection provides technical details on the adjustments made to the log-probability distribution of the policy as a result of non-linear transformations to distribution samples.

Projecting actions alters the policy's probability density, necessitating consideration of the change of variables principle [33]. This principle ensures valid volume scaling by requiring the transformation $f(x)$ to satisfy:

1. **Differentiability**: $f(x)$ must be differentiable to compute the Jacobian $J_f(x)$ and determine local volume scaling.
2. **Non-Singularity**: The Jacobian determinant must be non-zero ($\det(J_f(x)) \neq 0$) to prevent dimensional collapse.
3. **Invertibility**: $f(x)$ must be locally or globally invertible to ensure a one-to-one mapping between points in the original and transformed spaces.

These properties ensure the transformation is smooth, one-to-one, and well-behaved, enabling the use of the Jacobian adjustment $\log \pi'(x|s) = \log \pi(x|s) - \log|\det(J_f(x))|$ as a valid probability scaling factor.

Weighted Scaling Projection Layer.

Suppose we have variable $x \in \mathbb{R}^n_{>0}$ and scalar $y \in \mathbb{R}_{>0}$ and the following piecewise linear function:

$$\mathcal{P}(x,y) = \begin{cases} x & \text{if } \mathbf{1}^\top x \leq y \\ \frac{x}{\mathbf{1}^\top x} \cdot y & \text{if } \mathbf{1}^\top x > y \end{cases}$$

Case 1: $\mathbf{1}^\top x \leq y$.

$$\mathcal{P}(x,y) = x$$

$$\frac{\partial}{\partial x}\mathcal{P}(x,y) = \frac{\partial x}{\partial x}$$
$$J_{\mathcal{P}}(x,y) = I_n$$

Case 2: $\mathbf{1}^\top x > y$, where we apply the product rule and then the quotient rule of differentiation.

$$\mathcal{P}(x,y) = \frac{x}{\mathbf{1}^\top x} \cdot y$$
$$\frac{\partial}{\partial x}\mathcal{P}(x,y) = \frac{\partial}{\partial x}\left(\frac{x}{\mathbf{1}^\top x} \cdot y\right)$$
$$\frac{\partial}{\partial x}\mathcal{P}(x,y) = \frac{\partial}{\partial x}\left(\frac{x}{\mathbf{1}^\top x}\right) \cdot y$$
$$J_{\mathcal{P}}(x,y) = y \cdot \frac{1}{(\mathbf{1}^\top x)^2}\left(I_n \mathbf{1}^\top x - x \cdot \mathbf{1}^\top\right)$$
$$J_{\mathcal{P}}(x,y) = \frac{y \cdot I_n}{(\mathbf{1}^\top x)} - \frac{y \cdot x^\top}{(\mathbf{1}^\top x)^2}$$

We obtain the following Jacobian of function $\mathcal{P}(x,y)$:

$$J_{\mathcal{P}}(x,y) = \begin{cases} I_n & \text{if } \mathbf{1}^\top x \leq y \\ \frac{y}{(\mathbf{1}^\top x)^2}\left(I_n \mathbf{1}^\top x - x 1^\top\right) & \text{if } \mathbf{1}^\top x > y \end{cases}$$

Finally, we verify that Jacobian adjustment is allowed for the weighted scaling projection by:

1. $\mathcal{P}(x,y)$ has been shown to be differentiable.
2. Provided that $x, y > 0$, either case is positive definite as the diagonal elements are strictly positive. We obtain $\det(J_{\mathcal{P}}(x,y)) > 0$, thus the Jacobian in non-singular.
3. $\mathcal{P}(x,y)$ is locally invertible as $\det(J_{\mathcal{P}}(x,y)) \neq 0$.

Violation Projection Layer.

Suppose we have function $\mathcal{P}(x,A,b) = x - \eta_v A^\top (Ax-b)_{>0}$ with $x \in \mathbb{R}^n_{>0}$, $\eta_v \in \mathbb{R}_{>0}$, $A \in \mathbb{R}^{m\times n}_{\geq 0}$, $b \in \mathbb{R}^m_{\geq 0}$, and $m > n$.

Case 1: $\eta_v A^\top (Ax-b) = 0$.

$$\mathcal{P}(x,A,b) = x$$
$$\frac{\partial}{\partial x}\mathcal{P}(x,A,b) = \frac{\partial x}{\partial x}$$
$$J_{\mathcal{P}}(x,A,b) = I_n$$

Case 2: $\eta_v A^\top (Ax - b) > 0$, where we apply the chain rule on the second term.

$$\begin{aligned} \mathcal{P}(x, A, b) &= x - \eta_v A^\top (Ax - b) \\ \frac{\partial}{\partial x}\mathcal{P}(x, A, b) &= \frac{\partial}{\partial x}\left(x - \eta_v A^\top (Ax - b)\right) \\ J_{\mathcal{P}}(x, A, b) &= I_n - \eta_v A^\top A \end{aligned}$$

Both cases are combined in the following matrix formulation with diagonal matrix $\text{Diag} = \text{diag}((Ax - b) > 0)$:

$$J_{\mathcal{P}}(x, A, b) = I_n - \eta_v A^\top \text{Diag} A$$

Finally, we can confirm that Jacobian adjustment is allowed for the violation projection layer by:

1. $\mathcal{P}(x, A, b)$ is differentiable as shown above.
2. Due to the full rank nature of the identity I_n and $A^\top \text{Diag} A$ when $\text{Diag} = I_m$, we get $\det(J_{\mathcal{P}}(x, A, b)) \neq 0$, and hence the Jacobian is non-singular.
3. $\mathcal{P}(x, A, b)$ is locally invertible as $\det(J_{\mathcal{P}}(x, A, b)) \neq 0$. It is not globally invertible, due to the piece-wise nature of $(Ax - b)_{>0}$.

Policy Clipping

We can implement a clipped Gaussian distribution that enforces element-wise bounds on a standard Gaussian [72]. Let μ_θ and σ_θ^2 denote the policy's mean and variance, with bounds lb_{pc} and ub_{pc}, and $\Phi(\cdot)$ being the cumulative distribution function of the standard Gaussian. Actions are sampled from $\mathcal{N}(\mu_\theta, \sigma_\theta^2)$ and clipping the result to $[lb_{pc}, ub_{pc}]$. Provided this transformation, we compute the log probabilities $\log \pi(x|s)$ for action x by:

$$\log \pi(x|s) = \begin{cases} \log \Phi\left(\frac{lb_{pc} - \mu_\theta}{\sigma_\theta}\right) & \text{if } x \leq lb_{pc}, \\ -\frac{(x-\mu_\theta)^2}{2\sigma_\theta^2} - \log(\sqrt{2\pi\sigma_\theta^2}) & \text{if } lb_{pc} < x < ub_{pc}, \\ \log\left(1 - \Phi\left(\frac{ub_{pc} - \mu_\theta}{\sigma_\theta}\right)\right) & \text{if } x \geq ub_{pc} \end{cases}$$

B.2.2 Violation Projection Layer

We define a feasible region of action x as the polyhedron:

$$PH = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\},$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Constraints in PH may be violated during optimization. To quantify these violations, we introduce the violation function:

$$\mathcal{V}(x) = (Ax - b)_{>0},$$

where $\mathcal{V}(x)_{m_i} > 0$ indicates that constraint m_i is violated, and $\mathcal{V}(x)_{m_i} = 0$ means the constraint is satisfied.

Violation Gradient Descent. To minimize constraint violations, we update x for a fixed number of iterations using gradient descent on the violation term $\|\mathcal{V}(x)\|_2^2$, which represents the squared distance to feasibility. Differentiating with respect to x, we derive the update rule:

$$\begin{aligned} x' &= x - \eta_v \nabla_x \|\mathcal{V}(x)\|_2^2 \\ &= x - \eta_v 2A^\top \mathcal{V}(x). \end{aligned}$$

Since the step size $\eta_v \in (0, 1)$ is a tunable parameter, we simplify the update function to:

$$x' = x - \eta_v A^\top \mathcal{V}(x).$$

Theorem 1 (Convergence of Violation Gradient Descent)**.** *Let $x_0 \in \mathbb{R}^n$ be an initial point, and consider update:*

$$x_{k+1} = x_k - \eta_v A^\top \mathcal{V}(x_k),$$

where:

- $\mathcal{V}(x) = \max(0, Ax - b)$ *is the element-wise function onto nonnegative constraint values.*
- $\eta_v \in (0, 1)$ *is a step size parameter.*
- $A \in \mathbb{R}^{m \times n}$ *has full row rank.*
- *The feasible region $PH = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$ is nonempty.*

Then, the sequence $\{x_k\}$ satisfies:

1. *The function $g(x) = \|\mathcal{V}(x)\|_2^2$ is non-increasing.*
2. *x_k converges to a feasible point x^* or a local minimum violation point.*

Proof. Define the violation function:

$$g(x) = \|\mathcal{V}(x)\|_2^2.$$

Since $\mathcal{V}(x)$ is an elementwise projection onto nonnegative values, and some function $h(y) = \max(0, y)$ is convex and non-decreasing, then the function $g(x) = \|\mathcal{V}(x)\|_2^2$ is convex when $Ax - b$ is affine.

We apply gradient descent on $g(x)$ using the update rule:

$$x_{k+1} = x_k - \eta_v \nabla_x g(x_k).$$

By the standard descent lemma [36], for a sufficiently small step size η_v, we have:

$$g(x_{k+1}) \leq g(x_k) - \eta_v \|\nabla_x g(x_k)\|_2^2.$$

Since $\eta_v > 0$ and $\|\nabla_x g(x_k)\|_2^2 \geq 0$, it follows that:

$$g(x_{k+1}) \leq g(x_k).$$

Thus, $g(x_k)$ is non-increasing.

Since $g(x_k)$ is also lower-bounded by 0, it must converge to some limit $g^* \geq 0$. This implies:

$$\lim_{k\to\infty} \|\mathcal{V}(x_k)\|_2 = \tilde{c}, \quad \text{for some } \tilde{c} \geq 0.$$

If $\tilde{c} = 0$, then x_k converges to a feasible point, meaning $\mathcal{V}(x_k) = 0$. If $\tilde{c} > 0$, then x_k converges to a local minimum of $g(x)$, where no further descent is possible, satisfying $\nabla_x g(x_k) = 0$, which implies $A^\top \mathcal{V}(x_k) = 0$. □

B.3 Instance Generator

During training, we simulate problem instances based on a Gaussian distribution with element i:

$$q^{(i,j,k)} \sim \mathcal{N}\left(\mu^{(i,j,k)}, \sigma^{(i,j,k)}\right) \; \forall (i,j) \in TR, k \in K.$$

Here, μ is the expected value of cargo demand, initialized by a uniform generator $\mathcal{U}(lb, ub)$, while the standard deviation of demand is defined as $\sigma^{(i,j,k)} = CV \cdot \mu^{(i,j,k)}$, where the coefficient of variation (CV) controls the spread of each element based on $\mu^{(i,j,k)}$. Note that CV is normally defined as $CV^{(i,j,k)} = \frac{\sigma^{(i,j,k)}}{\mu^{(i,j,k)}}$, so we use $\sigma^{(i,j,k)} = CV \cdot \mu^{(i,j,k)}$ to control the spread of the distribution.

We initialize $\mu^{(i,j,k)} \sim \mathcal{U}(0, \overline{\mu^{(i,j,k)}})$, where the upper bound on the expected value is found as follows:

$$\overline{\mu} = \frac{2UR \cdot \mathbf{1}^\top c}{NC} \quad ,$$

where UR is the rate of total utilization present in the demand (e.g., 1.2 means total demand is 120% of total capacity), and $NC \in \mathbb{R}_{>0}^{|TR|}$ is a matrix to spread the demand over different elements proportional to the number of transports remaining to be loaded.

During generalization testing, we simulate problem instances based on a continuous uniform generator:

$$q^{(i,j,k)} \sim \mathcal{U}(lb^{(i,j,k)}, ub^{(i,j,k)}).$$

To ensure similar mean and variance of the instances, we derive parameters lb and ub from the definition of the continuous uniform distribution, as follows:

$$\mu^{(i,j,k)} = (lb^{(i,j,k)} + ub^{(i,j,k)})/2$$
$$(\sigma^{(i,j,k)})^2 = (ub^{(i,j,k)} - lb^{(i,j,k)})^2/12$$

We rewrite to:

$$lb^{(i,j,k)} = \mu^{(i,j,k)} - \sqrt{(12(\sigma^{(i,j,k)})^2)/2}$$
$$ub^{(i,j,k)} = \mu^{(i,j,k)} + \sqrt{(12(\sigma^{(i,j,k)})^2)/2}$$

B.4 Multi-Stage Stochastic MIP

B.4.1 Multi-Stage Scenario Tree

A scenario tree is a directed tree represented as $T_{ST} = (V_{ST}, E_{ST})$, where V_{ST} is the set of nodes, each corresponding to a decision or uncertainty realization at a given stage. $E_{ST} \subseteq V_{ST} \times V_{ST}$ is the set of directed edges representing transitions between nodes over time.

The tree consists of:

1. A root node $v_1 \in V_{ST}$, representing the initial state at the first port.
2. Stages $p = 1, 2, \ldots, N_P - 1$, where each node v belongs to a stage $p(v)$. We denote stages by p, as stages are equivalent to ports in a voyage.
3. Branching structure, where each node has child nodes that correspond to possible future realizations.
4. A probability measure $P_{ST} : V_{ST} \to [0, 1]$ assigning probabilities to nodes, ensuring:
$$\sum_{v' \in \text{child}(v)} \mathbb{P}(v') = \mathbb{P}(v), \quad \forall v \in V_{ST}.$$
5. Scenario paths $\phi \in \mathcal{Z}$, which are root-to-leaf paths representing possible realizations of uncertainty over time.

B.4.2 MIP Formulation

We define the MPP under demand uncertainty as a multi-stage stochastic MIP.

Decision Variables. The following variables are included:

- Vessel utilization: $\tilde{u}_{tr,k}^{b,d,\phi} \in \mathbb{Z}_{\geq 0}$
- Hatch overstowage: $\tilde{ho}_{p,b}^{\phi} \in \mathbb{Z}_{\geq 0}$
- Makespan of cranes: $\tilde{cm}_{p}^{\phi} \in \mathbb{Z}_{\geq 0}$
- Hatch movement: $\tilde{hm}_{p,b}^{\phi} \in \{0, 1\}$

All integer constraints are relaxed linearly in the implementation. Additionally, we use a sufficiently large constant, denoted by big M, to impose logical constraints as needed.

Objective. The objective function (B.5) maximizes the revenue with parameter $rev^{(i,j,k)} \in \mathbb{R}_{>0}$ and minimizes hatch-overstowage with parameter $ct^{ho} \in \mathbb{R}_{>0}$ and crane moves costs with parameter $ct^{cm} \in \mathbb{R}_{>0}$ over scenario paths $\phi \in \mathcal{Z}$ each with probability $\mathbb{P}_\phi$. We assume each scenario path has equal probability.

Constraints. Constraint (B.6) enforces that the onboard utilization cannot exceed the cargo demand q, whereas Constraint (B.7) limits each vessel location to the TEU capacity c for each bay $b \in B$ and deck $d \in D$. In Constraint (B.8), we indicate that hatches need to be opened if below deck cargo needs to be loaded or discharged. Based on these movements, Constraint (B.9) models the amount of hatch overstowage in containers. Subsequently, we compute the target of crane moves $\bar{z}$ in Constraint (B.10), after which Constraint (B.11) computes the excess number of crane moves $\tilde{cm}$.

Additionally, we model the longitudinal and vertical stability in Constraints (B.13) until (B.16). First, we compute the longitudinal moment, vertical moment and total weight in Constraints (B.13), (B.14) and (B.12), respectively. Second, Constraint (B.15) bounds lcg between $\underline{lcg}$ and $\overline{lcg}$. Third, Constraint (B.16) bounds vcg between $\underline{vcg}$ and $\overline{vcg}$. Both lcg and $\overline{vcg}$ are linearized equivalents of the original Constraints (7.1) and (7.2), respectively. Furthermore, we include non-anticipation in Constraint (B.17) to prevent leveraging future demand realizations.

$$\max \sum_{\phi \in \mathcal{Z}} \mathbb{P}_\phi \sum_{p \in P} \sum_{b \in B} \sum_{d \in D} \sum_{k \in K} \sum_{tr \in TR^+(p)} rev^{(i,j,k)} \tilde{u}^{b,d,\phi}_{tr,k} - ct^{ho} \tilde{ho}^{\phi}_{p,b} - ct^{cm} \tilde{cm}^{\phi}_{p} \tag{B.5}$$

$$\text{s.t.} \sum_{b \in B} \sum_{d \in D} \tilde{u}^{b,d,\phi}_{tr,k} \leq q^{\phi}_{tr,k} \quad \forall p \in P,\ tr \in TR^{OB}(p),\ k \in K,\ \phi \in \mathcal{Z} \tag{B.6}$$

$$\sum_{k \in K} \sum_{tr \in TR^{OB}(p)} teu_{tr,k} \tilde{u}^{b,d,\phi}_{tr,k} \leq c_{b,d} \quad \forall p \in P,\ b \in B,\ d \in D,\ \phi \in \mathcal{Z} \tag{B.7}$$

$$\sum_{k \in K} \sum_{tr \in TR^{M}(p)} \tilde{u}^{b,d_h,\phi}_{tr,k} \leq M \tilde{hm}^{\phi}_{p,b} \quad \forall p \in P,\ b \in B,\ \phi \in \mathcal{Z} \tag{B.8}$$

$$\sum_{k \in K} \sum_{tr \in TR^{AC}(p)} \tilde{u}^{b,d_o,\phi}_{tr,k} - M(1 - \tilde{hm}^{\phi}_{p,b}) \leq \tilde{ho}^{\phi}_{p,b} \quad \forall p \in P,\ b \in B,\ \phi \in \mathcal{Z} \tag{B.9}$$

$$\bar{z}^{\phi}_{p} = (1 + \delta^{cm}) \frac{2}{|B|} \sum_{tr \in TR^{M}(p)} \sum_{k \in K} q^{\phi}_{tr,k} \quad \forall p \in P,\ \phi \in \mathcal{Z} \tag{B.10}$$

$$\sum_{b \in b'} \sum_{d \in D} \sum_{k \in K} \sum_{tr \in TR^{M}(p)} \tilde{u}^{b,d,\phi}_{tr,k} - \bar{z}^{\phi}_{p} \leq \tilde{cm}^{\phi}_{p} \quad \forall p \in P,\ b' \in B',\ \phi \in \mathcal{Z} \tag{B.11}$$

$$tw^{\phi}_{p} = \sum_{k \in K} w_k \sum_{tr \in TR^{OB}(p)} \sum_{d \in D} \sum_{b \in B} \tilde{u}^{b,d,\phi}_{tr,k}$$

$$\forall p \in P,\ \phi \in \mathcal{Z} \tag{B.12}$$

$$lm_p^{\phi} = \sum_{b \in B} ld_b \sum_{k \in K} w_k \sum_{tr \in TR^{OB}(p)} \sum_{d \in D} \tilde{u}_{tr,k}^{b,d,\phi}$$

$$\forall p \in P,\ \phi \in \mathcal{Z} \tag{B.13}$$

$$vm_p^{\phi} = \sum_{d \in D} vd_d \sum_{k \in K} w_k \sum_{tr \in TR^{OB}(p)} \sum_{b \in B} \tilde{u}_{tr,k}^{b,d,\phi}$$

$$\forall p \in P,\ \phi \in \mathcal{Z} \tag{B.14}$$

$$\underline{lcg}tw_p^{\phi} \leq lm_p^{\phi} \leq \overline{lcg}tw_p^{\phi}$$

$$\forall p \in P,\ \phi \in \mathcal{Z} \tag{B.15}$$

$$\underline{vcg}tw_p^{\phi} \leq vm_p^{\phi} \leq \overline{vcg}tw_p^{\phi}$$

$$\forall p \in P,\ \phi \in \mathcal{Z} \tag{B.16}$$

$$\tilde{u}_{tr,k}^{b,d,\phi'} = \tilde{u}_{tr,k}^{b,d,\phi}$$

$$\forall p \in P, tr \in TR^{+}(p), k \in K,$$

$$b \in B, d \in D, \phi, \phi' \in \mathcal{Z} \mid q_{[p-1]}^{\phi} = q_{[p-1]}^{\phi'} \tag{B.17}$$

B.5 Deep RL Implementation Details

B.5.1 PPO Algorithm

PPO is an on-policy reinforcement learning algorithm that seeks to maximize expected cumulative reward while enforcing stable policy updates via clipped importance sampling [188], as outlined in Algorithm 9. The agent collects trajectories, computing n_{ppo}-step return to evaluate performance with $V_\theta(s)$ as estimated state value:

$$G_t^{(n_{\text{ppo}})} = \sum_{k_{\text{ppo}}=0}^{n_{\text{ppo}}-1} \gamma^k r_{t+k_{\text{ppo}}} + \gamma^{n_{\text{ppo}}} V_\theta(s_{t+n_{\text{ppo}}}), \tag{B.18}$$

To reduce variance, we adopt Generalized Advantage Estimation (GAE) [187]:

$$\hat{A}_t^{\text{GAE}} = \sum_{l_{\text{ppo}}=0}^{\infty} (\gamma\lambda)^{l_{\text{ppo}}} \delta_{t+l_{\text{ppo}}}, \tag{B.19}$$

$$\delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t). \tag{B.20}$$

Here, δ_t is the temporal difference (TD) residual, which quantifies the advantage of taking action x_t at state s_t.

The actor is updated using the PPO clipped surrogate loss:

$$\mathcal{L}_{\text{actor}}(\theta) = \mathbb{E}_t\Big[\min\big(\text{ratio}_t(\theta)\hat{A}_t^{\text{GAE}},$$

$$\text{clip}(\text{ratio}_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\text{GAE}}\big)\Big], \tag{B.21}$$

where the probability ratio is defined as:

$$\text{ratio}_t(\theta) = \frac{\pi_\theta(x_t|s_t)}{\pi_{\theta_{\text{old}}}(x_t|s_t)}. \tag{B.22}$$

The critic aims to minimize the squared TD error:

$$\mathcal{L}_{\text{critic}}(\theta) = \mathbb{E}_t\Big[\big(V_\theta(s_t) - G_t^{(n_{\text{ppo}})}\big)^2\Big]. \tag{B.23}$$

Finally, the total PPO objective, including feasibility regularization from Equation (7.11), is given by:

$$\begin{aligned}\mathcal{L}(\theta) = \mathcal{L}_{\text{actor}}(\theta) + \lambda_c \mathcal{L}_{\text{critic}}(\theta) + \lambda_f \mathcal{L}_{\text{feas}}(\theta) \\ - \lambda_e \mathbb{E}_t\big[\text{entropy}(\pi_\theta)\big],\end{aligned} \tag{B.24}$$

where λ_c, λ_f, and λ_e are weighting coefficients for the critic loss, feasibility regularization, and entropy regularization respectively.

Algorithm 9 Proximal Policy Optimization (PPO)

Require: Model parameters θ, steps n, learning rate η
for each gradient update **do**
  for each step t **do**
    Collect n-step trajectories $\{(s_t, x_t, r_t, s_{t+1})\}$
    Compute n-step returns $G_t^{(n_{\text{ppo}})}$
    Compute advantage estimates $\hat{A}_t^{\text{GAE}}$
  end for
  Update parameters: $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}(\theta)$
end for
return Policy π_θ

B.5.2 SAC Algorithm

Soft Actor-Critic (SAC) is an off-policy reinforcement learning algorithm that optimizes both reward maximization and entropy to encourage efficient exploration [78], as outlined in Algorithm 10. It is based on maximum entropy reinforcement learning, which aims to learn a stochastic policy that not only maximizes cumulative rewards but also maintains high entropy for robustness and stability. SAC leverages a soft Q-learning approach, using two Q-functions to mitigate overestimation bias, an entropy-regularized policy update, and an automatically adjusted temperature parameter to balance exploration and exploitation.

The algorithm maintains an actor network for policy learning, two Q-function critics for value estimation, a target Q-network for stable learning, and an adaptive temperature parameter to regulate entropy. The loss functions for standard SAC are derived from the Bellman backup equation and the policy gradient formulation, ensuring convergence to an optimal stochastic policy. We also include feasibility regularization from Equation (7.11) in the actor loss.

- Compute target Q-value:

$$\begin{aligned}Q_{\text{target}}(s_t, x_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, x_{t+1} \sim \pi}\Big[\\ \min_{l=1,2} Q_\theta^l(s_{t+1}, x_{t+1}) - \alpha \log \pi_\theta(x_{t+1}|s_{t+1})\Big]\end{aligned}$$

- Critic loss:
$$\mathcal{L}_{\text{critic}}(\theta) = \mathbb{E}\Big[(Q_\theta(s_t, x_t) - Q_{\text{target}}(s_t, x_t))^2\Big]$$

- Actor loss:
$$\mathcal{L}_{\text{actor}}(\theta) = \mathbb{E}\Big[\alpha \log \pi_\theta(x_t|s_t) - Q_\theta(s_t, x_t) + \lambda_f \mathcal{L}_{\text{feas}}(\theta)\Big]$$

- Temperature loss:
$$\mathcal{L}_\alpha(\theta) = \mathbb{E}\Big[- \alpha(\log \pi_\theta(x_t|s_t) + \text{entropy}_{\text{target}})\Big]$$

This formulation ensures stability and encourages exploration by adapting the trade-off between exploitation and exploration dynamically.

Algorithm 10 Soft Actor-Critic (SAC)

Require: Parameters: actor θ_{actor}, critics $\theta^1_{\text{critic}}, \theta^2_{\text{critic}}$, targets $(\theta^1_{\text{target}}, \theta^2_{\text{target}}) = (\theta^1_{\text{critic}}, \theta^2_{\text{critic}})$, temperature α, learning rate actor η_a, learning rate critic η_c, learning rate temperature η_α, soft update parameter τ, replay buffer $\mathcal{D}$.
for each iteration **do**
  for each environment step t **do**
    Sample action $x_t \sim \pi_\theta(x_t|s_t)$
    Perform transition $s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, x_t)$
    Observe reward $r_t = \mathcal{R}(s_t, x_t)$,
    Store (s_t, x_t, r_t, s_{t+1}) in $\mathcal{D}$.
  end for
  for each gradient step **do**
    Sample a minibatch (s_t, x_t, r_t, s_{t+1}) from $\mathcal{D}$.
    Compute target Q-value: $Q_{\text{target}}(s_t, x_t)$
    Update parameters:
  $\theta^l_{\text{critic}} \leftarrow \theta^l_{\text{critic}} - \eta_c \nabla_l \mathcal{L}_{\text{critic}}(\theta)$ for $l \in \{1, 2\}$
  $\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} - \eta_a \nabla \mathcal{L}_{\text{actor}}(\theta)$
  $\alpha \leftarrow \alpha - \eta_\alpha \nabla \mathcal{L}_\alpha(\theta)$
  $\theta^l_{\text{target}} \leftarrow \tau\theta^l_{\text{critic}} + (1 - \tau)\theta^l_{\text{target}}$ for $l \in \{1, 2\}$
  end for
end for
return Policy π_θ

B.5.3 Hyperparameters

The parameters of the MPP environment are shown in Table B.2.

Table B.3 provides the hyperparameters of projected and vanilla PPO and SAC.

B.5.4 Additional Experiments

In Table B.4, we analyze different configurations of feasibility regularization (FR). First, we evaluate performance under the same hyperparameters as AM-P policies. Second, we examine the effect of significantly increasing λ_f. Third, we assess performance with specific hyperparameter tuning, including λ_f. These experiments

TABLE B.2: Environment parameters

Parameters	**Symbol**	**Value**
Voyage length	N_P	4
Number of bays	N_B	10
Cardinality deck set	$\|D\|$	2
Cardinality cargo set	$\|K\|$	12
Cardinality transport set	$\|TR\|$	6
Vessel TEU	$\mathbf{1}^\top c$	1,000
Long term contract reduction	LR	0.3
Utilization rate demand	UR	1.1
lcg bounds	$(\underline{lcg}, \overline{lcg})$	(0.85,1.05)
vcg bounds	$(\underline{vcg}, \overline{vcg})$	(0.95,1.15)
Crane moves allowance	$\delta^{\overline{cm}}$	0.25
Overstowage costs	ct^{ho}	0.33
Crane move costs	ct^{ho}	0.5

indicate that FR can reduce the distance to the feasible region, however, achieving fully feasible instances remains a challenge.

TABLE B.3: Hyperparameters for projected and vanilla PPO and SAC

Settings		**Projection Algorithms**		**Vanilla Algorithms**	
Hyperparameters	**Symbol**	**PPO**	**SAC**	**PPO**	**SAC**
Actor Network		Attention	Attention	Attention	Attention
Number of Heads		8	8	4	4
Hidden Layer Size		128	128	256	256
Encoder Layers		3	3	2	1
Decoder Layers		3	3	3	3
Critic Network		$1 \times$ MLP	$2 \times$ MLP	$1 \times$ MLP	$2 \times$ MLP
Critic Layers		4	4	4	4
Target Network		No	Soft Update	No	Soft Update
Target Update Rate	τ	N/A	0.005	N/A	0.005
Dropout Rate		0.009	0.009	0.073	0.164
Max Policy Std.		1.931	1.931	1.118	1.779
Optimizer		Adam	Adam	Adam	Adam
Learning Rate	η	2.04×10^{-4}	2.04×10^{-4}	9.64×10^{-4}	9.10×10^{-4}
Batch Size		64	64	64	64
Embedding Size		128	128	128	128
Discount Factor	γ	0.99	0.99	0.99	0.99
GAE	λ	0.95	N/A	0.95	N/A
Value Coefficient	λ_c	0.50	N/A	0.50	N/A
Entropy Coefficient	λ_e	0.010	Learned	0.061	Learned
Feasibility Penalty	λ_f	0.0677	0.0677	0.302	0.065
Clip Parameter	ϵ	0.2	N/A	0.2	N/A
Replay Buffer		No	Yes	No	Yes
Replay Buffer Size		N/A	10^4	N/A	10^4
Mini-batch Size		16	16	32	16
Update Epochs		5	1	1	1
Entropy Target		N/A	$-\lvert X \rvert$	N/A	$-\lvert X \rvert$
Projection Learning Rate	η_v	0.05	0.05	N/A	N/A
Projection Epochs		100	100	N/A	N/A
Inference Projection Stop	δ_v	0.05	0.05	N/A	N/A
Training Budget		7.2×10^7	7.2×10^7	7.2×10^7	7.2×10^7
Validation Budget		5.0×10^3	5.0×10^3	5.0×10^3	5.0×10^3
Validation Frequency		Every 20%	Every 20%	Every 20%	Every 20%

TABLE B.4: Performance evaluation on N instances of feasibility regularization (FR) with hyperparameter (H.P.) settings: projected hyperparameters (Proj.), ensuring a fair comparison with projection-based policies, and tuned hyperparameter (Tune), optimized specifically for PPO and SAC with FR. While we use λ_f as the control parameter for FR, tuning involves adjusting a Lagrangian multiplier for each constraint. Average performance metrics include objective value in profit (Ob.), inference time in seconds (Time), percentage of feasible instances (F.I.), and total absolute distance to the feasible region $d(PH(s_t))$. Note that † indicates infeasible objectives.

Methods					**Testing ($N = 30$)**			
Alg.	**Model**	**F.M.**	**H.P.**	λ_f	**Ob. ($)**	**Time (s)**	**F.I. (%)**	$d(\boldsymbol{PH}(s_t))$
SAC	AM	FR	Proj.	0.0677	$1139.78^{\dagger}$	13.82	0.00	62.77
SAC	AM	FR	Proj.	0.677	$1031.55^{\dagger}$	13.45	0.00	120.74
SAC	AM	FR	Tune	0.065	$1113.03^{\dagger}$	12.63	0.00	37.55
PPO	AM	FR	Proj.	0.0677	$2606.87^{\dagger}$	13.36	0.00	3171.86
PPO	AM	FR	Proj.	0.677	$2593.29^{\dagger}$	13.20	0.00	3381.59
PPO	AM	FR	Tune	0.302	$1842.46^{\dagger}$	11.74	0.00	754.21

Appendix C

Appendices of DRL under Uncertainty at Scale

C.1 MPP Parameters

The parameters of the MPP are shown in Table C.1.

TABLE C.1: MPP parameters

Parameters	Symbol	Values
Voyage lengths	N_P	{4,5,6}
Number of bays	N_B	20
Cardinality deck set	$\|D\|$	2
Cardinality block set	$\|BL\|$	2
Cardinality cargo set	$\|K\|$	12
Cardinality transport set	$\|TR\|$	6
Vessel TEU	$\mathbf{1}^\top c$	20,000
Long term contract reduction	LR	0.3
Utilization rate demand	UR	1.1
lcg bounds	$(\underline{lcg}, \overline{lcg})$	(0.85,1.05)
vcg bounds	$(\underline{vcg}, \overline{vcg})$	(0.95,1.15)
Crane moves allowance	$\delta^{\overline{cm}}$	0.25
Overstowage costs	ct^{ho}	0.33
Crane move costs	ct^{ho}	0.5

C.2 Instance Generator

This Appendix introduces the instance generator used to sample demand, after which a descriptive analysis visualizes the demand distribution.

Algorithm 11 generates transport matrices of cargo demand based on a perturbed uniform distribution, designed to simulate stowage planning problem instances. Given vessel capacity c and perturbation factor ρ, an upper bound matrix ub is first computed proportionally to the total vessel capacity. To introduce controlled randomness, a perturbation is applied element-wise using samples $U_{i,j,k} \sim \mathcal{U}(0,1)$, scaling each upper bound within the interval $[1-\rho, 1+\rho]$. This yields the perturbed bounds $\tilde{ub}$, which represent the maximum admissible demand per transport (i,j) and cargo type k. Demand samples $q_{i,j,k}$ are then drawn from a uniform distribution over the interval $[1, \tilde{ub}_{i,j,k}]$, where the outcome is continuous or discrete depending

on whether the model assumes real-valued or integer-valued demand variables. The expected value μ and standard deviation σ of the uniform distribution are derived analytically from $\tilde{ub}$, reflecting the first and second moments of the demand distribution. The resulting tuple (q, μ, σ) provides necessary information on the demand distribution.

Algorithm 11 Generate Cargo Demand with Uniform Distribution

Input: Capacity c, perturbation ρ
Output: Realized, expected and standard dev. demand (q, μ, σ)

Compute upper bound matrix $ub \propto UR \cdot \mathbf{1}^\top c$
Apply random perturbation with $U_{i,j,k} \sim \mathcal{U}(0,1)$:

$$\tilde{ub}_{i,j,k} = ub_{i,j,k} \times \left(1 + (2 \times U_{i,j} - 1) \times \rho\right) \forall (i,j) \in TR, k \in K,$$

Generate uniform sample:

$$q_{i,j,k} \sim \begin{cases} \mathcal{U}(1, \tilde{ub}_{i,j,k}) & \text{if } q \in \mathbb{R}_{>0}^{n_q} \\ \mathcal{U}_{\mathbb{Z}}(1, \tilde{ub}_{i,j,k}) & \text{if } q \in \mathbb{Z}_{>0}^{n_q} \end{cases} \quad \forall (i,j) \in TR,\ k \in K$$

Compute expected value: $\mu = \frac{1}{2}\tilde{ub}$
Compute standard deviation: $\sigma = \frac{\tilde{ub}}{\sqrt{12}}$

return q, μ, σ

C.3 Implementation Details of AI2STOW

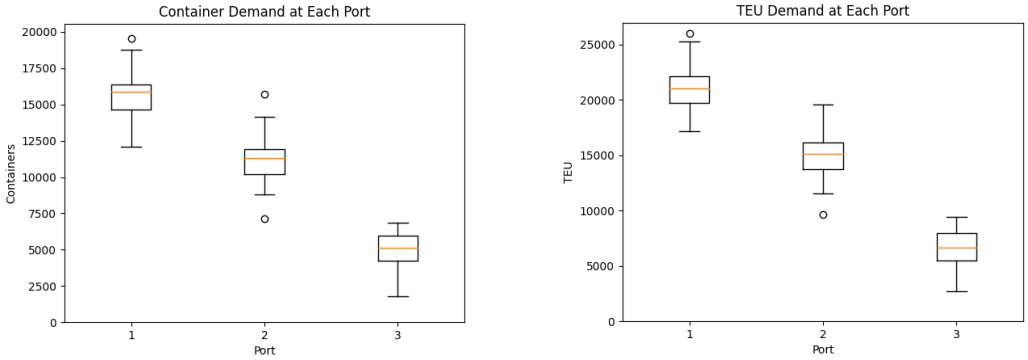
In this appendix, we first define the SAC algorithm, then discuss implemented projection layers, and finally describe relevant hyperparameters. Parts of this appendix are adapted from our previous work [217].

C.3.1 SAC Algorithm

Soft Actor-Critic (SAC) is an off-policy reinforcement learning algorithm that optimizes both reward maximization and entropy to encourage efficient exploration [78], as outlined in Algorithm 12. It is based on maximum entropy reinforcement learning, which aims to learn a stochastic policy that not only maximizes cumulative rewards but also maintains high entropy for robustness and stability. SAC leverages a soft Q-learning approach, using two Q-functions to mitigate overestimation bias, an entropy-regularized policy update, and an automatically adjusted temperature parameter to balance exploration and exploitation.

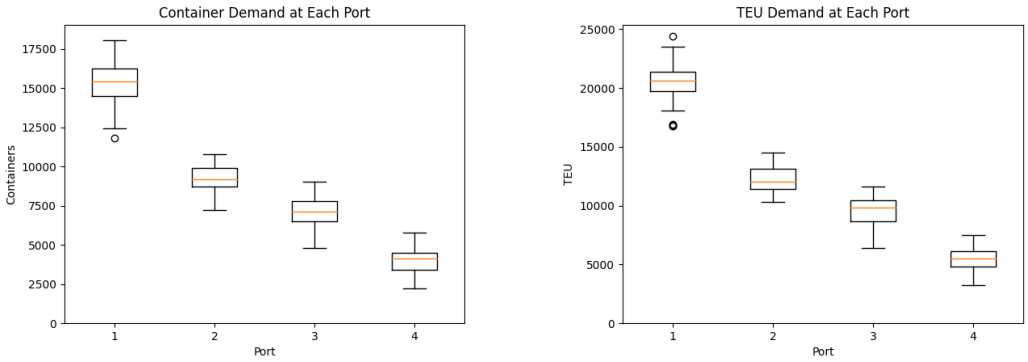
The algorithm maintains an actor network for policy learning, two Q-function critics for value estimation, a target Q-network for stable learning, and an adaptive temperature parameter to regulate entropy. The loss functions for standard SAC are derived from the Bellman backup equation and the policy gradient formulation, ensuring convergence to an optimal stochastic policy. We also include feasibility regularization from Equation (8.32) in the actor loss.

- Compute target Q-value:



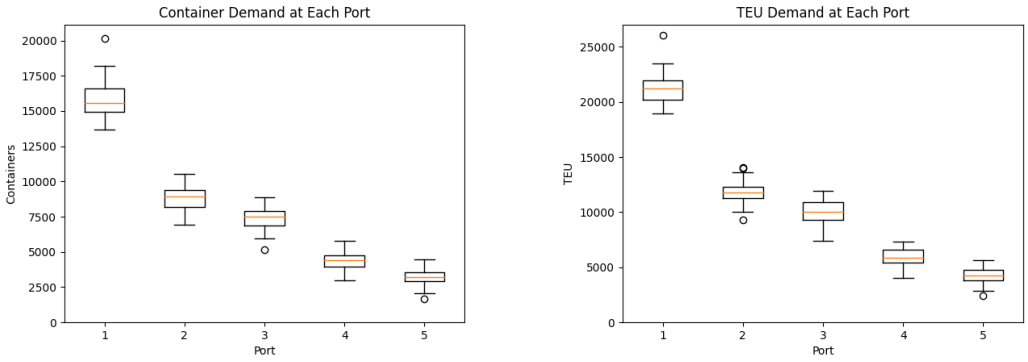
(A) Container demand distribution (B) TEU demand distribution

FIGURE C.1: Simulated demand for $N_P = 4$ by instance generator



(A) Container demand distribution (B) TEU demand distribution

FIGURE C.2: Simulated demand for $N_P = 5$ by instance generator



(A) Container demand distribution (B) TEU demand distribution

FIGURE C.3: Simulated demand for $N_P = 6$ by instance generator

$$Q_{\text{target}}(s_t, x_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, x_{t+1} \sim \pi} \Big[\min_{l=1,2} Q^l_\theta(s_{t+1}, x_{t+1}) - \alpha \log \pi_\theta(x_{t+1}|s_{t+1}) \Big]$$

- Critic loss:

$$\mathcal{L}_{\text{critic}}(\theta) = \mathbb{E}\Big[(Q_\theta(s_t, x_t) - Q_{\text{target}}(s_t, x_t))^2 \Big]$$

- Actor loss:

$$\mathcal{L}_{\text{actor}}(\theta) = \mathbb{E}\Big[\alpha \log \pi_\theta(x_t|s_t) - Q_\theta(s_t, x_t) + \lambda_f \mathcal{L}_{\text{feas}}(\theta) \Big]$$

- Temperature loss:

$$\mathcal{L}_\alpha(\theta) = \mathbb{E}\Big[-\alpha(\log \pi_\theta(x_t|s_t) + \text{entropy}_{\text{target}}) \Big]$$

This formulation ensures stability and encourages exploration by dynamically adapting the trade-off between exploitation and exploration.

C.3.2 Projection Layers

This subsection introduces the projection layers used in the article.

Violation Projection

In previous work [217], we discussed the violation projection on a convex polyhedron of constraints. For full technical details, we refer to [217].

Algorithm 13 defines a violation projection (VP) layer that reduces inequality constraint violations by shifting a point x closer to the feasible region of the convex polyhedron $PH = \{x \in \mathbb{R}^n_{>0} : Ax \leq b\}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The inequality constraint $Ax \leq b$ ensures that PH is convex, enabling gradient-based reduction of violation $\mathcal{V}(x)$ [36].

To measure feasibility, we compute the element-wise violation $\mathcal{V}(x) = (Ax - b)_{>0}$, where $\mathcal{V}(x)_{m_i} > 0$ indicates that constraint m_i is violated, and $\mathcal{V}(x)_{m_i} = 0$ indicates satisfaction. To minimize violations, we iteratively update x using gradient descent on the squared violation norm $\|\mathcal{V}(x)\|_2^2$, which approximates the distance to the feasible region PH. The update step is given by:

$$x' = x - \eta_v \nabla_x \|\mathcal{V}(x)\|_2^2 \tag{C.1}$$

This is equivalent to the update rule shown in Algorithm 13.

During training, the VP layer executes for a fixed number of `epochs`. However, during inference, a stopping criterion is introduced: the projection halts when the change in total violation, $\mathbf{1}^\top \mathcal{V}(x') - \mathbf{1}^\top \mathcal{V}(x)$, falls below a threshold δ_v. As a result, the VP layer reduces the distance of x to the feasible region, incorporating constraint awareness into otherwise unconstrained policies.

Algorithm 12 Soft Actor-Critic (SAC)

Require: Actor parameters θ_{actor}, critic parameters θ^1_{critic}, θ^2_{critic}
Initialize target critics: $\theta^1_{\text{target}} \leftarrow \theta^1_{\text{critic}}$, $\theta^2_{\text{target}} \leftarrow \theta^2_{\text{critic}}$
Initialize temperature parameter α
Set hyperparameters: learning rates η_a, η_c, η_α; soft update rate τ_{su}; replay buffer $\mathcal{D}$
for each training iteration **do**
    for each environment step **do**
        Sample action $x_t \sim \pi_{\theta_{\text{actor}}}(x_t \mid s_t)$
        Observe next state $s_{t+1} \sim \mathcal{T}(s_{t+1} \mid s_t, x_t)$
        Observe reward $r_t = \mathcal{R}(s_t, x_t)$
        Store transition (s_t, x_t, r_t, s_{t+1}) in buffer $\mathcal{D}$
    end for
    for each gradient update step **do**
        Sample minibatch (s_t, x_t, r_t, s_{t+1}) from buffer $\mathcal{D}$
        Compute target Q-value:
            Sample $x_{t+1} \sim \pi_{\theta_{\text{actor}}}(x \mid s_{t+1})$
            $Q_{\text{target}} \leftarrow r_t + \gamma \cdot \min_{l=1,2} Q_{\theta^l_{\text{target}}}(s_{t+1}, x_{t+1}) - \alpha \log \pi_{\theta_{\text{actor}}}(x_{t+1} \mid s_{t+1})$
        for $l = 1$ to 2 **do**
            Update critic θ^l_{critic} using:
                $\theta^l_{\text{critic}} \leftarrow \theta^l_{\text{critic}} - \eta_c \nabla_{\theta^l_{\text{critic}}} \mathcal{L}_{\text{critic}}$
        end for
        Update actor parameters:
            $\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} - \eta_a \nabla_{\theta_{\text{actor}}} \mathcal{L}_{\text{actor}}$
        Update temperature:
            $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \mathcal{L}_\alpha$
        for $l = 1$ to 2 **do**
            Soft update target critic:
                $\theta^l_{\text{target}} \leftarrow \tau_{su} \cdot \theta^l_{\text{critic}} + (1 - \tau_{su}) \cdot \theta^l_{\text{target}}$
        end for
    end for
end for
return Policy $\pi_{\theta_{\text{actor}}}$

Algorithm 13 Violation Projection Layer

Require: Input vector $x \in \mathbb{R}^n_{>0}$, parameters (A, b, η_v, δ_v)
Ensure: Projected vector x'
Initialize $x' \leftarrow x$
Define violation function: $\mathcal{V}(x) \leftarrow \max(Ax - b, 0)$ (element-wise)
for each iteration $i = 1$ to `epochs` **do**
    Set $x \leftarrow x'$
    Update $x' \leftarrow x - \eta_v A^\top \mathcal{V}(x)$
    if $\mathbf{1}^\top \mathcal{V}(x') - \mathbf{1}^\top \mathcal{V}(x) \leq \delta_v$ **then**
        break
    end if
end for
return x'

Policy Clipping

In our previous work [217], we also introduced policy clipping (PC) to enforce TEU capacity constraints. Policy clipping applies element-wise lower and upper bounds to a vector x, ensuring it remains within a specified box-constrained region. This is implemented using the function:

$$\mathcal{C}(x, lb_{pc}, ub_{pc}) = \max\big(\min(x, ub_{pc}), lb_{pc}\big)$$

where lb_{pc} and ub_{pc} represent the element-wise lower and upper bounds, respectively. While PC is simple and efficient, it is only applicable to box constraints. For more complex convex constraint structures, we rely on the VP layer. Full implementation details are provided in [217].

Convex Program Layer

Given a convex polyhedron $PH = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, we incorporate a differentiable convex optimization layer following [1], where the solution to a parameterized convex problem is treated as the output of a network layer. Specifically, we solve

$$x^\star(\theta) = \arg\min_{x \in PH} f(x, \theta),$$

where θ denotes parameters produced by upstream layers and $f(x, \theta)$ is a convex objective. Gradients $\partial x^\star / \partial \theta$ are computed via implicit differentiation of the KKT conditions, enabling end-to-end training. However, since the mapping $\theta \mapsto x^\star(\theta)$ is defined implicitly via the solution of an optimization problem, the Jacobian determinant $\det(\partial x^\star / \partial \theta)$ is not tractable. As a result, the layer cannot be used to compute log-density corrections via the change-of-variables formula, limiting its applicability in likelihood-based or flow-based generative models. use in likelihood-based models.

To promote numerical stability, we incorporate slack variables into the stability constraint and apply a large penalty (e.g., 10^4) for violations. Specifically, we relax constraints of the form $g(x) \leq 0$ to $g(x) \leq \varepsilon$, and penalize ε in the objective via $\lambda_{cp}\|\varepsilon\|_1$. This soft enforcement accommodates intermediate actions that may temporarily breach stability constraints, as can arise in the MDP.

C.3.3 Hyperparameters

TABLE C.2: Hyperparameters of AI2STOW. Note that the hyperparameters are identical for DRL-FR, except for λ_f as we performed parameter tuning to set it to $\lambda_f = 0.195$

Hyperparameters	Symbol	AI2STOW
Actor Network		Attention
Number of Heads		8
Hidden Layer Size		512
Encoder Layers		3
Decoder Layers		4
Critic Network		$2 \times$ MLP
Critic Layers		5
Target Network		Soft Update
Target Update Rate	τ_{su}	0.005
Dropout Rate		0.009
Max Policy Std.		9.460
Optimizer		Adam
Learning Rate	η	1.46×10^{-4}
Batch Size		64
Embedding Size		128
Discount Factor	γ	0.99
Entropy Coefficient	λ_e	Learned
Feasibility Penalty	λ_f	0.283
Replay Buffer Size		10^4
Mini-batch Size		32
Entropy Target		$-\lvert X\rvert$
Projection Learning Rate	η_v	0.010
Projection Epochs		273
Inference Projection Stop	δ_v	0.024
Finetuned Projection Learning Rate	$\eta_v{}^\star$	0.01
Finetuned Projection Epochs		300
Finetuned Inference Projection Stop	$\delta_v{}^\star$	0.01
Slack penalty of CP	λ_{cp}	1×10^4
Training Budget		7.2×10^6
Validation Budget		5.0×10^3
Validation Frequency		Every 20%
Inference Rollouts		5

C.4 Additional Experiments

Table C.3 illustrates the rapid growth in computational demands of the scenario tree in the SMIP-NA model with $S_{ST} = 20$. As the voyage length N_P increases, the number of scenario paths $|\mathcal{Z}|$, as well as the runtime and memory usage, grow substantially. For $N_P = 4$, both runtime and memory consumption are measured directly and serve as a baseline. Assuming an optimistic case of linear scaling with respect to $|\mathcal{Z}|$, the extrapolated values for $N_P = 5$ and $N_P = 6$ already exceed practical hardware and runtime limits by a significant margin. This highlights the challenge of solving large-scale instances of SMIP-NA with conventional computational resources.

TABLE C.3: Evaluation of runtime and peak memory usage for the SMIP-NA model with $S_{ST} = 20$, as the scenario tree expands with increasing voyage length N_P. Runtime and memory usage for $N_P = 5$ and $N_P = 6$ are extrapolated linearly based on the growth of $|\mathcal{Z}|$ relative to $N_P = 4$, and are marked with an asterisk ($*$) to indicate estimated values.

SMIP-NA	**Metrics**		
N_P	$\lvert\mathcal{Z}\rvert$	**Runtime (s)**	**Memory (GB)**
4	400	1576.77	36.01
5	8,000	31,535.40*	720.23*
6	160,000	630,708.00*	14,404.66*