University of Nottingham
UK | CHINA | MALAYSIA

Deep Reinforcement Learning for Truck Task Dispatching Optimization Problem in a Real-life Marine Container Terminal

Thesis submitted to the University of Nottingham for the degree of **Doctor of Philosophy, Dec 2024.**

Jiahuan Jin

20219257

Supervised by

Ruibin Bai
Rong Qu

Signature _____

Date _____ / _____ / _____

Abstract

In recent decades, dynamic real-time task dispatching has emerged as an essential area of study within the context of modern logistics and global supply chain development. Challenges associated with response timeliness, uncertainties, and solution generalization of this problem have gradually emerged among various real-life cases. This thesis focuses on the truck task dispatching optimization problem under the background of marine container terminals - the pivots of ocean transportation. Proceeding from the practical application angle, several key bottlenecks of the examined problem in a container terminal are addressed, and several corresponding solutions are provided.

At the earlier stage of this research, a Real2Sim simulation framework is developed, which reproduced the most concerned details and logic of a real-world container terminal. Mechanisms that help to close the performance gap between reality and simulation are designed, which makes the obtained solutions as practical as possible. In the first primary research work, a spatial attention-based deep reinforcement learning (DRL) approach is applied to the examined problem. The DRL method verifies its effective performance and demonstrates the capability to properly cope with the multi-scenario issue. In this stage, the feasibility of DRL-based methods to solve online optimization problems is confirmed, and a solid

foundation is set up for the subsequent research work.

The examined problem is extended to a multi-objective optimization (MOO) version in the second stage of the research. Recently, demand for dynamic MOO is fast-emerging due to severe market competition and ever-growing requirements for customization and agility in business services. However, most of the existing evolutionary-based MOO approaches generate a finite set of trade-off solutions, which usually cannot efficiently obtain the most desired preference. To tackle such an issue, a preference-agile multi-objective optimization (PAMOO) methodology is proposed to permit users to dynamically adjust and interactively assign preferences. To achieve this, a novel uniform network is designed that could properly handle arbitrary user preferences. Benefit from such an attribute, a preference calibration method is then developed to further enhance the policy set quality.

For complex real-world optimization problems, it is costly for the DRL agent to learn a sophisticated policy from scratch, and the techniques to accelerate the training process are practical for real-life applications. This thesis explores the mechanisms to tackle such issues by introducing prior expert knowledge. For the single-objective case, an expert network-assisted dispatching model is designed, which has shown great convergence efficiency and the ability to handle high-level uncertainties. Following the similar principle, a policy fusion approach is proposed for the MOO problem, which reduces the training cost and demonstrates its potential to solve complex real-life optimization problems.

Acknowledgements

As I complete my doctoral dissertation, I find myself overwhelmed with a sense of gratitude when looking back on the past few years of my PhD career. This academic journey has been filled with challenges and opportunities, and I am deeply indebted to numerous individuals who have offered their support, guidance, and encouragement along the way.

First and foremost, I would like to express my most sincere appreciation to my esteemed supervisor, Prof. Ruibin Bai. During these years of my doctoral studies, he guided me with patience and wisdom through every stage of my research. With a rigorous academic attitude, profound expertise in the field, and an open-mindedness towards new ideas, he has provided me with invaluable insights that have shaped my understanding of the subject matter and sharpened my research skills. I am truly grateful for the time, energy, and care that Prof.Ruibin Bai has dedicated to my academic growth, and this dissertation would not have been possible without such outstanding mentorship. Also, I would like to recognize Prof. Rong Qu as my second supervisor for her continuous guidance and support throughout my doctoral research. Some detailed and incisive comments on my drafts during the writing process played an important role in my academic achievements.

Many thanks go to Prof. Tianxiang Cui. From the comprehension of the research topic to the construction of the theoretical framework, from the checking of the algorithm details to the refinement of the manuscript, every step has been accompanied by his meticulous and constructive feedback. Whenever I encountered difficulties or felt lost in the complex maze of academic research, his incisive advice always helped me break through the barriers and regain my confidence. Special appreciation is also extended to Prof. Huan Jin for the cooperation in some research projects. Her critical thinking towards the research field and related methodologies has taught

me so many lessons. Besides, I would like to acknowledge Prof. Jianfeng Ren for his crucial suggestions and comments on my research papers. His brilliant insights and inspiration on academic writing always make me think out of the box.

Last but not least, I am grateful to my senior classmates Dr. Yuchang Zhang, Dr Chaofan Tu, and Dr Xinan Chen for their generous help and support at the early stage of my PhD journey. I sincerely hope that all your careers will continue to thrive and prosper. The gratitude also extends to other members of the port laboratory. It is a great honor to have the opportunity to learn and work in such an outstanding research team.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter mainly focuses on the background of PhD research, including the introduction of Ningbo-Zhoushan port, Meishan terminal, the layout, and the major components in a container terminal. This research selects container truck task dispatching as the examined problem because it is often considered as the primary focus in a container terminal, as it provides crucial synergy between the sea-side operations and yard-side activities, and hence can greatly affect the terminal throughput and quay crane utilization. Moreover, its sensitivity towards the various scenarios and uncertainties further increases the challenges of the examined problems.

A taxonomy of truck dispatching optimization problems based on several classification factors is given in this chapter. The purpose of doing so is to provide multifaceted aspects of the examined problem and make the readers better comprehend the scope, practical aims, and challenges of this work. Finally, the main works and yielded outcomes during the PhD career, which also include some preparatory and branch-line works, are briefly introduced. The main contributions and the structure of this thesis

are listed at the end of this chapter.

1.1 Research Background

1.1.1 Ningbo Zhoushan Port

Ocean shipping is becoming the major transportation mode for global trade as maritime transportation has increased rapidly. More than 70% of the global trade by value is carried by sea, and the maritime freight transport will maintain a sustainable growth in several decades in the future, according to the estimation (Forum, 2021). Ningbo-Zhoushan port is one of the most valuable ports in China and plays a crucial part in China's international business and economic development. In 2023, the container throughput of Ningbo-Zhoushan port reached 35.3 million TEU (twenty-foot equivalent unit), ranking third in the world, and the cargo throughput reached 132.4 million tons, maintaining the world's first place for 15 consecutive years. The port consists of several port areas, which are usually called container terminals and are located on the coastlines beside Ningbo and Zhoushan, Zhejiang province. The Figure 1.1 (Wang et al., 2024) indicates the distribution of several main container terminals of Ningbo-Zhoushan port. Each terminal takes charge of its own specific trade business or shipping lines according to its position and geological conditions. Several research studies in this thesis are conducted against the background of the Meishan terminal. A detailed components and logic inside a container terminal is introduced in the section 1.1.2.

Figure 1.1: Distribution of main container terminals of Ningbo-Zhoushan port.

1.1.2 Overview of a Container Terminal

Container terminals are considered as the transportation hubs of global trade, which mainly use containers as the standard carriers. The advantages of the container transportation also lie in such standardization and the customized system based on the container shape. There are several main components in a container terminal, such as berth, quay crane, transporter, storage yard, and gate. The photographs of these container terminal components are shown in Figure 1.2.

Quay crane is the specialized equipment that is used for loading and unloading operations for the container vessels. Specifically, it transfers containers between vessels and transporters besides the vessels. According to the operational logic and business of a container terminal, the work efficiency of

(a) Quay crane (b) Yard crane

(c) Berth (d) Yard

(e) Gate

Figure 1.2: Several components of a container terminal.

the quay cranes basically decides the productivity bottleneck of the entire container terminal, since a higher quay crane utilization could reduce the vessel's berthing time. Quay cranes are considered as the most important equipment in a container terminal since they are expensive, and the operational cost is high. For such reasons, the utilization of quay cranes is required to be maintained at a certain level. Therefore, almost all the operations scheduling of a container terminal is conducted to maintain or

increase the operational efficiency of quay cranes as much as possible.

Berth is the area beside the coastline of a container terminal that is used for vessels' docking. Usually, a berth is required to be allocated and equipped with several quay cranes at the time of a vessel's arrival. With the development of the container industry, the berths that could accommodate larger tonnage vessels are the crucial resources and core competitiveness of a container terminal. The utilization of berths also reflects the busyness degree of a container terminal and is affected by the uncertain arrival time of vessels, weather conditions, and terminal daily operations management. Therefore, the scheduling and optimization for berth utilization also attract numerous academic studies.

Yards are used for storing the containers temporarily and are considered as buffer areas for loading to vessels or unloading from vessels. Yard is one of the most important resources in a container terminal. The Fig 1.3 presents the basic structure of a single yard, where a unique location of a container could be described by a quadruple $< yard, bay, row, tier >$. According to different types of business and containers, the yard could be further divided into a standard container yard, a clearance yard, a frozen container yard, a dangerous cargo yard, or an empty container yard. The management of yards includes re-arranging the positions of each container during the off-pick time, allocating the yard area or locations inside a yard for the gate-in or gate-out containers, and scheduling the yard cranes, which are important research directions of container terminal yard management.

The yard crane is the equipment that is located at each yard and aims at transferring the containers between transporters and the storage yards. Usually, the yard crane's movement is based on tires or rails in a yard

area. Similar to the quay cranes, the scheduling of the yard cranes also greatly affects the operation flow of the containers, thus becoming one of the factors that limit the throughput of the container terminal.

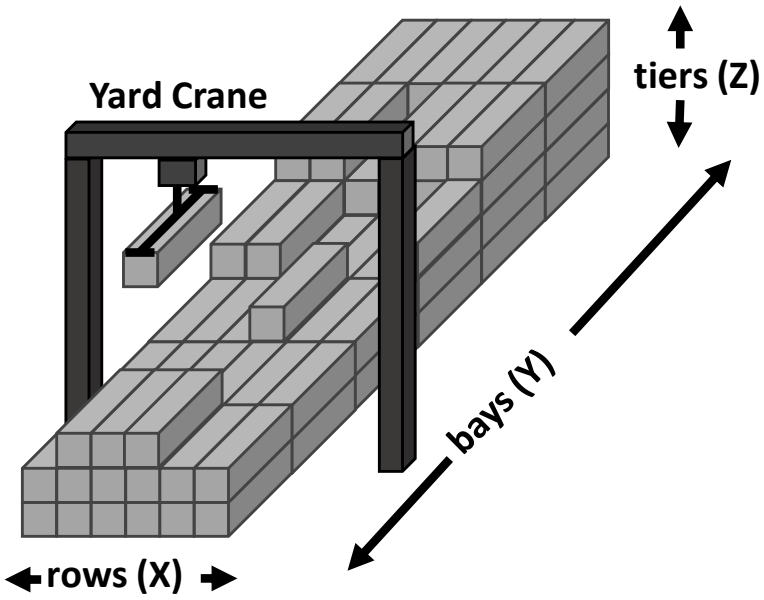


Figure 1.3: Container Location in a Yard

Transporter indicates either a manned truck or a driverless vehicle (introduced in section 1.2.1) and is in charge of transporting the containers horizontally in a container terminal. It is usually called a container truck in general. The container trucks are also divided into inner trucks and outer trucks. Inner trucks mainly focus on the transportation jobs inside the terminal area and belong to the container terminal. Outer trucks take charge of delivering the containers between container terminals and outside areas, which belong to customer companies. The scheduling of the container trucks is the primary operation for the container terminal optimizations since the trucks' operations connect several different types of equipment.

The gate is the boundary of a container terminal and is the entrance of the outer trucks that pick up and deliver the container from outside. It is also used for checking and recording during the container gate-in and gate-out process. With the increasing informative and intelligent level of the container terminals, the gate is becoming the hub of container information and bridges the close connections between customs, container companies, ship companies, and container terminal companies.

Generally, berth and quay cranes consist the seaside. Storage yard and gate area comprise the landside. The gate partitions the inside and outside of a container terminal. The transport area is considered as the intersection of the seaside and landside. The horizontal layout of the container terminal is described in Figure 1.4.

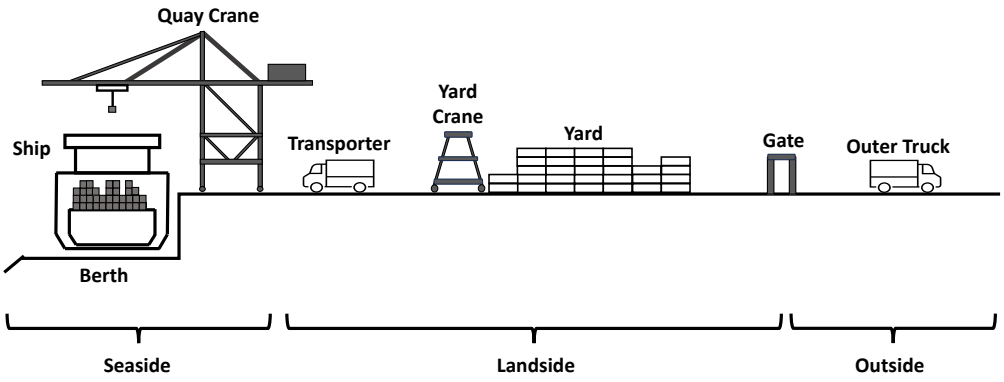


Figure 1.4: The horizontal layout of a container terminal

All these components in a container terminal are closely connected in the terminal's daily operations and management. Comprehension of the logic and effects of these equipment helps to better understand the various optimization problems in a container terminal. As the pivots among the international ocean logistics, container terminals are faced with the burden of increasing cargo throughput and are expected to cope with volatile

and dynamic situations to achieve high productivity. Studies of the container terminal management science and the corresponding optimization problems are fostering the modernization and intelligence degree of ports progressively. Some typical optimization problems in a container terminal are presented in the next section.

1.2 A Taxonomy of the Examined Problem

The section makes a taxonomy of the truck dispatching problem in container terminals based on the exploration in real-world cases and the reviewed papers. The purpose of doing so is to make the examined problem more comprehensive and thorough. Unlike the classical optimization problem, such as the vehicle routing problem (VRP), which has standard naming and strict mathematical definition for each variant problem, the examined problem is a real-life practical optimization problem that usually lacks a standard form and varies from case to case. According to the previous investigation, the examined problem could be classified through the following factors.

1.2.1 Schedulable Vehicle

The first classification factor of the examined problem is the target object in the dispatching process. In this thesis and my published papers, such a problem is called truck dispatching. In other reviewed papers, the schedulable vehicles could have other choices, which are summarized as follows. The corresponding photographs of these vehicles are shown in Figure 1.5.

1. **Truck** is the simplest and common schedulable vehicle. The truck in this thesis indicates the container terminal's inner piloted trucks. One characteristic of such a vehicle is that some details of the implementation of a task may be decided by the driver, such as choosing a specific driving route, whereas a driverless vehicle usually follows a pre-arranged route.

2. **Automatic Guided Vehicle (AGV)** is usually used in automated warehouses and manufacturing factories. AGV has been introduced to automated container terminals in recent decades (Fazlollahtabar and Saidi-Mehrabad, 2015; Zhong et al., 2020; Xing et al., 2023; Drungilas et al., 2023; Rashidi and Tsang, 2011; Xiaolong and Jiawei, 2016; Zhang et al., 2021). Its largest characteristic is the driverless property, and each route and task assignment is centrally controlled. Such vehicles are suitable for a highly automated and intelligent port.

3. **Straddle carrier** is used for transferring empty containers in a container terminal (Royset et al., 2009; Zehendner et al., 2015; Dkhil et al., 2018; Cai et al., 2012; Soriguera et al., 2007). Its job logic has little difference from truck and AGV because it only works between quay cranes and empty container yards.

4. **Automated Lifting Vehicle (ALV)** is another vehicle that transfers containers between sea-side and yard-side, which is also used in automated container terminal (Nguyen and Kim, 2009; Gupta et al., 2017; Sadeghian et al., 2014; Bae et al., 2011; Roy and de Koster, 2018). Unlike other vehicles, whose loading or unloading container operations are executed by cranes but ALV could lift the container from a buffer area (on the ground near the QC or YC).

5. **Automated Intelligent Vehicle (AIV)** or intelligent autonomous

vehicle (IAV) is similar to the AGV. The advantage of them is a certain degree of self-decision ability. Apart from moving autonomously by pre-defined paths, they could also react to various obstacles compared with AGVs (Zaghdoud et al., 2012; Nguyen et al., 2018; Zaghdoud et al., 2013).

(a) Truck

(b) AGV

(c) Straddle carrier

(d) ALV

(e) AIV

Figure 1.5: Various types of schedulable vehicles for transferring containers in a terminal.

1.2.2 Optimization Objectives

Several objectives are concerned in the container terminal truck dispatching problem. First of all, quay crane utilization is the most common objective to be maximized in this problem since quay cranes are the most valuable equipment in a port. Increasing their utilization could shorten the vessels' berthing time, which has a great influence on the business benefit of a port. The variants of this objective that are related to quay crane utilization, such as QC idle time, average makespan, or total makespan are also adopted in some other research. Another common objective is the (empty) traveling distance of the transporters because plenty of container terminal managements start to consider the energy consumption and carbon emission (Mansouri et al., 2015). Barely researches optimize this objective individually, and it usually exists in the multi-objective optimization field (Kim et al., 2013; Homayouni and Tang, 2013; Hu et al., 2019; Dkhil et al., 2017; Liu et al., 2016). Some other research also considers some objectives like equipment (QC or YC) moving distance or outer truck waiting time.

1.2.3 Dispatching and Scheduling

Dispatching and scheduling are both expressions of methods to allocate container transporting jobs to vehicles. The difference lies in the information needed to carry out the allocation. For scheduling problems, a predictive job sequence, both with the arrival times of jobs, is required to carry out a scheduling (Zhicheng et al., 2014). Accordingly, it outputs a schedule plan - when and which job to take for every vehicle. Dispatching usually indicates a real-time decision-making process where the dispatching of tasks only relies on the state information at some specific time step.

The expression dispatching and scheduling usually also indicates online and offline optimization.

1.2.4 Vehicle Trigger and Job Trigger

Container truck dispatching can be initiated by either a vehicle or a job. For vehicle-initiated dispatching, the vehicle starts to execute a new task only when it has just finished its previous tasks. For job-initiated dispatching, any available task at any time could be dispatched to any truck, even if the truck has not finished its current job. Most research focuses on the vehicle-initiated dispatching problem because it is easy to implement. The job-initiated dispatching is more complex because the action space is larger, and it allows decision-makers to plan for longer-term horizons.

1.2.5 Simulation and Mathematical Modeling

There are two methods that could implement the truck dispatching optimization: simulation and mathematical modeling. For the simulation approach, the process of truck dispatching is implemented based on the support of some simulation tools or building the simulation from scratch by programming (Angeloudis and Bell, 2011). Such methods are usually laborious and time-consuming to build the experiment environment, but can achieve flexible modification of the logic and could be applied to the online optimization problem. For the mathematical modeling approach, the advantage is the ability to achieve an optimal solution compared with the simulation approach. However, such a method could only optimize offline optimization problems, and the time for obtaining an optimal solution

is usually considerable.

1.3 Main Work in PhD Career

This section introduces the main work of my PhD research and the relevant contributions. My main research topic focuses on the online container truck task dispatching problem in marine container terminals. Some other branch-line research topics, such as port-related forecasting problems and reinforcement learning-based approaches for combinatorial optimization problems, are also involved in my PhD career.

The early stage of the research is devoted to the investigation of the business logic and operations of the container terminal, including the review of the container terminal optimization papers and on-the-spot investigation of a real container terminal in Ningbo. One of my research works [2] that utilizes a decomposition-ensemble methodology to solve a container daily gate-in(out) forecasting problem is yielded during this period.

Based on the foundation and comprehension of the container terminal in the early stage, a simulation environment of the Ningbo-Zhoushan port Meishan terminal, which is our research target is developed. In this process, I strive to reproduce the details and logic of the real-world container terminal, especially the factors that may have an influence on the examined problem, such as stochastic service time and traffic congestion. The simulation becomes the platform for algorithm training and strategy evaluation. In addition, the simulation model also contributes to a digital twin program of Ningbo Daxie Container Terminal Co., Ltd, and the research [1] which proposed a novel and pioneering methodology deep reinforcement

learning hyper-heuristic (DRL-HH) framework.

In the meantime, the learning-based (basically reinforcement learning) combinatorial optimization approaches are becoming a popular research topic. Abundant research about RL and COPs is reviewed and studied at this time. This work directly contributes to a survey paper that discusses how machine learning assists to solve vehicle routing problem [3], together with my supervisor and other team members. The comprehension and experience of DRL-HH methodology [1] also contribute to the work [6] that uses a similar approach to solve the curse of action space dimensionality of multi-agent reinforcement learning. The idea that utilizing RL-related approaches to solve COPs contributes to another research work [4], which is the seminal work to introduce the RL method for the jigsaw puzzle problem.

The first main research work basically focuses on the study of the online truck task dispatching problem, and the objective is to maximize the QC utilization. Generally, the RL method is adopted, and the algorithm is trained on the simulation model, which is developed at an earlier stage. The overall methodology of this research is similar to the previous work [1], but the performance is improved by modifying the reward signals and avoiding the invalid action space. This work basically demonstrates the feasibility of the RL as a methodology for the examined problem and is a milestone in my main research timeline. The output of this study has been published as a journal paper [5].

The second main research extends the existing methodology to solve a multi-objective optimization (MOO) problem where another objective, truck empty distance, is taken into consideration. What's more, the traditional

multi-objective optimization methods like MOEA/D (Zhang and Li, 2007) and NSGA-II (Deb et al., 2002) only generate finite solutions and are usually unable to evenly explore the approximated Pareto front. The proposed method in this work helps to alleviate such drawbacks. This contribution is achieved by designing a customized network structure for MOO, and the feature crossing operator is verified as a key element to make this method work. This could be meaningful research in practice, which uses RL to solve a real-world complex multi-objective optimization problem, especially when the user preference of the objectives is required to be adjusted dynamically (Roijers et al., 2013). This work is already finished as a journal paper under review.

Finally, the mechanisms to accelerate the training are designed by introducing the prior expert knowledge to enhance the RL agent, and such consideration is a vital issue that goes through the entire PhD research. For the first single-objective problem, a two-stage approach that leverages imitation learning and a novel expert network-assisted structure is designed. For the subsequent MOO problem, a methodology called policy fusion is proposed, which makes the features extracted by a single objective policy help to augment the MOO agent. The related work is reported in the papers [5].

The publications during the PhD career are listed below:

[1] Zhang, Y., Bai, R., Qu, R., Tu, C., and Jin, J. (2022a). A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. European Journal of Operational Research, 300(2):418–427.

[2] Jin, J., Ma, M., Jin, H., Cui, T., and Bai, R. (2023). Container terminal daily gate in and gate out forecasting using machine learning methods.

Transport Policy, 132:163–174.

[3] Bai, R., Chen, X., Chen, Z.-L., Cui, T., Gong, S., He, W., Jiang, X., Jin, H., Jin, J., Kendall, G., et al. (2023). Analytics and machine learning in vehicle routing research. International Journal of Production Research, 61:4–30.

[4] Song, X., Jin, J., Yao, C., Wang, S., Ren, J., and Bai, R. (2023). Siamese-discriminant deep reinforcement learning for solving jigsaw puzzles with large eroded gaps. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pages 2303–2311.

[5] Jin, J., Cui, T., Bai, R., and Qu, R. (2024). Container port truck dispatching optimization using real2sim based deep reinforcement learning. European Journal of Operational Research, 315(1):161–175.

[6] Cui, T., Yang, X., Jia, F., Jin, J., Ye, Y., and Bai, R. (2024b). Mobile robot sequential decision making using a deep reinforcement learning hyper-heuristic approach. Expert Systems with Applications, 257:124959.

1.4 Contributions of the Thesis

The main work of the PhD is summarized above, among which the container truck dispatching is the main research topic. The contributions of this thesis are concluded as below.

- A high fidelity simulation model of a real container terminal is developed, and a Real2Sim framework is proposed that helps to close the gap of policy performance between reality and simulation.

- The feasibility and effectiveness of the RL-based approach are verified through the single objective container truck task dispatching problem. Some modifications are adopted that help to improve the previous work (Zhang et al., 2022a) and result in significant performance gains.

- A methodology that uses RL to solve a real-life MOO problem is proposed. Such approaches only use a uniform model to handle arbitrary objective preferences and allow users to dynamically adjust the preference weights.

- Two mechanisms that help to accelerate the training of RL agents by introducing prior expert knowledge for both single and multi-objective optimization problems are designed.

1.5 Overview of the Thesis

The remaining of the thesis is constructed as follows.

- **Chapter 2: Literature Review**

  This chapter provides a thorough review of the research that is relevant to our research topics and techniques, including container terminal combinatorial optimization problems, the current methodologies for the examined container truck task dispatching problem, approaches that utilize reinforcement learning algorithms, methodologies of multi-objective optimization, and researches that adopt simulation techniques.

- **Chapter 3: Online Container Truck Task Dispatching Prob-**

lem

The examined problem is introduced in detail in this chapter. The entire process and logic of the container truck dispatching are described. The main challenges of the problem and research motivations are discussed. Finally, a mathematical formulation of the examined problem is given, and simulation development to achieve the logic of the problem is introduced.

- **Chapter 4: Methodology for Single-Objective Dispatching**

  The basic methodology to utilize the reinforcement learning approach to solve the examined problem is reported in this chapter. The contents also include the Markov decision process (MDP) modeling, problem instance and scenario design, and some improvements that enhance the previous work.

- **Chapter 5: Methodology for Multi-Objective Dispatching**

  The examined problem is extended to a version of multi-objective optimization in this chapter. A proposed methodology called preference agile multi-objective optimization (PAMOO) is proposed. The experimental results demonstrated its performance on solution quality, generalization, and sample efficiency.

- **Chapter 6: Mechanisms for Prior Expert Knowledge Augmentation**

  To accelerate the training process and avoid the agent learning from scratch when faced with the challenges of multi-scenarios and multi-objective cases, two mechanisms that introduce prior expert knowledge to augment the agent dispatching policies, namely, the expert network assisted dispatching model and policy fusion approach, both for single and multi-objective optimization cases are proposed.

- **Chapter 7: Conclusion and Future Work**

  This chapter concludes and deeply deciphers the main works and contributions of the thesis. The practical significance of the proposed methodologies towards reality is discussed. Some limitations of the work and several potential improvement directions are also provided.

Chapter 2

Literature Review

This chapter covers the research related to the works and research directions in this thesis. Firstly, the common solutions to tackle the aforementioned container terminal-related optimization problems are studied. These approaches provide the reference value when considering the examined problem and selecting the benchmark methods. For the examined truck dispatching problem, the common solutions are classified into heuristic approach, meta-heuristic, hyper-heuristic, and offline methods. These methods reveal the challenges of the examined problem and indicate some concerns when designing the relevant solutions, such as feature engineering. Consequently, a review section about reinforcement learning is given since it is the main methodology of this thesis. This section focuses on the theoretical foundations, classical RL algorithms, the recent progress of utilizing RL to solve some traditional optimization problems, and research that uses RL to solve truck dispatching-related problems. The next section introduces the methodology of multi-objective optimization since MOO is one of the research directions in this thesis. This section mainly introduces the traditional MOO approaches that are based on evolutionary

computation, research about MOO of container terminal operations, and multi-objective reinforcement learning. Finally, the research that adopts simulation-based methods is reviewed. Different approaches for developing the container terminal simulation are concluded. A special concept - digital twin, along with its achievements in recent years, is briefly introduced.

2.1 Container Terminal-Related Optimization Problems

Over several decades, a series of port-related optimization problems and operations are defined such as quay crane or yard crane scheduling, truck task dispatching, berth allocation, and container storage assignment (Carlo et al., 2014), where most of them are characterized with NP-hard nature (Vacca et al., 2007; Kim and Lee, 2015). Optimizations of these operations and problems could help to improve the utilization ratio of related equipment, shorten the vessels' duration in the berth, thus gaining more business competitiveness for a container terminal. In reality, more customized port-related optimization problems are also concerned according to different taxonomies (Weerasinghe et al., 2024). The study of these problems and relevant solutions could help to better comprehend the challenges and underlying logic of the examined truck dispatching problem.

In this section, berth allocation, yard crane scheduling, and container relocation problems are reviewed. All these port-related optimization problems could make interactions to the truck dispatching problem more or less, thus increasing the degree of dynamics and uncertainties of this problem. For example, the container space allocation could determine containers' birth-

places and destinations in the process of truck dispatching. The yard crane scheduling and container relocation operation happen at the intermediate segment of a truck transportation task, which affects the completion time of this task. Berth allocation and quay crane scheduling occur at the seaside of a container terminal, which is directly related to the quay cranes, thus influencing the utilization of the quay cranes, which is one of the objectives of the truck dispatching problem. The truck task dispatching is the examined problem in this research, and a more detailed review is given in the following sections.

2.1.1 Berth Allocation Problem

Once vessels approach a container terminal, adequate space slots need to be allocated for the vessels to handle subsequent loading and unloading of containers. Such a process is called berth allocation in the literature. The allocation policy could also affect the efficiency of trucks and quay cranes. To model such a problem, a common way is to discretize the berth space and time into integer berth sections and time units. Each vessel needs at least one berth section and a planned time horizon to finish the loading and unloading operations. The Figure 2.1 presents an example of a berth allocation plan for four vessels, where vessel 1 requires three berth sections and ten hours for the berthing, and the gray cell indicates the idle berth sections for a time period.

A general formulation of the berth allocation problem is presented as follows. Some defined variables are presented in the Table 2.1. The relative positions of vessel rectangles depicted in Figure 2.1 are defined by the expressions (2.1) and (2.2).

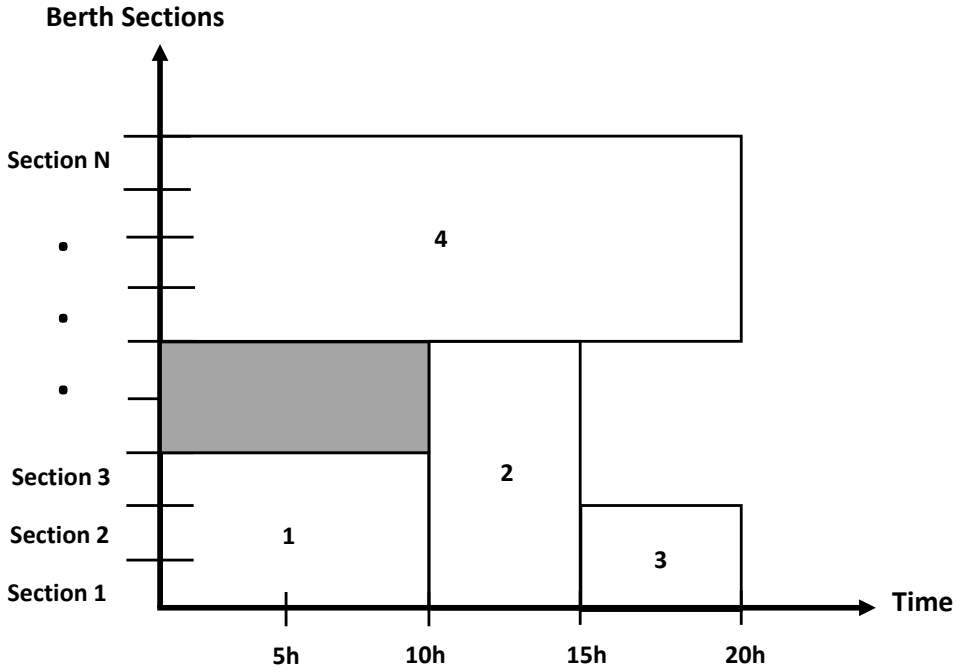


Figure 2.1: An example of a berth allocation plan.

Variables	Description
p_i	The processing time of vessel i.
l_i	The length of vessel i which is measured by the required berth sections.
a_i	Arrival time of vessel i.
d_i	Departure time of vessel i.
s_i	The operation start time of vessel i.

Table 2.1: Variables of berth allocation problem formulation.

$$\alpha_{ij} = \begin{cases} 1, & \text{Vessel rectangle } i \text{ locates at the left of} \\ & \text{the vessel rectangle } j \text{ with no overlap} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

$$\beta_{ij} = \begin{cases} 1, & \text{Vessel rectangle } i \text{ locates below the} \\ & \text{vessel rectangle } j \text{ with no overlap} \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

The mathematical formulation of the berth allocation problem could be presented as below.

$$\min \sum_{i=1}^{S} (d_i - a_i) \tag{2.3}$$

s.t

$$p_i \geq 1 \quad \forall i \tag{2.4}$$

$$l_i \geq 1 \quad \forall i \tag{2.5}$$

$$\alpha_{ij} + \alpha_{ji} + \beta_{ij} + \beta_{ji} \geq 1 \quad \forall i, j \tag{2.6}$$

$$\alpha_{ij} + \alpha_{ji} \leq 1 \quad \forall i, j \tag{2.7}$$

$$\beta_{ij} + \beta_{ji} \leq 1 \quad \forall i, j \tag{2.8}$$

$$s_i + p_i = d_i \quad \forall i \tag{2.9}$$

The optimization objective is to minimize the vessel's duration of stay at the container terminal, including the waiting time and operation time

in order to make vessels leave the terminal faster, which is depicted by Expression (2.3). Constraints (2.4) and (2.5) indicate each vessel takes at least one single time and berth section unit. Expression (2.6), (2.7) and (2.8) guarantee any two vessel rectangles have no overlapped area, which means no vessels could share the same berth sections at a same time in another word. Equation (2.9) indicates that a vessel is require to leave the container terminal immediately once it finish its all loading and unloading operations. Such a formulation approach is the basis of the berth allocation problems. Some other variants are considered in the literature. For example, the berth location could be discrete or continuous, and other optimization objectives are minimizing the total waiting time of vessels or the total make-span of the handling time (Imai et al., 2001; Guan and Cheung, 2004).

At the early stage of the berth allocation study, cases are considered when all vessels are available for the berth allocation (Imai et al., 1997), which makes it a planning problem. The study of berth allocation usually focuses on the dynamic version in recent decades, where the arrival time sequence of vessels is unknown while the vessel's handling time could be known, somehow estimated, or affected by quay crane assignment operations (Rodrigues and Agra, 2022; Dragović et al., 2024). Most of the research interests focus on the uncertainties of the vessel handling time (Guo et al., 2021; Agra and Rodrigues, 2022), service priorities (Imai et al., 2003), continuity attribute of berth area (Lim, 1998; Cordeau et al., 2005), and multi-port berth allocation (Martin-Iradi et al., 2022). In reality, the vessel's handling time relies on the number of quay cranes associated with the vessel. Therefore, a large number of researchers also study the integrated berth allocation and quay crane assignment since these two problems are closely related (Park and Kim, 2003; Meisel and Bierwirth, 2006; Imai et al., 2008; Meisel and

Bierwirth, 2009).

2.1.2 Yard Crane Scheduling Problem

At any time, there may be several trucks queuing in a yard, waiting for the yard crane to load or unload containers for the arriving trucks. The yard crane scheduling problem is to decide the yard crane service sequence for several queuing trucks. The Figure 2.2 shows a case when five trucks with their corresponding transportation tasks are queuing in this yard block, and the yard crane needs to choose the next truck to serve. Usually, the yard crane scheduling is conducted under the concerns about factors like crane moving distance, urgency level of each task, and subsequent status. The common objective is to minimize the total waiting time of trucks or maximize the quay crane utilization as an auxiliary policy. According to some empirical studies, the policy of yard cranes scheduling exerts sensitive effects on the truck dispatching optimization, thus affecting the quay crane utilization.

The formulation of a static version of the yard crane scheduling is provided here, where the arrival times of trucks are assumed to be known in advance. Let a_i be the truck arrival time of job i and $a_i \leq a_{i+1}$. The number of jobs in the yard block is n. The p_i is the processing time of the job i, which includes the yard operation and possible container relocation process. The b_i is the bay index of the job i. Let l be the length of a bay and v be the constant speed of trucks in the yard area. Assume e_i is the end time that the yard crane completes the processing of job i. The sequential order relations between two jobs are defined by Equation (2.10). Then the problem could be formulated as follows.

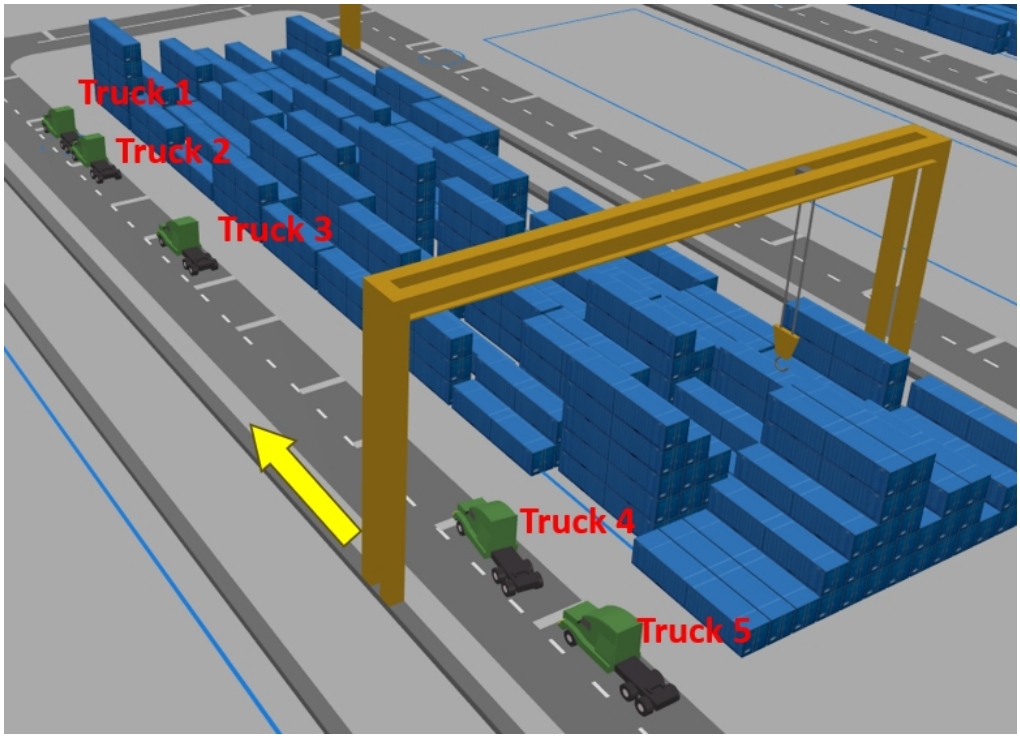


Figure 2.2: An indication of yard crane scheduling problem

$$X_{ij} = \begin{cases} 1, & \text{if job } j \text{ is served right after job } j \\ 0, & \text{otherwise} \end{cases} \tag{2.10}$$

$$\min \sum_{i=1}^{n} (e_i - p_i - a_i) \tag{2.11}$$

s.t

$$e_i \geq a_i + p_i \quad \forall i \tag{2.12}$$

$$X_{ij}(e_j - e_i) = X_{ij}(\frac{|b_i - b_j|l}{v} + p_j) \quad \forall i, j \tag{2.13}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \quad \forall i \in [1, n-1] \tag{2.14}$$

$$\sum_{i=1}^{n} X_{ij} = 1 \quad \forall j \in [2, n] \tag{2.15}$$

The objective (2.11) aims at minimizing the total job waiting time. Constraint (2.12) gives the relations about the job completion time, crane processing time, and truck arrival time. Constraint (2.13) describes the time interval between two successive jobs. Constraint (2.14) and (2.15) ensures X_{ij} describes a legal crane service sequence. The decision space lies in X_{ij}, which could affect the e_i in the objective indirectly.

The yard crane scheduling is indeed a type of one-machine scheduling problem (Vallada et al., 2023). The coming trucks are the arriving jobs at random time steps, and the yard crane could be considered as the machine. Once the machine finishes one job, the scheduler would allocate one of the queuing jobs to the machine or do nothing. The aims of yard crane scheduling are usually to reduce the waiting time of transporters or the operation time of the yard crane. In some scenarios, especially for cases where external trucks are involved, the due times of containers to be retrieved need to be guaranteed. In other cases, crane scheduling serves as an auxiliary decision-making process that collaborates with other operations, such as truck dispatching or routing. One stream of the solutions relies on the known job sequence or predictive arrival time of trucks (Li et al., 2009; Guo et al., 2011; Chang et al., 2011). Some other researchers treat this process as a real-time decision problem, and reinforcement learning should be considered as a suitable method (Aydin and Öztemel, 2000; Kim et al.,

2003; Fotuhi et al., 2013). The yard crane scheduling could also cooperate with container relocation operations (Galle et al., 2018), AGV scheduling (Zhang et al., 2024), and container space allocation operation (Yang et al., 2022).

2.1.3 Container Relocation Problem

The container relocation problem (CRP) happens at the time when a target container to be retrieved is located under other containers. Retrieving the target container requires removing any containers on top of the target container. Figure 2.3 indicates the process of container relocation in a single yard bay, where containers 4 and 5 need to be removed in advance before the fetching of the target container 3.

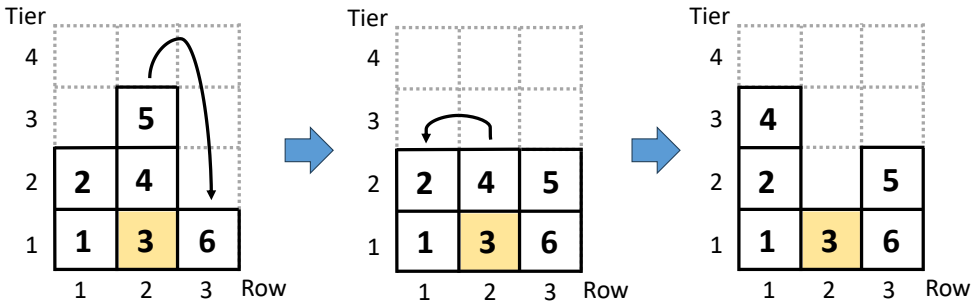


Figure 2.3: Indication of a container relocation process

With the frequency of relocation operations increasing, the operational efficiency of a yard would decrease rapidly. The container relocation is to decide a proper position for the relocated container during the container retrieval process. A better relocation strategy could reduce the number of relocation operations to increase the efficiency of the yard. Figure 2.4 shows several relocation paths for containers $c1$, $c2$, and $c3$ according to various strategies, which cause a different number of relocation operations

r. The relocating strategy could also be divided into one bay relocation, multiple bay relocation, and relocation with consideration of attributes of containers (Zheng, 2018).

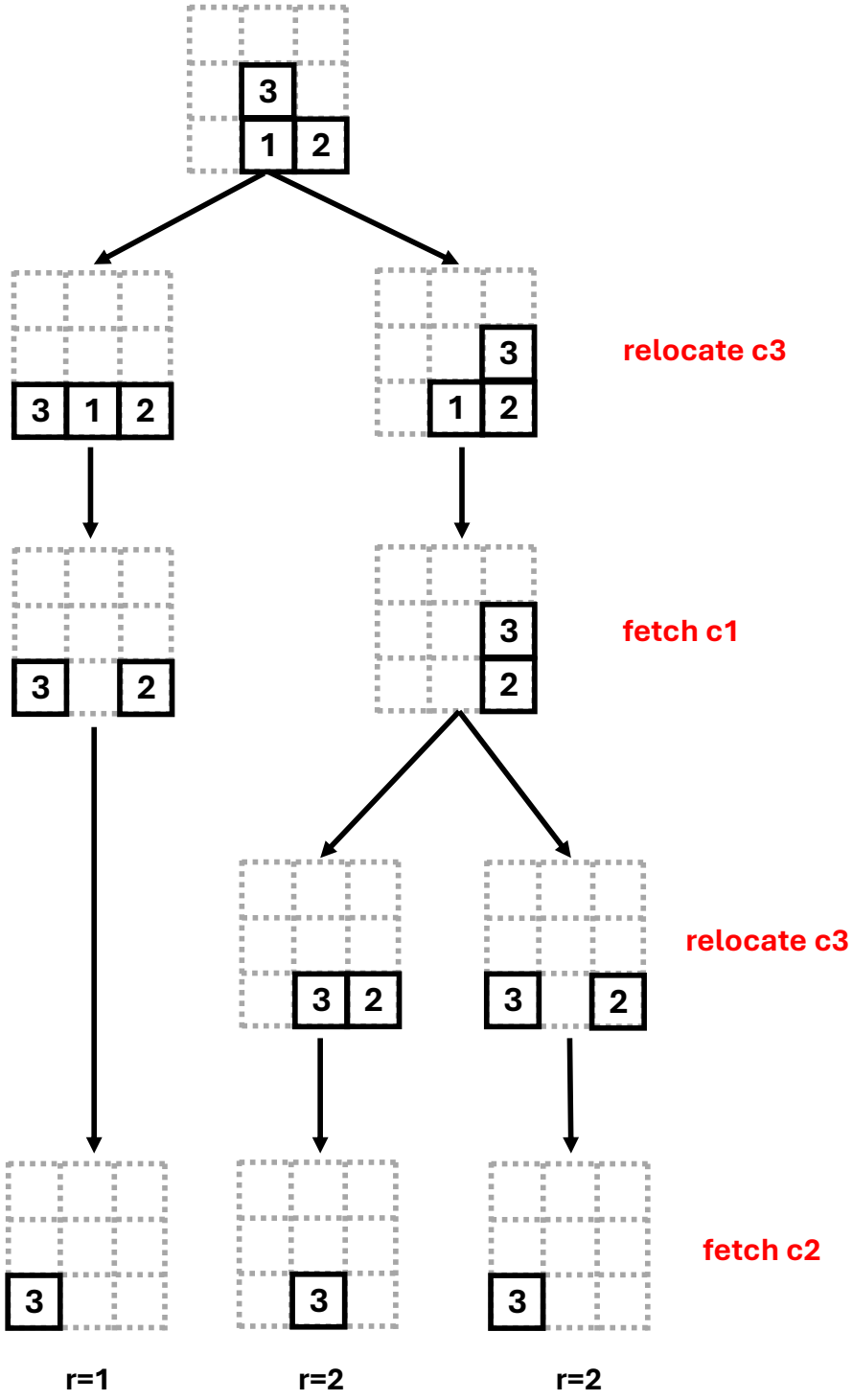


Figure 2.4: Indication of container relocation paths

The simplest version of the container relocation problem is static, assuming all the containers are retrieved in a prescribed order. The objective is to minimize the total container relocation moves. Such a problem is firstly studied by Kim and Hong (2006). Usually, the static CRP could be solved by mathematical modeling-based such as integer programming (Wan et al., 2009) and column generation (Zehendner and Feillet, 2012). However, the real-life container relocation process always happens along with the new containers' arrival, which makes this problem become dynamic container relocation problem (DCRP). The common methods for DCRP are heuristic approaches (Zhu et al., 2012; Hakan Akyüz and Lee, 2014; Expósito-Izquierdo et al., 2014). Some researchers also consider the cross-bay relocation operations where the objective in such a setting is the total operation time (Lee and Lee, 2010).

RL-based approach is also used to solve the dynamic container relocation problem (DCRP) (Bucur and Hungerländer, 2017). The author defined the state as the current container bay configuration and the sequence of the arrival and departure containers. The action is defined as a bay column either for stacking an arrival container or for the relocated position for the container blocking the current retrieved container. The reward was set minus a scalar for every relocation move and zero for other legal moves to minimize the total relocation moves. The reinforcement learning model and problem-specific heuristic are used to guide the Monte Carlo Tree Search to generate the best moving sequence.

Container pre-marshalling is another important operation during the off-peak time of the container terminal, which aims at making the container retrieval procedure quicker by re-ordering the containers beforehand. Hottung et al. (2020) proposed deep learning assisted heuristic tree search to

solve the container pre-marshalling problem. A tree search procedure is modeled for the problem. The node in the search tree is defined as the configuration of containers at the same bay, and the solution is considered as the path from the root to one of the feasible leaf nodes. A branching network for choosing the branch to explore and a bounding network for predicting the cost of completing the current solution are used to guide the tree search procedure.

2.2 Traditional Methods of the Examined Problem

2.2.1 Heuristic Approaches

The heuristic approach is the most widely used methodology and is easy to implement for such decision problems. The first heuristic method is called dedicated dispatching, which is the simplest approach and is still used by many real-world container terminals. Dedicated dispatching makes all trucks organized as different groups, and each group of trucks only executes tasks that are dedicated to one particular quay crane or ship. Such a method is effortless to develop and deploy in reality. However, many studies have proved that a dedicated dispatching policy causes a high empty driving ratio and is infeasible for an automatic and intelligent container terminal (Tao and Qiu, 2015; Le-Anh et al., 2004; Grunow et al., 2007).

The most common heuristics consider some attributes, such as distance, estimated time, or QC queue length as priority factors when allocating a

task to a specific truck. These features are dynamically changed during the entire process. Koo (2013) discussed the superiors and inferiors of vehicle-trigger and task-trigger dispatching modes.

Dulebenets (2016) explored five simple dispatching rules, namely, First Available, Round Robin, Random Vehicle, Shortest Distance, and Shortest Queue, which means as the names suggest. A comprehensive evaluation of these rules is implemented and tests different quay crane performance indicators. Such dispatching rules are designed by prioritizing the task assignment and are computationally efficient. However, these methods fail to achieve competitive performance because of their greedy attributes and are usually served as baselines in most research. Some commonly used priority factors in the explored papers are summarized in Table 2.2.

Symbol	Priority Factor
RD	No priority, random dispatch
SD	Shortest Distance
RR	Round Robin
SC	Shortest QC Queue
HU	Highest QC Urgent Level
HD	Highest Demand
LS	Lowest Supply

Table 2.2: Priority Factors.

Chen et al. (2016) proposed a manually crafted dispatching rule that based on supply-and-demand mechanism of QC, where supply is defined as an estimated number of truck that could able to reach the specific QC in a fix-size time window and the demand is defined as the estimated number of container that the QC need to operate in this time window. Such a heuristic considers spatial and temporal factors and is proven to outperform the existing dispatching method that is used in Ningbo Port Meshan Container Terminal.

2.2.2 Meta-heuristic Algorithms

One stream of the operations research community is dedicated to automatically search or generate a problem solution, which is called heuristic search or meta-heuristic. Solution has to be encoded as an evolvable representation (usually a vector in genetic algorithm and a decision tree in genetic programming) and is iteratively refined during the evolving process (Osman and Kelly, 1997; Voß, 2000). Genetic programming (GP) has advantages in obtaining real-time decision heuristics. It encodes individual chromosomes as representations that are able to yield decisions (usually tree structures) and makes the individuals evolve with similar operators of genetic algorithms such as crossover, mutation, and reproduction (Koza and Poli, 2005; Koza, 1994). Chen et al. (2020b) proposed a data-driven GP heuristic that could obtain a decision-making policy for the online truck dispatching problem. The experiments demonstrated that GP GP-based method could learn some hidden factors during the evolving process and outperformed manually handcrafted heuristic approaches.

2.2.3 Hyper-heuristic Algorithms

It is worth noting that the hyper-heuristic framework stood out in the operations research community in the past two decades. Unlike meta-heuristic that operates directly on specific solutions, hyper-heuristic operates on the heuristic space, which makes it capable of handling cross-domain optimization (Pillay and Qu, 2018). In other words, the hyper-heuristic framework uses heuristics to choose or generate low-level heuristics (see Figure 2.5). The hyper-heuristic framework has achieved great success in various traditional CO problems (Soria-Alcaraz et al., 2014; Rahimian et al., 2017;

Ahmed et al., 2019) and shows its potential for real-world complex online optimization problems. In the work of Chen et al. (2016), a genetic algorithm-based hyper-heuristic framework for truck dispatching is proposed. Three low-level heuristics are designed, specifically, short distance first, highest unbalanced task amount first, and highest urgent level first dispatching. Chen et al. (2022) proposed a double-layer genetic programming hyper-heuristic to cope with the challenges of multi-scenario issues of seaport truck dispatching by designing logical and arithmetical layers in a GP individual to handle the scenarios and decisions simultaneously.

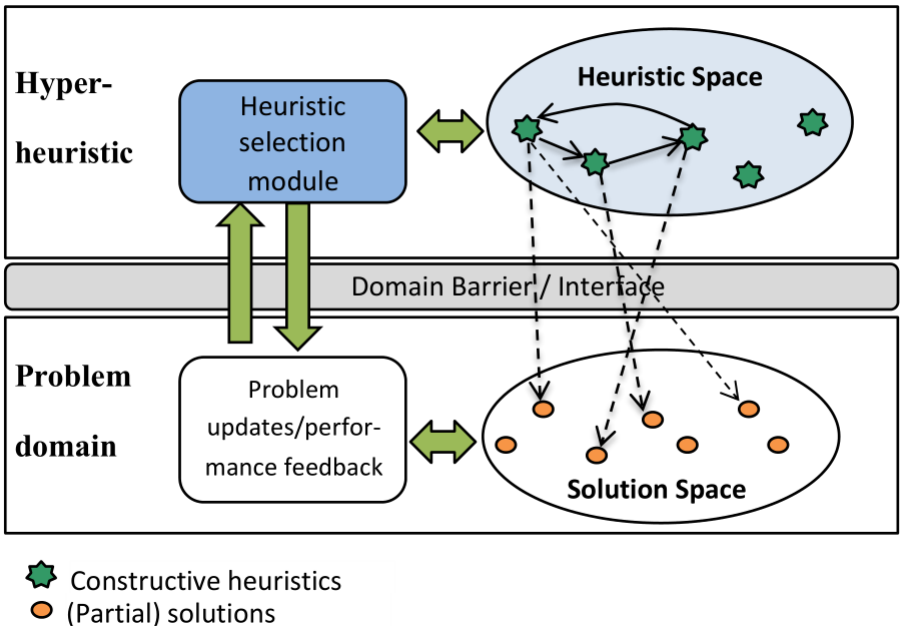


Figure 2.5: Indication of the hyper-heuristic framework.

2.2.4 Offline Optimization Methods

The method described above aims at solving the online optimization of truck dispatching. Such problems could also be solved in an offline manner by meta-heuristics or exact algorithms. Thus, the solution becomes a long-term dispatching sequence planning. For exact algorithms, the problem

needs to be precisely formulated as a mathematical model that includes an objective function and a set of constraints. The exact method usually relies on the divide-and-conquer strategy, such as branch-and-bound, which guarantees optimality but is computationally infeasible for large-scale problems. Approximations could somehow alleviate such an issue by making the algorithm finish in polynomial time, but at the expense of losing the guarantee of optimality (Bengio et al., 2021). Cao et al. (2010) focused on integrating yard truck and yard crane scheduling. A mixed integer programming (MIP) model is formulated for a mathematical solver. Two decomposition methodologies, namely, the general Benders' cut-based method and the combinatorial Benders' cut-based method to reduce the computational cost. Lu and Jeng (2006) modeled the yard truck dispatching policy as a min-max nonlinear integer programming model. The method is also combined with the heuristic approach for better performance. The experiment shows its competitiveness against some existing dispatching rules.

Genetic algorithm (GA) is a suitable way for offline scheduling because a dispatching plan could be easily encoded as a decision sequence, which serves as the chromosome vector in the genetic algorithm framework. Lee et al. (2010) used a hybrid genetic algorithm and minimum cost flow network model to minimize the makespan at the quay side. The ready times for tasks are encoded as chromosomes in GA, and the minimum cost flow is used to determine the prime movers of the task sequence. The result shows its performance against the local search-based method. Cao et al. (2008) focused on combining the truck dispatching problem and the container storage allocation problem. The objective is to balance the travel time and queuing time of each container in yards. A GA-based approach and a greedy heuristic are designed to solve the problem. The result shows its ability to obtain optimal solutions in small-sized instances. Bose et al.

(2000) used a double cycle mode dedicated dispatching heuristic where the trucks repeat the loading and unloading task cycle for one quay crane to reduce the deadhead rate. A genetic algorithm is then introduced to implement dynamic dispatching. The result shows that GA based dynamic dispatching outperforms the dedicated dispatching method. Kim and Bae (2004) proposed a look-ahead dispatching policy that took the QC's future task information into consideration to reduce the QC operation delay. The problem was formulated as an MIP model, and a heuristic algorithm was designed to solve it within a reasonable computational time. Vis et al. (2005) considered this problem from another point of view, where all available delivery tasks were forced to be finished in a time window, and the objective was to minimize the amount of deployed vehicles. To tackle the truck dispatching problem with meta-heuristic, a solution (dispatching sequence) needs to be encoded as an evolvable representation, which can be iteratively improved during the evolving process. Other similar researches that use meta-heuristic for truck dispatching in container terminal could be found in (He et al., 2013; Nishimura et al., 2005; Choi et al., 2011; Skinner et al., 2013; Luo et al., 2016; Niu et al., 2014; Luo and Wu, 2015). Hybrid approaches of these methods also show significant research interests (Chen et al., 2013; Hsu et al., 2021; He et al., 2015). There is abundant literature for the offline version of the examined problem. However, offline optimization aims at planning a long-term dispatching sequence for a static problem instance rather than a real-time decision-making problem, which makes it difficult to be generalized to different cases with uncertainties.

2.3 Reinforcement Learning

This section briefly introduces the development of reinforcement learning (RL) nowadays and its progress in operational research. Since our main approach for the examined problem is based on the RL method, research about the optimization of truck dispatching-related problems using RL approaches is also included.

2.3.1 Foundations of Reinforcement Learning Algorithms

Reinforcement learning solves the sequential decision-making problems which are formalized as the Markov Decision Process (MDP), where the state in the next time step is only decided by the current state and is independent of the previous states (Bellman, 1957). The agent (decision maker) optimizes its policy by choosing the proper action at each time step based on the state by interacting with the environment to maximize the expectation of accumulated rewards in the future (Sutton et al., 1998). The interactions between agent and environment are shown in the Figure 2.6.

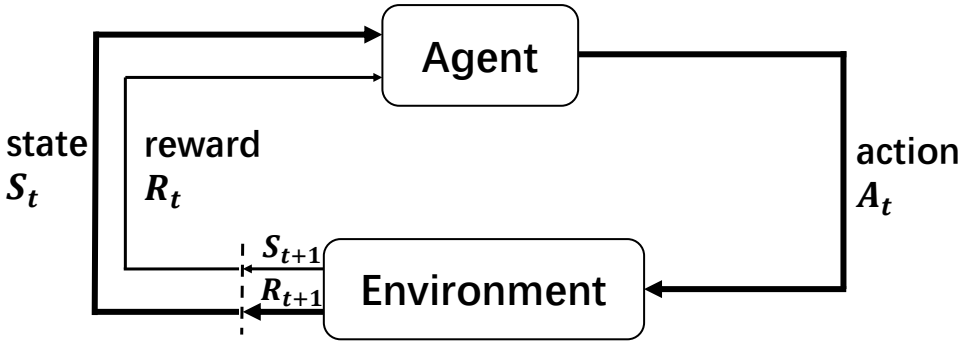


Figure 2.6: Agent-Environment Interaction

The theoretical foundations of reinforcement learning are based on the Bellman equations (Bellman, 1966), which reveal the relations among different state values and state-action values. It is the basic group of equations that is used to solve RL tasks. The Bellman equations could be solved by dynamic programming approaches (Howard, 1960) through a decomposition methodology view. The algorithms of dynamic programming could be further divided into policy iteration and value iteration (Sutton et al., 1998). However, dynamic programming requires the precondition that the probability of each state transition in an environment is known, which is also called model-based methods. In most of the real-world problems, the transition probabilities among states cannot be calculated or are unavailable. The model-free approaches are required in such cases, and one of the most typical algorithms is the Monte-Carlo (MC) method (Hammersley, 2013), which is an unbiased estimation of value functions. Another important model-free method is the temporal-difference (TD) method (Sutton, 1988), which could learn from incomplete trajectories by leveraging the idea of bootstrapping (Mooney et al., 1993).

Notably, the on-policy and off-policy refer to two different RL training paradigms. The on-policy means the policy for data sampling and policy for updating is the same, and off-policy indicates the opposite. Based on such attributes, the TD method could be further classified into SARSA (Sutton, 1988), the on-policy TD, and Q-learning (Watkins and Dayan, 1992), the off-policy TD.

Both the MC and TD find the optimal policy by estimating state-action values, which are called value-based methods. Another family of RL algorithms directly optimizes the policy itself and is called the policy-based method. A policy is defined as the mapping from state to probability

distribution of corresponding actions. The optimal policy is obtained by optimizing the parameters of the policy through policy gradient techniques such as the Reinforce algorithm (Williams, 1992). A common methodology that adopts both value-based and policy-based methods is called the actor-critic framework (Konda and Tsitsiklis, 1999).

Deep reinforcement learning (DRL) refers to the RL methods that are combined with deep learning techniques. With the development of deep learning, several milestone deep neural networks such as the convolutional neural network (CNN) (LeCun et al., 1998) and recurrent neural network (RNN) (Hochreiter, 1997) have been proposed and continuously promote the reinforcement learning field. The mapping from state to state-action values or the probability of actions could be implemented by a deep neural network when using DRL methods. Such manners could solve the issue of high-dimensional and continuous state or action space. In addition, the DRL also makes the policy possess the attribute of generalization.

One outstanding DRL method is deep-Q-network (DQN) (Mnih et al., 2015), which is an off-policy Q-learning method based on deep neural networks. DQN also adopts several mechanisms, such as experience replay and target network. Consequently, several special mechanisms and variants that improved the performance of DQN are proposed such as double DQN (Van Hasselt et al., 2016), prioritized experience replay (Schaul, 2015), dueling network architecture (Wang et al., 2016), multi-step learning (Hester et al., 2018), distributional Q-learning (Bellemare et al., 2017) and the combinations of these mechanisms (Hessel et al., 2018). The policy-based methods of DRL, such as asynchronous actor critic (A3C) (Mnih, 2016), trust region policy optimization (TRPO) (Schulman, 2015), and proximal policy optimization (PPO) (Schulman et al., 2017) have become the main-

stream of traditional DRL approaches.

DRL has achieved remarkable successes in the sequential decision-making field to date (Wang et al., 2022; Landers and Doryab, 2023). It has drawn much research attention since the publication of the milestone works (Silver et al., 2016, 2017; Mnih et al., 2015). Video games are the most commonly used environments for verifying the ability of RL at early stage of its development (Vinyals et al., 2019) and more competitive results has been achieved in various research areas, including robotics (Liu et al., 2021), natural language processing (Wang et al., 2021), computer vision (Tuan et al., 2021; Sun et al., 2021), autonomous driving (Talpaert et al., 2019; Milz et al., 2018; Li et al., 2020a), recommendation systems (Zheng et al., 2018; Chen et al., 2019a), and finance (Cui et al., 2023, 2024; Liu et al., 2020a, 2021b). Notably, its applications in game theory (Silver et al., 2017) and nuclear fusion control (Degrave et al., 2022) even expand the knowledge boundary of humankind.

2.3.2 Reinforcement Learning for Canonical Combinatorial Optimization Problems

Combinatorial optimization (CO) is the key point for numerous important applications such as engineering, transportation, finance, and management science. It has continuously attracted enormous attention from all kinds of research communities for over a century. CO aims at finding a combination of the decision variables from a finite set that optimizes the objective function while satisfying various constraints, where the objective function is usually a value function to be maximized or a loss function to be minimized. The decision space is usually high-dimensional and large-scale,

moreover, the NP-hard nature of most of the CO problems further makes it impossible to enumerate decision variables in a feasible time. Classical approaches for the CO problem could be divided into the exact method, approximation method, and heuristic method (Gutin and Punnen, 2006). An exact method relies on the divide-and-conquer strategy, such as the branch-and-bound strategy, which guarantees optimality but is computationally infeasible for large-scale problems. The approximation method cannot guarantee the optimal solution, but it could finish in polynomial time. The heuristic method is the most widely used approach for CO problems. Even though such approaches lack theoretical support, they are often able to find near-optimal solutions at a relatively fast speed. However, most of such methods heavily rely on the problem-specific heuristics painstakingly developed by domain experts.

With the advances in sequence-to-sequence learning framework (Sutskever et al., 2014) and the rise of computing power in recent decades, using neural network models for CO problems has been revisited Vinyals et al. (2015). The idea is to use high-quality solutions of the problems as labels and to train a neural network that can construct a solution directly from the problem specification in a supervised manner. However, such approaches are undesirable, especially for NP-hard CO problems, because getting high-quality labels is always expensive and even infeasible for some practical problems. It has been demonstrated that the generalization ability of supervised learning neural networks for CO problems is poor even when the optimal labels are provided (Bello et al., 2017). Bello et al. (2017) is a pioneering work that used a learning-based approach to solve CO problems and achieve competitive results by introducing deep reinforcement learning techniques. Specifically, negative tour length is chosen as the reward signal, and the network structure is similar to (Vinyals et al., 2015). Al-

though the computational cost of training is enormous, the results have shown near-optimal performance on TSP instances with up to 100 nodes and optimal solutions on KnapSack instances with up to 200 items. Such an idea is a milestone and makes an inception that utilizing DRL to solve CO problems.

Attention mechanism (Vaswani et al., 2017) has also been explored as the key elements of the network for CO problems (Deudon et al., 2018; Nazari et al., 2018; Kool et al., 2018), which further improve the work (Bello et al., 2017). Deudon et al. (2018) also utilized principal component analysis (PCA) to exploit spatial invariance of the input and batch normalization techniques (Ioffe and Szegedy, 2015) to improve network training. The result showed that such a method, combined with the 2-opt heuristic (Croes, 1958), which is an effective classic heuristic for solving TSP, could outperform the benchmark methods. Nazari et al. (2018) addressed the invariant issue of the policy network and Kool et al. (2018) made further improvement by leveraging the transformer architecture (Vaswani et al., 2017).

Most of the combinatorial optimization problems can be formulated by graphs. For example, the TSP is to find a minimal cost Hamilton circle over an undirected graph. Khalil et al. (2017) is the first study that deployed a learning-based method on graph structure data to solve combinatorial optimization problems. A graph embedding method called Structure2Vec (S2V) is proposed for graph representation. The basic idea of S2V is to represent a node's feature in a graph by aggregating the feature information of its neighbors. A vector containing the information of the graph topology could be obtained after repeating several steps of such a process. The output of the S2V is linked to a deep-Q-network for refining a given

solution perturbatively. The author tested this method on the minimum vertex cover, maximum cut, and traveling salesman problem. The result is better than traditional heuristic methods and shows great generalization performance.

Instead of learning a constructive heuristic that generates the solutions directly, da Costa et al. (2020) focused on improvement (or perturbative) heuristics that could refine a given solution iteratively until reaching a near-optimal solution. In his work, a method that could learn a policy to generate the 2-opt heuristic for TSP is proposed. Moreover, the author modified the initial pointing mechanism, which makes it easy to extend to k-opt operations.

Inspired by the large neighborhood search (LNS) algorithm for vehicle routing problems (Shaw, 1998), where the destroy operator removes a subset node of the current solution and the repair operator is used to generate a permutation of the selected elements and insert them back, Gao et al. (2020) proposed a method to learn the local search heuristics. Motivated by the graph attention network (GAT) mechanism (Veličković et al., 2018) which is an effective method to represent the graph topology by propagating the neighbor node information through the attention mechanism, the author proposed a modified version called Element-wise GAT with Edge-embedding (EGATE) which not only consider the information of nodes but also the arc between the nodes. The method is evaluated by CVRP and CVRPTW problems, and both outperform the classic hand-crafted heuristics and neural combinatorial optimization approach for VRP. Also, the method can tackle large-scale instances (over 400 nodes).

Chen et al. (2020a) is also motivated by the LNS algorithm (Shaw, 1998)

and proposed a similar method called dynamic particle removal (DPR) using Hierarchical Recurrent Graph Convolutional Network (HRGCN). The degree (size and the allocation of the sub-nodes) of the destroy operator is dynamically determined. The HRGCN can be aware of spatial (graph topology) and temporal (embedding in previous iterations) context information. The CVRPTW instances with up to 800 nodes are used to test the performance of the approach method, the results showed that the DPR outperforms the initial LNS heuristics.

Another LNS-based algorithm proposed by Hottung and Tierney (2020) is called neural large neighborhood search (NLNS), which used a similar destroy-repair framework. The method was specifically adapted to support parallel computing, which is one of the contributions of this method. Such an attribute could support two patterns: batch search, which solves a set of instances simultaneously, and the single instance search, which solves only one instance concurrently. The method is demonstrated by the capacitated vehicle routing problem (CVRP) and the split delivery vehicle routing problem (SDVRP). The result showed that such a method outperforms the classic LNS and heuristic methods, and the performance is close to the state-of-the-art method.

NeuRewriter (Chen and Tian, 2019) is another effective proposed approach for iteratively refining a given solution. However, in the NeuRewriter framework, two policies (region select policy and rule select policy) need to be learned, which makes the training process a little bit cumbersome. To tackle this problem, Wu et al. (2021) integrated two policy networks into one by modifying the network structure. Specifically, the author adopted compatibility computation in the model to generate a probability matrix where one of its elements specifies the two nodes to be swapped. To capture

the node position information, sinusoidal positional encoding is introduced to the embedding layer. Moreover, the mechanisms of skip connection (He et al., 2016) are adopted in the model. The result shown by this method outperformed the NeuRewriter.

Similarly, Lu et al. (2019) proposed an iterative improvement method called "Learn to Improve (L2I)" that generates a solution perturbatively. It is noteworthy that such a method outperformed LKH3 (Helsgaun, 2017), the state-of-the-art method of capacitated vehicle routing problems (CVRP). This method firstly designs two classes of operations, namely, improvement operators and perturbation operators, which are used to improve and destroy part of the solution respectively. The network serves as a controller that selects the corresponding heuristics. The effect of historical actions is also taken into consideration in the model. This approach could be considered as a reinforcement learning-based hyper-heuristic, but it is not explicitly pointed out in the paper. The L2I model outperformed the SOTA method on CVRP, which is an impressive result. Basically, most of the perturbative methods outperformed the constructive methods in this field because of their local search nature. However, a higher time cost is required to obtain a high-quality solution compared with the constructive methods. The extremely high searching time issue of work (Lu et al., 2019) has been pointed out in successive research.

Generally, most of the models that are trained on small-size problem instances fail to maintain their performance while evaluating the large-size instances. Training on large instances seems to be a solution, but it brings a higher computational cost for both training and generating data. Moreover, reinforcement learning training usually converges at a quite low speed. Learning-based methods with high training time and low generalization

performance would lose competitiveness compared to traditional heuristic methods.

2.3.3 Reinforcement Learning for Truck Dispatching-Related Optimization Problems

The RL-related approaches for the optimization problems in a container terminal have been explored in recent years (Grafelmann et al., 2023). Zeng et al. (2011) made an early exploration that uses RL to solve the integrated yard crane scheduling and truck dispatching problems in a container terminal. Two agents of both problems act autonomously to optimize corresponding operations. The experimental results show that the RL method could outperform any heuristic approaches. However, the limitation is low generalization ability because of the lack of using deep neural networks to fit the Q value. The research interests for the container truck dispatching based on RL include leveraging the novel network structure (Chen et al., 2021), improving the agent training paradigm (Zhang et al., 2023a), multi-agent reinforcement learning (Hu et al., 2023; Che et al., 2024; Zhou et al., 2024), and RL-assisted evolutionary computation (Chen et al., 2024).

Hyper-heuristic is a kind of methodology that incorporates expert knowledge, which avoids the algorithm search solutions from scratch. The hyper-heuristic and RL complement each other's advantages since RL could help to learn the mapping relation between input states and the corresponding low-level heuristics. Zhang et al. (2022a) explored such a mechanism and proposed a deep reinforcement learning-based hyper-heuristic (DRL-HH) framework for combinatorial optimization problems. The proposed method is verified on the 2D bin-packing problem and the truck dispatch-

ing problem in a container terminal, and the uncertainty factors in reality are fully considered. Ten low-level heuristics are designed based on some quay crane-related priority factors. The network is trained by a double deep Q network framework (Van Hasselt et al., 2016). At each dispatching step, the RL agent is invoked to select a low-level heuristic, and the heuristic will output a specific action (which task to assign). This research is a novel attempt to combine the advantages of reinforcement learning and hyper-heuristic. The competitive performance of the proposed framework is experimentally proven on both a traditional combinatorial optimization problem and a real-world complex optimization problem.

The container truck dispatching problem could be considered as a full-truckload version of the dynamic pickup and delivery problem (DPDP). RL-related methodologies are also investigated from this scope, such as innovating attention blocks in a neural network (Li et al., 2021), multi-agent reinforcement learning (Zong et al., 2022), and hierarchical reinforcement learning (Ma et al., 2021). The RL-based methods for the food delivery problem (Jahanshahi et al., 2022) and the taxi dispatching problem (Liang et al., 2021; Liu et al., 2020b; Qin et al., 2020) are also investigated since they could be derived from the original version of DPDP when satisfying some constraints, such as time windows of customers. According to the taxonomy of truck dispatching problems, food delivery and taxi dispatching problems belong to the job-trigger category. In addition, RL could also be utilized in other industrial fields such as surface mining, which has similar logic to the container truck dispatching problem (Afrapoli et al., 2019; de Carvalho and Dimitrakopoulos, 2021).

2.4 Multi-Objective Optimization

Multi-objective optimization (MOO) has become a crucial topic in the scope of operational research or container terminal operation (Ehrgott and Gandibleux, 2003; Tian et al., 2021). Unlike a single-objective problem, which only one optimal solution is required, MOO aims at exploring a set of solutions with different trade-offs among the objectives. It's common to use Pareto dominance to describe the relation of any two solutions in this set. A solution is said to be Pareto optimal if there is no other solution that could dominate it. The set of all possible Pareto optimal solutions is defined as the Pareto set, and the visualization of the Pareto set on the objective space is called the Pareto front. To find the exact Pareto set is extremely challenging in the MOO field since obtaining a single Pareto optimal solution is quite difficult. Therefore, most of the existing MOO methods are developed to find an approximated Pareto set within a feasible computational time. MOO is significantly concerned among container terminal management since different equipment and problems are involved. The methods towards MOO problems in the container terminal are to find a high-quality approximated Pareto set for different preferences of decision-makers.

In this section, research about traditional MOO methodologies, RL-related approaches for MOO, and MOO problems in container terminals are introduced.

2.4.1 Traditional Multi-Objective Optimization Algorithms

NSGA-II (Deb et al., 2002) is one of the most exemplary approaches during the development of MOO, which is based on the idea of the genetic algorithm (GA). GA starts with an initial population that contains a set of solutions. New populations are generated by conducting genetic operators among parent populations, and elite individuals are retained at each evolving iteration. The fitness of individuals is improved by repeating such a process. For single-objective problems, the fitness is usually defined as the optimization objective value. In NSGA-II, the fitness is evaluated by the non-domination rank and crowding distance. Specifically, the fitness of an individual is better if it is dominated by fewer individuals and has a higher crowding distance towards its neighboring individuals. NSGA-II has high computational efficiency for its non-dominant sorting and maintains great diversity of the solution set. It has been considered as a solid work of the MOO field and a classical MOO baseline algorithm.

Most MOO problems require tremendous search effort even in the case that only one single preference is involved. Therefore, heuristic such as local search operator (Johnson, 1990) are introduced to evolutionary MOO. Pareto local search (PLS) (Paquete et al., 2004) and multi-objective genetic local search (MOGLS) (Jaszkiewicz, 2002a; Ishibuchi and Murata, 1998; Jaszkiewicz, 2002b) are proposed based on a local search mechanism, which simultaneously optimizes several sub-problems with the corresponding weighted scalar objectives. Such a method is considered as a simple and intuitive methodology, which is defined as a decomposition mechanism in successive literature. Decomposition approaches convert MOO to several single-objective optimization problems through scalarization functions

(such as weighted-Tchebycheff or weighted sum functions), then sophisticated approaches for single-objective optimization could be adopted for them separately.

MOEA/D (Zhang and Li, 2007) is another representative algorithm in MOO community. Unlike NSGA-II, which evaluates individuals by their non-dominant relations and crowding distance, MOEA/D calculates the fitness by aggregating objectives to a scalar value. The key idea of the MOEA/D is to decompose a MOO problem into several scalar-valued sub-problems and optimize these single-objective problems in a single run. These sub-problems are defined by different weight vectors. In each iteration, the information of neighborhood sub-problems is used to update the current individual. Similar to the traditional GA, new individuals could be generated by genetic operators such as crossover or mutation. Such a method is considered as a collaborative mechanism that helps to explore the search space more effectively. It is proven that MOEA/D possesses a lower computational complexity than MOGLS and NSGA-II. There are also a series of its variants (Ke et al., 2013; Wang et al., 2015; Zhang et al., 2009) that leveraged the idea of decomposition and collaborative mechanism during the evolution process.

Many-objective optimization problems indicate the cases where more than two objectives are involved, which also verifies the suitability for using a decomposition-based framework (Ishibuchi et al., 2014). Hybrid approaches that comprised both dominance-based and decomposition-based mechanisms are also investigated (Li et al., 2014). Hyper-heuristic (Burke et al., 2013), which is introduced above, could also serve as an option for MOO problems (Maashi et al., 2014, 2015). Notably, some learning-based methods have also flourished for tackling MOO problems (Lin et al., 2022;

Zhang et al., 2022b; Li et al., 2020b; Wu et al., 2019) along with the success of neural combinatorial optimization (Bello et al., 2017). However, most of these approaches only focus on canonical problems such as the multi-objective traveling salesman problem (MOTSP) (Lust and Teghem, 2010) or the multi-objective vehicle routing problem (MOVRP) (Jozefowiez et al., 2008), which lack effort for the cases in real-world applications.

2.4.2 Multi-Objective Optimization in Container Terminals

Multi-objective versions of truck dispatching in container terminals are widely investigated since multiple equipment, such as quay cranes, yard cranes, and trucks are involved in this process, and each equipment has its indicators that are concerned. For our examined problem, QC utilization and truck empty travel distance are considered. Objectives in MOO usually have a strong trade-off property. Choe et al. (2016) proposed an online preference learning approach for truck dispatching of container terminals. A neural network serves as a dispatching policy that outputs the pairwise preference degrees between trucks and tasks. Kim et al. (2013) adopted a noisy multi-objective evolutionary algorithm where an accumulative sampling mechanism is used to alleviate the influence of several environment uncertainties, thus strengthening the model's adaptiveness to variant scenarios. Homayouni and Tang (2013) raised the multi-objective coordinated scheduling problems between trucks and quay cranes. A modified genetic algorithm is adopted, and computational feasibility towards the optimal scheduling solution is analyzed in this work. Hu et al. (2019) considered a bunch of containers as the minimal operational unit and applied a heuristic-adaptive genetic algorithm to minimize task completion time

and the truck's empty traveling distance. Other container terminal-related integrated MOO problems have also been investigated, such as integrated vehicle scheduling and container storage allocation (Dkhil et al., 2017), integrated QC assignment and berth allocation (Prayogo et al., 2022), jointly berth-yard allocation (Liu et al., 2016), and bi-objective berth allocation problem (Zhen and Chang, 2012).

2.4.3 Multi-Objective Reinforcement Learning

Multiple objectives are required for most of the real-world decision-making applications, implicitly or explicitly (Roijers et al., 2013). Multi-objective reinforcement learning (MORL) is a special direction in the RL community to handle diversified objectives. There are two main classifications of MORL, namely single-policy and multi-policy algorithms, depending on whether there is only one preference weight to be considered. In cases of single-policy, a concrete preference over objectives is given, and hence the problem is converted to a single-objective optimization problem. The research concern is usually about exploring suitable scalarization functions for specific problems (Vamplew et al., 2008). Multi-policy algorithms are dedicated to approximate the Pareto front by generating a set of policies. Multi-policy approaches could be further classified into outer loop methods and inner loop methods (Hayes et al., 2022). Similar to the decomposition mechanisms in MOO (Zhang and Li, 2007), outer loop methods obtain the policy set by solving several single-objective problems separately, and the simplest way to conduct outer loop methods is to repeatedly run a single policy algorithm several times with different preference weights (Parisi et al., 2014).

Outer loop methods aim at exploring training paradigms which can accelerate the learning process, such as reusing parameters of neural network (Li et al., 2020b; Zhang et al., 2022b) or learn to initialize the network parameters to make it adapt to various preferences faster (Chen et al., 2019b). In contrast, inner loop methods focus on producing multiple policies directly in a single run, such as weight-conditioned network (Abels et al., 2019), multi-objective fitted Q-iteration (MOFQI) (Castelletti et al., 2011), and Pareto Q-learning (Van Moffaert and Nowé, 2014; Ruiz-Montiel et al., 2017).

2.5 Simulation Methods

In this section, some simulation-based research is discussed. The target of the simulation is still focused on the operations and logic in a container terminal. Then the concept of digital twin, which is highly relevant to solving real-world CO problems is introduced.

2.5.1 Container Terminal Simulation Approaches

Simulation is an effective approach to model the real-world decision-making processes. For CO problems in a container terminal, a simulation approach is suitable for the operations that are sensitive to the dynamic behavior of equipment and could alleviate the complexity of mathematical modeling. A great number of researchers have utilized various simulation approaches for different container terminal operations (Angeloudis and Bell, 2011). Yun and Choi (1999) used SIMPLE++, an object-oriented simulation tool, to develop a container terminal analysis system. Legato and Mazza (2001)

used the Visual SLAM language for discrete event simulation of a berth planning procedure to test some resource allocation problems. Bielli et al. (2006) developed a distributed discrete event-based simulator with Java to model the decision support system in a port. The real-world data are also used to calibrate and validate the simulator. Bin et al. (2008) simulated the truck dispatching process using the AnyLogic software. The blackboard system and the message communication mechanism are used in a multi-agent collaborative system. Yangl et al. (2018) also used the AnyLogic software to simulate the AGV dispatching process in the container terminal to develop the optimization algorithm.

2.5.2 Digital Twin

The digital twin is a kind of mapping from an actual system to its virtual or digital representation (Tao et al., 2022). A physical entity, a virtual entity, and their relations formed a complete digital twin. The objective is to establish a precise, general, and real-time connection between the virtual and real world, thus to offer a simulation environment for the optimization algorithms, product designs, and extreme condition analysis. In this way, a digital twin can describe, diagnose, predict, and optimize the real entities. The concept of the digital twin was first proposed by Grieves (2005) and applied to product life cycle management. The theory of digital twin has been developed for over a decade and has created a great number of successful real-world applications such as supply chain, intelligent manufacturing, intelligent medical diagnosis, and intelligent city (Lin-Yao et al., 2019). One of the great achievements is the City Brain, the first batch of digital twin cities created by Alibaba Group (Zhang et al., 2019). The model is used to conduct research like automatic accident alerting, traffic

light optimization, and traffic flow forecasting. The digital twin technique should be effective and vital for supporting real-world combinatorial optimization problems. Some studies also utilize digital twin techniques in the container terminal optimization (Gao et al., 2023; Zhang et al., 2023b).

2.6 Summary

According to the investigation about these related topics, it is believed that deep reinforcement learning approaches should be suitable to solve the examined online container truck task dispatching optimization problem for the following reasons.

- Traditional rule-based approaches are vulnerable to some uncertain factors such as traffic congestion, layout changes, and equipment breakdown. Well-developed rules (especially for the solutions that are generated from mathematical models) always fail to be feasible on instances with unseen uncertainties. When new uncertainties are considered, reinforcement learning-based methods only need to build uncertainties in the simulation environment and extend the state feature design rather than rebuild new heuristics or low-level heuristics to consider such effects, because the heuristic generation is automated during the RL training.

- Reinforcement learning agent generates the decision-making policies by deep neural networks, which could make it obtain high generalization to different cases compared with heuristic searching-based methods. This could be explained by the theory of the generalization ability of deep neural networks (Neyshabur et al., 2017). Such an at-

tribute makes RL-based approaches suitable for online optimization that requires decisions to be made in real-time.

- The objective is always deterministic in the combinatorial optimization problem realm. Such an objective is an important indicator for designing a reward mechanism that guides the direction for updating the network in reinforcement learning algorithms. In contrast, reward design is hard for some traditional RL application fields like video games because most of their ultimate purposes are oversimplified (win or lose). Reward shaping is always required in such cases, and tuning reward mechanisms in the RL algorithm is painstaking and time-consuming.

- State feature engineering is easier compared with heuristic searching approaches. In RL design, different raw state features or even noise information could be included to construct the state as the input of the network. It benefits from the automatic feature engineering ability of deep neural networks, which handles or filters different state components in the training process. In contrast, noise or insensitive information always reduces the performance of heuristic searching models or makes them fail to converge at the worst, since agents of heuristic searching methods have no intention to select or filter different feature components, and they heavily rely on manual feature engineering.

- Reinforcement learning-based approaches have potential for multi-tasking and integrated optimization in container terminal-related problems. Such a realm belongs to real-world optimization and management problems with more complex logic and higher-level uncertainties. With techniques like hierarchical reinforcement learning and multi-agent reinforcement learning, RL-based approaches are ex-

pected to become a universal framework for container terminal integrated optimization. Since research about RL in the context of container terminals is quite limited, our work could be considered as a trial and set up a solid foundation for this direction.

Chapter 3

Online Container Truck Task Dispatching Problem

This chapter focuses on the description of the proposed online container truck task dispatching problem. Firstly, the process of truck dispatching in a real-world container terminal environment is introduced in detail, including the specific task instructions and several constraints that are set up by the real-life business. The optimization objectives and the main concerns when building the solution to the examined problem are discussed. Based on the comprehension and the investigation of this problem, the main challenges and the research aims are reported. Then, a mathematical formulation of the truck dispatching problem is given, where the cause of the objective value is analyzed. Finally, the development of the simulation environment is introduced, including how the relevant uncertainties, such as traffic congestion, are achieved. Moreover, the concept of Real2Sim in this thesis is also described.

3.1 Container Truck Dispatching Process

This section introduces the concrete operational progress of the examined problem in detail. As briefly introduced in chapter 2, truck dispatching is a dynamic matching between container trucks and transportation tasks. A simplified layout of a container terminal is shown in Figure 3.1. The red arrows represent the truck moving directions allowed at different areas. In this case, each vessel is equipped with 2 loading QCs and 2 unloading QCs.

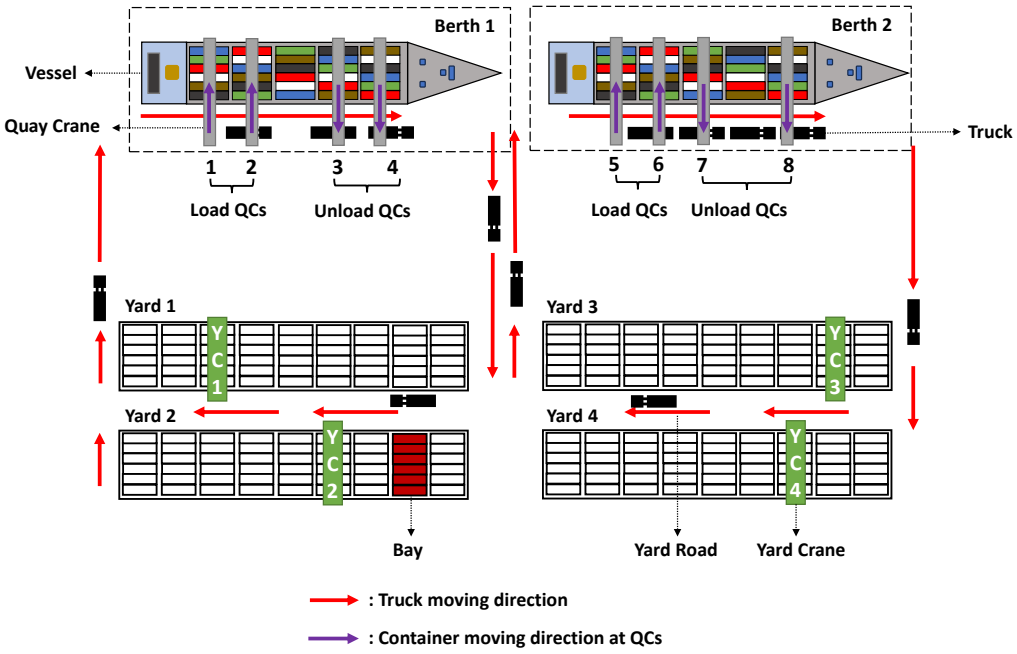


Figure 3.1: A cut-out example of the container terminal layout.

In a container terminal, quay cranes (QCs) and yard cranes (YCs) are two types of closely related equipment in the truck dispatching process. Trucks are responsible for transferring containers either from QCs to YCs or from YCs to QCs, which indicates unloading or loading containers, respectively. The truck's traveling route needs to follow the traffic directions of one-way roads in the container terminal.

Except for the YC and QC, a task is also related to the source and target bay locations. The specific definition of a transportation task in this study

Id	Source	Destination	Type	Source Bay	Destination Bay
1	QC6	32	DISCHARGE	15	12
2	5	QC2	LOAD	20	23

Table 3.1: An example of task instructions.

is indicated by Table 3.1. It contains a unique task ID, the source location, the source bay, the destination location, the destination bay, and the task type. For task 1, the source container at bay 15 dedicated to QC6 is about to transfer to yard 32 at bay 12. The source or destination location is either QC or YC. Task type, which is denoted by the terms LOAD and DISCHARGE in container terminal literature, indicates loading to the vessel or unloading from the vessel, respectively. There are two kinds of QC, namely loading and unloading QC, where loading QCs only serve LOAD tasks and unloading QCs only serve DISCHARGE tasks. The source and destination bay are the stopping places for the truck and cranes besides the yard or vessel. The container loading or unloading operations are only executed when the trucks and the crane both reach the target bay. Generally, the process of executing a task that is assigned to a truck is described as below:

1. Drive to the source bay at the source location.
2. Wait in the queue until the container is loaded to the truck by the crane.
3. Drive to the destination bay at the destination location.
4. Wait in the queue until the container is unloaded from the truck by the crane.
5. Start to execute a new task or return to the park (depot).

The Figure 3.2 describes a specific transportation task that is introduced above from a spatial perspective, where the task is to fetch a container from QC1 and deliver it to Yard B. The red and blue line indicates the empty and loaded traveling path respectively.

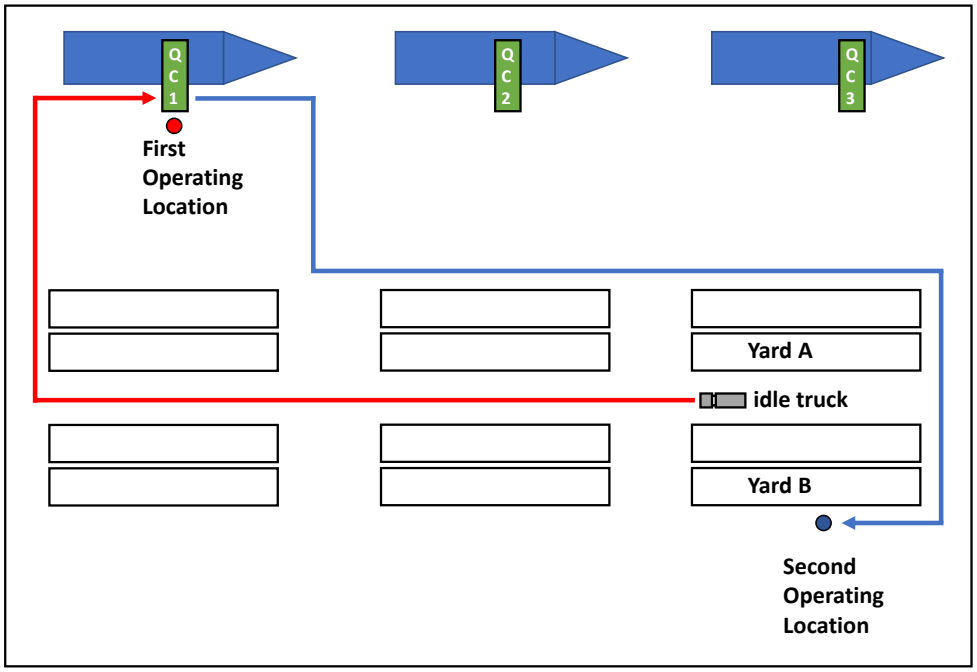


Figure 3.2: Indication of a container truck transportation task.

This study focuses on the vehicle trigger version of the examined problem as described in 1.2.4, which means the decision-making happens at the time step when some truck becomes idle (just finishes its previous task).In addition, a truck only executes and carries one single task at the same time.

In a container terminal, tasks are organized as lists that belong to each QC. In another work, each QC maintains a list of tasks that are related to it. Once a task of a specific QC is dispatched, it needs to be removed from the QC's task list. When a truck finishes its previous task, the central scheduler needs to select a new task and assign it to this idle truck. Such a decision is required to be made within a short period (basically a few seconds). Usually, only the first task in each QC's task list is available to

be selected at some time according to the operational constraints of the container terminal. For instance, if there are n QCs with their non-empty task list, this means the central scheduler has n options for the current truck dispatching decision. In addition, some container terminal operation-related rules set up constraints for the examined problem. For example, the QC operation constraint requires all the QCs to serve the trucks in their task dispatching order, which indicates that sometimes trucks with later dispatched tasks are required to wait in their QC queue if trucks with earlier dispatched tasks do not reach this QC.

QC make-span is defined as the total time that the QC takes to finish all its assigned tasks. In the single objective version of the examined application, the objective is to minimize the summation of all QC's idle time in their QC make-span, since minimizing such objective helps to improve the QC utilization, and thus to increase the throughput of the entire container terminal by shortening the berthing time of vessels. Defining the objective as such better reflects this purpose. Consequently, this objective is formulated based on the perspective of QC operation flow that is presented in Figure 3.3, where $T_1^q, T_2^q, T_3^q, T_4^q$ are the starting times that quay crane q starts to operate the containers of different tasks. $O_1^q, O_2^q, O_3^q, O_4^q$ are their corresponding operating duration, which are indicated by the grey cells. Each QC has two kinds of states, namely, operating and idle. The QC operation flow presents the repeated process in which a QC's state switches between these two states. In multi-objective cases, the truck's empty traveling distance is taken into consideration.

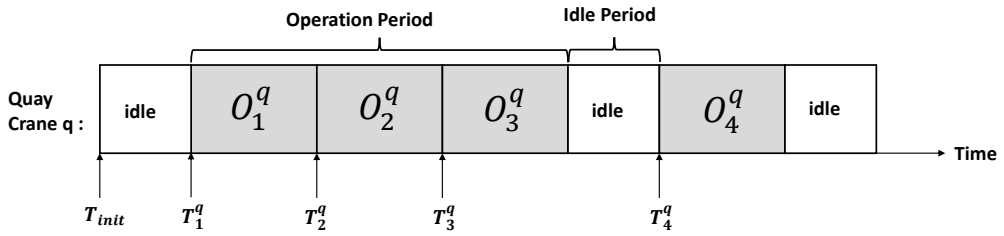


Figure 3.3: An example of QC operation flow.

3.2 Challenges and Motivation

Existing studies for container truck dispatching problems have already taken large efforts on model-driven approaches, which first formulate the target problem by mathematical models, and then adopt various optimization algorithms to obtain the (optimal) solutions (Weerasinghe et al., 2024). There are several mature methodologies for these model-driven COPs. The exact algorithms, usually based on the Branch-and-Bound framework (Tomazella and Nagano, 2020), make full use of the structures of the constraints and objective functions to partition the solution space while ensuring optimality. These approaches could obtain the optimal solution, but at the expense of prohibitive searching cost for large problem instances caused by the exponential time complexity. Alternatively, approximation algorithms, such as heuristic approach, heuristic search, or hyper-heuristic, fail to guarantee the optimality but can generate high-quality solutions in an acceptable computational time (Silver, 2004).

Even though the approximation algorithms are suitable for some classical COPs, once the problem configuration changes slightly, the obtained solutions need to be revised or re-optimized. To adopt the approximation algorithms in a new problem configuration is an open challenge derived from the No Free Lunch (NFL) theorem (Wolpert and Macready, 1997). One main issue of the model-driven method is that it usually concerns determin-

istic variants of the problem, where some strong assumptions are required in the model. Since these assumptions are usually incompatible with the practical scenarios, as a result, the solutions obtained by a model-driven approach may be infeasible to deploy in real-world applications because of the high-level uncertainties (Manikas et al., 2020).

Taking our examined problem as an example, the uncertainty factors come from several aspects, such as the service time of QC or YC operations, the moving speed of equipment, and the degree of yard congestion (Lu and Le, 2014; Liu et al., 2021a). Traditional model-driven methods are usually vulnerable to these uncertainties. The solutions generated in such an offline manner may confront the disturbance caused by these uncertainties in a non-deterministic environment, thus producing the performance drop of the solution, which is accumulative through the entire decision-making process and eventually makes the solution infeasible (Zhang et al., 2022a). As the approach that solves optimization problems with uncertainties, stochastic programming can partially alleviate such issues (Birge and Louveaux, 2011), but it often leads to extremely complex models that tend to be intractable and ineffective for most practical problems.

Apart from the uncertainties and dynamic factors, the optimizations in a real-world container terminal can be far more complex. Take various container terminal operations as an example, several closely related sub-problems are always jointly considered, including quay crane scheduling, berth allocation, yard crane scheduling, container space allocation, and finally the truck task dispatching. Any single operation can influence others, and yet the joint optimization of these sub-problems is more difficult (Kizilay et al., 2020). Therefore, when focusing on one of these problems individually, other sub-problems become another kind of uncertainty. For

example, the estimation of truck queuing time in a yard, which could be considered as a reference factor for truck dispatching decisions, is affected by the concrete yard crane dispatching policy. What's more, random situations such as general disruptions may occur (Rodrigues and Agra, 2022). These factors may place a burden on obtaining high-quality solutions for some common approaches.

Apart from the dynamic and uncertainty factors above, the real-life container terminal authorities may be faced with trade-off purposes in their daily management. For instance, equipment such as quay cranes needs to be made best used as effectively as possible during the peak time, thus to shorten the vessels' operation make-span. During spare time, other operational costs such as energy consumption, labor cost, and equipment maintenance need more consideration. In addition, situations in reality are not always these two extreme cases but fall into the intermediate states. Therefore, it is required that the trade-off purposes should be dynamically concerted at different preference levels, which raises more challenges to the management methodology. From another perspective, such trade-off purposes could also be considered as an uncertainty about user preferences.

In the multi-objective version of the examined problem, the truck empty distance (as depicted by the red line in Figure 3.2) is taken into consideration. More specifically, the average empty traveling distance per task is designed as the second objective to be minimized. Minimizing the QC idle time is considered as a primary focus for a truck dispatching strategy in a container terminal because the throughput productivity of a port heavily relies on how continually QCs execute loading and unloading operations, which is directly related to the port companies' economic benefit and business competitiveness. Reduction of truck empty travel distance could

also save operational costs by decreasing fuel consumption and equipment maintenance costs. Although its profit is inferior to minimizing the QC idle time, minimizing the empty travel distance is also related to reducing the carbon emission and air pollution, which has already drawn considerable concerns by society and governments (Mansouri et al., 2015).

To tackle the multi-objective truck dispatching problem, existing methods aim at seeking a set of Pareto optimal policies, where no policy can outweigh in all objectives. Solving time for approximating the Pareto optimal set would exponentially increase compared with optimizing a single objective policy, since the user preference over different objectives is usually unknown in most cases (Roijers et al., 2013). The algorithms that are extensively adopted are based on multi-objective evolutionary computation, which are able to obtain a set of non-dominant solutions in a single run through evolving a population of candidate solutions. Two well-known approaches for MOO problems are MOEA/D (Zhang and Li, 2007) and NSGA-II (Deb et al., 2002). Numerous MOO algorithm variants are based on these two algorithms (Ke et al., 2013; Li et al., 2014; Wang et al., 2015). However, most of these mature approaches for MOO are designed for canonical CO problems, which cannot satisfy the special requirements of the examined truck dispatching problem. Another drawback of the evolutionary MOO approaches is revealed by their finite number of solutions. More specifically, the quality and diversity of the solution set could only be guaranteed by an adequate population size and insufferable long-term evolving generation. Otherwise, users may fail to choose the most suitable policy for a specific scenario.

To sum up, considering the aforementioned challenges of the examined problem, our designed methodology is required to be robust to various

uncertainties. What's more, the algorithm should give a specific decision in a short response time since it is a real-time decision-making optimization problem. For the MOO version problem, the obtained solution set should possess enough policy diversity, which guarantees that the most proper preference weight for users is available. Therefore, a deep reinforcement learning-based methodology is proposed for the examined problem. In this process, the real-life complexities and uncertainties are fully concerned. For MOO dispatching problem, a novel methodology called preference agile multi-objective optimization (PAMOO) is designed that make the model use a uniform and custom-designed neural network to generate dispatching decision with arbitrary preferred trade-offs rather than generate a finite set of dispatching policies and allows users to dynamically decide the trade-off according to real-time situation and interactive adjust preference easily. Finally, mechanisms that introduce the prior expert knowledge to augment the model, which help to accelerate the agent training, are explored for both single and multi-objective optimizations.

3.3 Mathematical Formulation

As described above, quay cranes (QCs) and yard cranes (YCs) are two most representative equipment in a container terminal. A standard transportation task can be defined by its first and second operating nodes and its index in the corresponding QC task list. A task can be either from QC to YC (import containers) or from YC to QC (export containers). Upon the arrival of a vessel, the set of containers to be loaded and unloaded can be determined in advance, and a fixed number of QCs are assigned to service the vessel. Each QC is attached to a list of tasks that need to

be dispatched and transported in a predefined sequence. The dispatching process is repeatedly triggered by a task request from an idle truck. The automated dispatch module (i.e., dispatch algorithm) firstly evaluates all candidate tasks available at the active QCs and assigns the most suitable task to the target truck, and then the truck will visit the first and second nodes of the assigned task to complete the transportation. A waiting time is imposed if there are other preceding trucks at any node because both QCs and YCs can only handle unit tasks each time. Upon completion of the task, the truck becomes idle again and triggers a new request until all tasks are completed. Truck dispatching requires careful consideration to avoid interruptions of QC operations (QC waiting for the incoming trucks). A bi-objective formulation that simultaneously minimizes both the total idle time of QCs and the total truck empty travel distance is adopted in this section. The objective functions and relative constraints are achieved through the simulation that is introduced in the next section.

For the convenience of reading, the notations used in this section are summarized and classified in Table 3.2. The problem is mathematically formulated as follows. Denote Q and Y the set of QCs and YCs of the container terminal and d the parking lot of the truck fleet (i.e., depot), $d \notin Q \cup Y$. All trucks are initialized in the depot at the beginning of the simulation. Let $N = Q \cup Y \cup \{d\}$. The term working instruction refers to a unit-sized transportation task that each truck can maximally handle at any time. The set of all tasks is denoted as U. W^q indicates the task list of q^{th} QC, $W^q \subseteq U$ and w_i^q refers to the i^{th} task in W^q, $w_i^q \in W^q$. Let f_i^q and s_i^q be the first and second operating locations of the task w_i^q and $f_i^q, s_i^q \in Q \cup Y$. Denote V the truck fleet, $|Q| \leq |V| \leq |U|$, and each truck v is involved in the dispatching process. The term O_i^q and L_i^q indicates the QC's operation duration and truck waiting time in the QC queue for task w_i^q, $O_i^q > 0, L_i^q \geq 0$.

Notation	Description
Given Information	
Q	The set of quay cranes
Y	The set of yard cranes
N	The set of all positions in a container terminal
U	The set of all transportation tasks
W^q	The task list of q^{th} quay crane
w_i^q	The i^{th} task of q^{th} quay crane
f_i^q	The first operation location of task w_i^q
s_i^q	The second operation location of task w_i^q
V	The set of all dispatchable trucks
$\delta(x, y)$	The distance between location x and y, $x, y \in N$
T_{init}	The start time of the simulation.
Decision Variables	
$\alpha(w_i^q, v)$	A binary variable to indicate whether task w_i^q is assigned to truck v or not.
Auxiliary and Internal Variables	
O_i^q	The quay crane operation time of task w_i^q
L_i^q	Truck waiting time in the QC queue for task w_i^q
D_i^q	The time step when task w_i^q is dispatched
T_i^q	The time duration the task w_i^q takes to arrive its QC
$\tau(x, y)$	The travel time between location x and y, $x, y \in N$
$\beta(t, v)$	The location of truck v at time step t
$\lambda(w_i^q)$	The yard crane service time of task w_i^q
$\phi(w_i^q)$	The loading or unloading type of task w_i^q
$z(w_i^q, w_{i'}^{q'})$	The variable to indicate whether task w_i^q and $w_{i'}^{q'}$ are executed by the same truck in a consecutive order.
T_{end}	The end time of the simulation.
Objectives	
Function (3.5)	Total QC idle time
Function (3.6)	Total truck empty traveling distance
Constraints	
Constraint (3.7)	Distance between any two locations is positive
Constraint (3.8)	Traveling time between any two locations is positive
Constraint (3.9)	Each task in U is dispatched to only one truck
Constraint (3.10)	The first task in each QC task list is dispatched at the beginning of the simulation
Constraint (3.11)	Tasks in each QC list are required to be dispatched in a pre-defined order
Constraint (3.12)	Each QC is required to operate tasks in the same order when they are dispatched
Constraint (3.13)	It defines the waiting time of truck in QC queue when the quay crane is busy
Constraint (3.14)	It ensures the QC operates a the task immediately at the arrival of the truck when the QC is idle
Constraint (3.15)	It guarantees trucks execute tasks continuously

Table 3.2: Notations used in the problem formulation.

Denote D_i^q the time step when task w_i^q is dispatched. T_{init} is the start time of the simulation and T_{end} refers to the end time once all the task in U are finished. Several functions that help to model some details during the truck dispatching process are established as below. The functions $\tau(x, y)$ and $\delta(x, y)$ are used to query the travel time and distance from location x to y separately, where $x, y \in N$. Expression $\beta(t, v)$ return the current position of the truck v at time step t and $\beta(t, v) = d$ if $t = T_{init}$. The formula $\lambda(w_i^q)$ calculates the yard crane service time (including waiting in the yard queue) for a given task w_i^q.

Equation 3.1 decides if a specific task w_i^q is assigned to truck v.

$$\alpha(w_i^q, v) = \begin{cases} 1, & w_i^q \text{ is assigned to } v \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Expression (3.2) indicates the loading or unloading type of a given task w_i^q.

$$\phi(w_i^q) = \begin{cases} 1, & w_i^q \text{ is loading task} \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Expression (3.3) indicates that the tasks w_i^q and $w_{i'}^{q'}$ are executed by the same truck in a consecutive order.

$$z(w_i^q, w_{i'}^{q'}) = \begin{cases} 1, & w_i^q \text{ and } w_{i'}^{q'} \text{ are executed by the same} \\ & \text{truck in a consecutive order} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Let T_i^q (3.4) compute the time duration that a truck takes to arrive at the QC in task w_i^q.

$$T_i^q = [\tau(f_i^q, s_i^q) + \lambda(w_i^q)]\phi(w_i^q) + \sum_{v \in V} \tau(\beta(D_i^q, v), f_i^q)\alpha(w_i^q, v) \tag{3.4}$$

Formally, we can have the following problem formulation:

$$\min \sum_{q=1}^{|Q|} \sum_{i=2}^{|W^q|} \max(D_i^q + T_i^q - D_{i-1}^q - T_{i-1}^q - L_{i-1}^q - O_{i-1}^q, 0) + \sum_{q=1}^{|Q|} T_1^q \tag{3.5}$$

$$\min \sum_{q=1}^{|Q|} \sum_{i=1}^{|W^q|} \sum_{v \in V} \delta(\beta(D_i^q, v), f_i^q)\alpha(w_i^q, v) \tag{3.6}$$

s.t

$$\delta(x, y) > 0 \quad \forall x \neq y, \quad x, y \in N \tag{3.7}$$

$$\tau(x, y) > 0 \quad \forall x \neq y, \quad x, y \in N \tag{3.8}$$

$$\sum_{v \in V} \alpha(w_i^q, v) = 1 \quad \forall w_i^q \in U \tag{3.9}$$

$$D_1^q = T_{init} \quad \forall q \in [1, |Q|] \tag{3.10}$$

$$D_i^q \geq D_{i-1}^q \quad \forall q \in [1, |Q|] \tag{3.11}$$

$$D_i^q + T_i^q + L_i^q \geq D_{i-1}^q + T_{i-1}^q + L_{i-q}^q + O_{i-1}^q \quad \forall q \in [1, |Q|] \tag{3.12}$$

$$\begin{aligned} &L_i^q = D_{i-1}^q + T_{i-1}^q + L_{i-q}^q + O_{i-1}^q - D_i^q - T_i^q \\ &if \quad D_i^q + T_i^q < D_{i-1}^q + T_{i-1}^q + L_{i-q}^q + O_{i-1}^q \end{aligned} \tag{3.13}$$

$$L_i^q = 0 \quad if \quad D_i^q + T_i^q \geq D_{i-1}^q + T_{i-1}^q + L_{i-q}^q + O_{i-1}^q \tag{3.14}$$

$$D_i^q \alpha(w_i^q, v) = z(w_i^q, w_{i'}^{q'})(D_i^q + T_i^q + L_i^q + O_i^q) \quad \forall v \in V \quad \forall w_i^q, w_{i'}^{q'} \in U \tag{3.15}$$

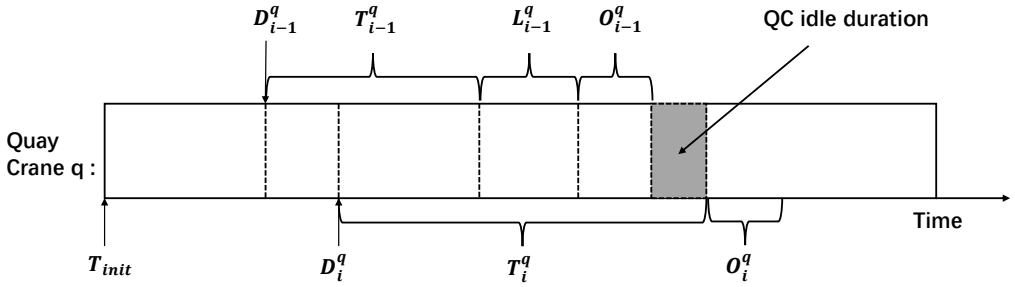


Figure 3.4: An example that truck with task w_i^q arrives at QC after prior task's completion. The QC idle duration is caused in this case.

The objective function (3.5) is the total QC idle time to be minimized. There are some special time steps during a task's duration, and the rela-

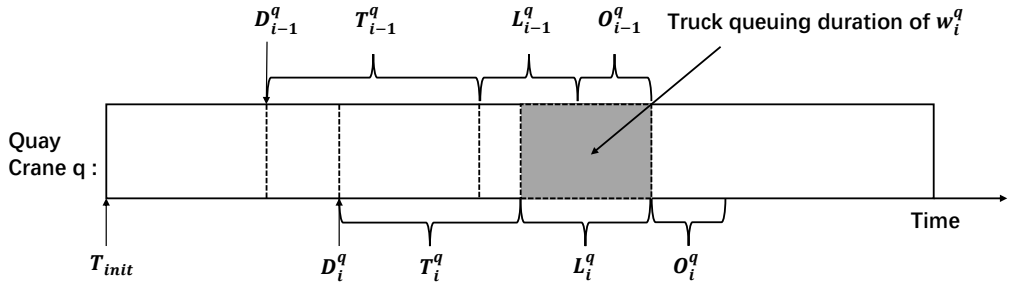


Figure 3.5: An example that truck with task w_i^q arrives at QC before prior task's completion. The truck queuing duration is caused in this case.

tive positions of these time steps in a timeline between two adjacent tasks are used to represent the objective, which is further illustrated in detail according to Figure 3.4 and 3.5. Function (3.6) is the second objective, the total truck empty travel distance to be minimized. Expressions (3.7) and (3.8) indicate that the traveling distance and time are positive for any two different nodes in N. Equation (3.9) guarantees each task in U is dispatched to exactly one truck. Equation (3.10) limits the first task of each QC to be dispatched at the beginning of the simulation. Constraint (3.11) restricts each task to be dispatched in a pre-defined order. Expression (3.12) makes sure that each QC must operate tasks in the exact same order as dispatching, and a task starts to be operated by QC only when its prior task is completed. Equation (3.13) computes the time that a truck with task w_i^q needs to wait in the target QC queue when it arrives at the target QC before the completion of the previous task w_{i-1}^q. Equation (3.14) ensures that QC operates the task w_i^q immediately if the truck arrives after the completion of the prior task w_{i-1}^q (no queuing duration). Constraint (3.15) guarantees each truck executes tasks continuously.

Notably, the locations of the nodes in $Q \cup Y$ are non-stationary because of the movements of both QC and YC. Hence, the function $\delta(x, y)$ for any nodes is variant and real-time computed. Due to the stochastic service time of cranes, O_i^q is also a variable that includes the uncertainty factors

in real container terminals. Furthermore, the terms L_i^q and $\lambda(w_i^q)$ are also unpredictable because of the high uncertainty level of the environment and could only be confirmed after the occurrence of the related events. Most of the research neglects these details, which makes the examined problem more challenging and distinct from their works.

Notably, the proposed formulation of the examined problem is completely self-defined and dedicated to the specific structure of the container truck task dispatching problem. Basically, the examined problem is similar to the problems in (Chen et al., 2016; Zhang et al., 2022a; Chen et al., 2022). Rather than those mathematical programming models that could be directly used for computing the solutions, our proposed problem formulation is only treated as a detailed specification of the examined problem and the instructions for building the simulation, since the proposed methodology is not a model-driven solution.

3.4 Development of Simulation

The implementation of the container terminal simulation is introduced in this section. The simulation system is a full-scale environment for the Ningbo-Zhoushan port MeiShan terminal. The details like road distance, import or export yard distribution, and moving speed of the QC and YC follow the details in the real-world cases. The environment is implemented through a discrete event simulation (DES) approach where the agent (container, trucks) in the environment repeatedly travels through a pre-defined event sequence. For example, a container's life cycle is initiated in an operational location (vessel or yard), waiting to be transferred by a truck, and then exits from the event sequence. A truck's logic flow is to repeatedly

transfer containers from the first operational location to the second location until all the tasks in a single instance are finished. Such a simulation process is built by AnyLogic software, and the discrete-event logic flow is described by the Figure 3.6.

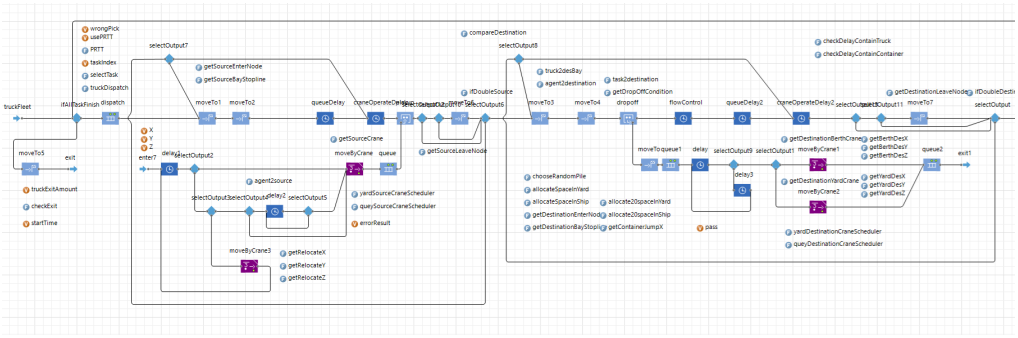


Figure 3.6: Container terminal simulation logic.

The major principle of implementing a discrete event simulation (DES) could be represented by a timeline which contains all events about to happen in a simulation run. When an event happens, it may cause more events that will happen at some times in the future. For example, dispatching a task to a truck will cause the event of loading and unloading operations of cranes in the future. The newly generated events will be inserted into this timeline in real time. The simulation time moves forward by jumping from event to event. The events are the key factors that change the state of the system, such as the QC queue length or truck state. When all event happens, the simulation result is computed spontaneously (system end state). The Fig 3.7 illustrates the basic principle of DES that is introduced above.

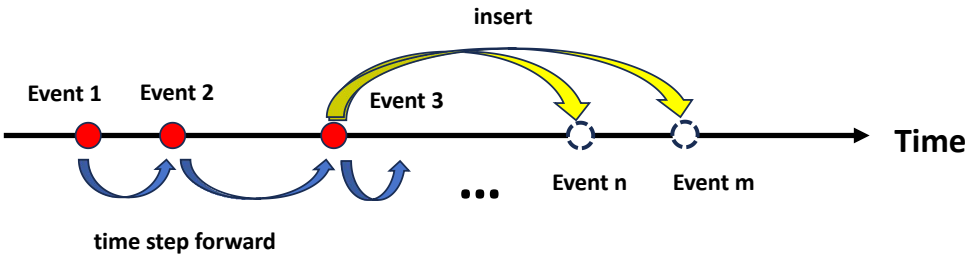


Figure 3.7: Illustration of discrete event simulation

The implemented simulation could support at most 7 vessel berths, 28 QCs, and 110 yards at the same time. Such a setting depends on the implementation stage of the simulation, and increasing the upper bounds requires further engineering work. These settings are adequate to simulate a real-world container terminal and could support users in generating abundant problem instances with different configurations, such as total task number, truck number, QC number, and yard distribution. Our research used different instance configurations for training and testing, which are introduced in section 4.4.

AnyLogic provides 2d and 3d views of visualizations for users to better and conveniently observe and analyze the implemented simulation. Figure 3.8 presents a screenshot of the 2d view for the simulation system. There are also some user-defined dynamic indicators for each berth, such as QC utilization, queuing trucks, and remaining task number, which in order to better observe the environment at a specific time step. The simulation tries to reproduce details in real-world scenarios, such as the truck queuing at berth or yard operations, as much as possible. Figure 3.9 presents several screenshots of various components in the simulation environment.

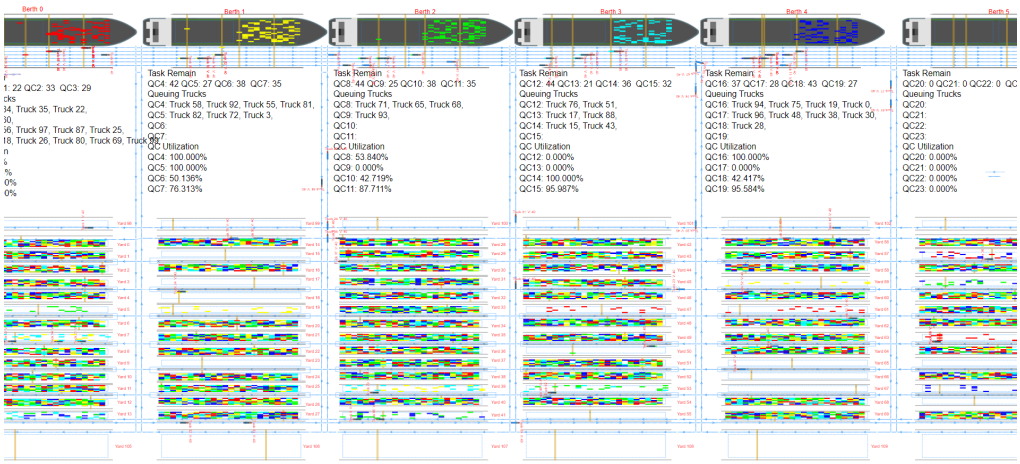
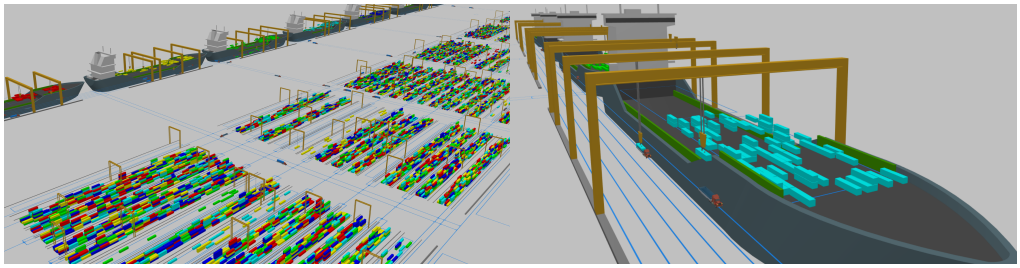
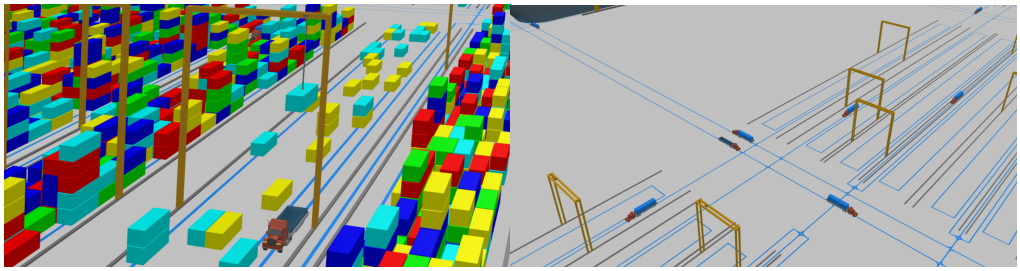


Figure 3.8: 2D View of the Simulation System.



(a) Container Terminal (b) Berth



(c) Yard (d) Intersection

Figure 3.9: Screenshots of different components in container terminal simulation.

The Figure 3.10 indicates the framework hierarchy of the simulation model through an engineering development perspective. The components, like truck, road, container, crane, and ship at the bottom, are built-in units in the AnyLogic software. Then the components of a port, such as berth, yard, road network, and truck fleet, could be implemented through the combination of these basic built-in units. According to the data monitoring in the simulation process, the user could be able to make decisions for different equipment such as trucks, QCs, and YCs. In this thesis, trucks are the target schedulable agents. Some interfaces for other operation policies are reserved, such as container relocation, container space allocation, and yard crane scheduling. The specific policy for these operations could be specified by the user to initialize the simulation execution. The entire simulation process could be encapsulated into a Java package. A socket communication mechanism is built for data transmission between the processes of the simulation environment and the training algorithm.

The framework is convenient for training the reinforcement learning agent. With the help of this simulation environment and framework, the problem scenarios and instance configurations could be conveniently designed. It set up a solid foundation for algorithm training and related experiments. The proposed methodology that makes the reinforcement learning agent flexible adapts to different scenarios. The result demonstrates the great performance and generalization ability against the benchmark algorithms and the heuristic approach.

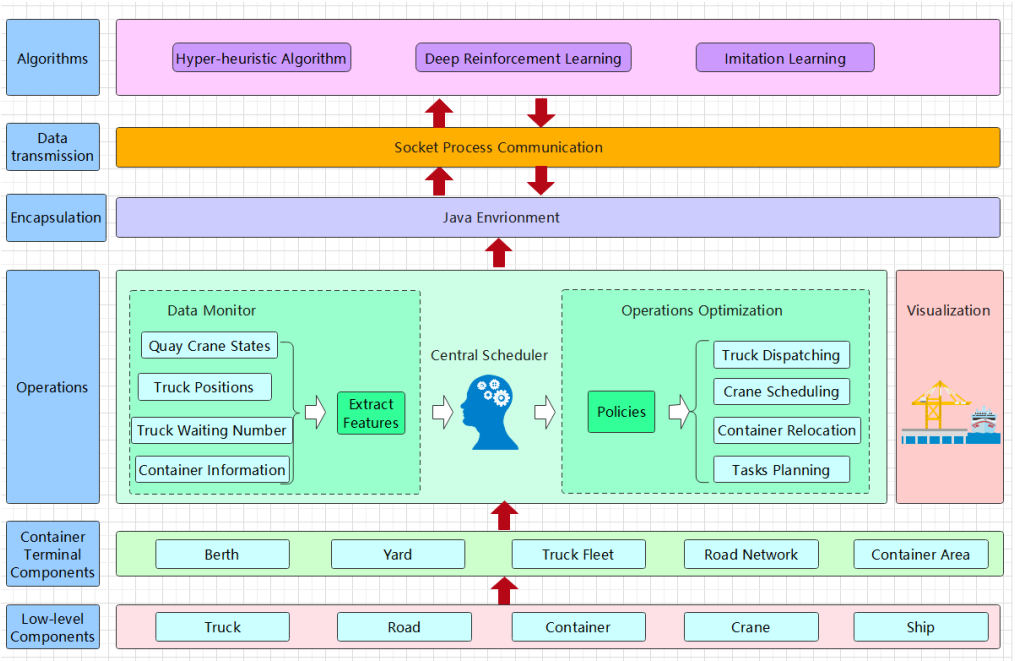


Figure 3.10: Container Terminal Simulation Framework.

The parameter settings and the implementation of some uncertainties of the simulation are introduced here. The purpose of doing so is to reproduce the high-fidelity details of reality as much as possible. The environment simulates the full life cycle of the examined truck dispatching process, together with other relevant operations such as quay crane scheduling or container relocation, and the uncertain factors are also taken into consideration. Considering real-world cases, it is not reasonable to treat truck speed as constant even if it is traveling on an empty road. The truck's

traveling speed varies all the time and is affected by its states, such as load weight or traffic congestion. In this developed simulation environment, the truck's traveling speed on the road is defined as a random distribution. Similarly, the service time of the quay cranes or yard cranes is also defined as given distributions in consideration of that the crane loading and unloading operations are executed manually and the proficiency variation exists.

The yard congestion degree is considered as another uncertainty factor. Usually, a three-lane road is shared by two neighboring yards in yard block areas. The middle lane is reserved for trucks traveling through, the other two lanes are used for trucks' temporary docking and waiting for the crane services. Under specific cases, this three-lane road is quite congested because of the truck collision avoidance. Therefore, the truck's speed is set inversely proportional to the number of trucks on this road when it enters this three-lane road. Some specific simulation parameters are given in Table 3.3.

Some uncertainty factors that are introduced above are formulated here. Denote C_t^v as the traveling speed of truck v at time step t and $g_t^{y,y'}$ as the total number of trucks in the queues of two adjacent yards y and y' that share a three-lane road at time step t. Equations (3.16) and (3.17) are truck load state indicators at time step t that affect truck traveling speed.

$$\beta(v,t) = \begin{cases} 1, & v \text{ is on-load state at time step } t \\ 0, & v \text{ is empty-load state at time step } t \end{cases} \tag{3.16}$$

$$\phi(v,t) = \begin{cases} 1, & v \text{ is on its target yard road of the} \\ & \text{current task at time step } t \\ 0, & \text{otherwise} \end{cases} \tag{3.17}$$

Equation (3.18) represents the truck speed in different areas of the container terminal, in kilometers per hour.

$$C_t^v = \begin{cases} c_1 \sim f_1, & \beta(v,t) = 1, \phi(v,t) = 0 \\ c_2 \sim f_2, & \beta(v,t) = 0, \phi(v,t) = 0 \\ 10, & g_t^{y,y'} > 10, \phi(v,t) = 1 \\ 30 - 2g_t^{y,y'}, & 1 \leq g_t^{y,y'} \leq 10, \phi(v,t) = 1 \end{cases} \tag{3.18}$$

where c_1 and c_2 are sampled from their probability distributions for truck speed with empty or loaded state, f_1 and f_2, respectively. f_1 and f_2 are obtained through analysis from the container terminal operation in reality.

Table 3.3: Simulation Parameters.

Parameters Name	Value Range (unit)
Loaded Truck Speed at Road	[30, 40] (km/h)
Empty Truck Speed at Road	[40, 50] (km/h)
Truck Speed at Yard	[10, 30] (km/h)
Crane Bridge Speed	[0.5, 1.5] (m/s)
Crane Trolley Speed	[1.0, 2.0] (m/s)
Crane Hoist Speed	[2.0, 3.0] (m/s)

To evaluate the accuracy of the implemented simulation, two kinds of testing approaches, namely, unit testing and integrated testing, are provided. The unit testing is to ensure that each low-level component in the simulation environment is able to represent the real-world entities. Some testing cases include the traveling time between two fixed locations, the moving

time of the quay cranes or yard cranes, and the service time of the cranes to handle containers. The real value of this time could be obtained through the investigation of a real-world container terminal, such as analysis of the business data and time measurement of relevant recorded videos. The unit testing is used to confirm some low-level simulation parameters, such as equipment speeds. The integrated testing is to guarantee that the results (objective values or some interval variables) of the entire simulation process are able to match the real-world scenarios. The integrated testing is achieved by generating the problem instances where all tasks and the dispatching policy completely follow real-world scenarios according to the historical data. The result gap between the simulation and real-world case is used to adjust the simulation logic and is guaranteed to be limited within an acceptance level.

Based on previous work, a Real2Sim framework is developed with the aim of model training environment for the container terminal. In general, training a policy directly in a real-world environment is infeasible and unsafe. However, based on the efforts made for the simulation, events that occur in reality could be accelerated. Abundant aspects of the real-world container terminal daily operations, such as physical entities, operational logic, and various uncertainty factors, are modeled (see Figure 3.11). To this end, a high-fidelity and effective training environment that is customized for the examined truck dispatching optimization problem is implemented. What's more, some representative scenarios are concluded and utilized to configure the problem instances and initialize the simulation in the training process based on the historical data analysis in the real container terminal. In addition, a sophisticated heuristic policy, which is verified in reality, is considered as a kind of prior expert knowledge and is used to augment the reinforcement learning agent (described in Sec 6). Therefore, the devel-

oped training environment has made full use of the entities, logic, scenarios, historical data, and human experience in real container terminals, and thus, the practical application potential of a well-trained agent through this Real2Sim framework is promoted.

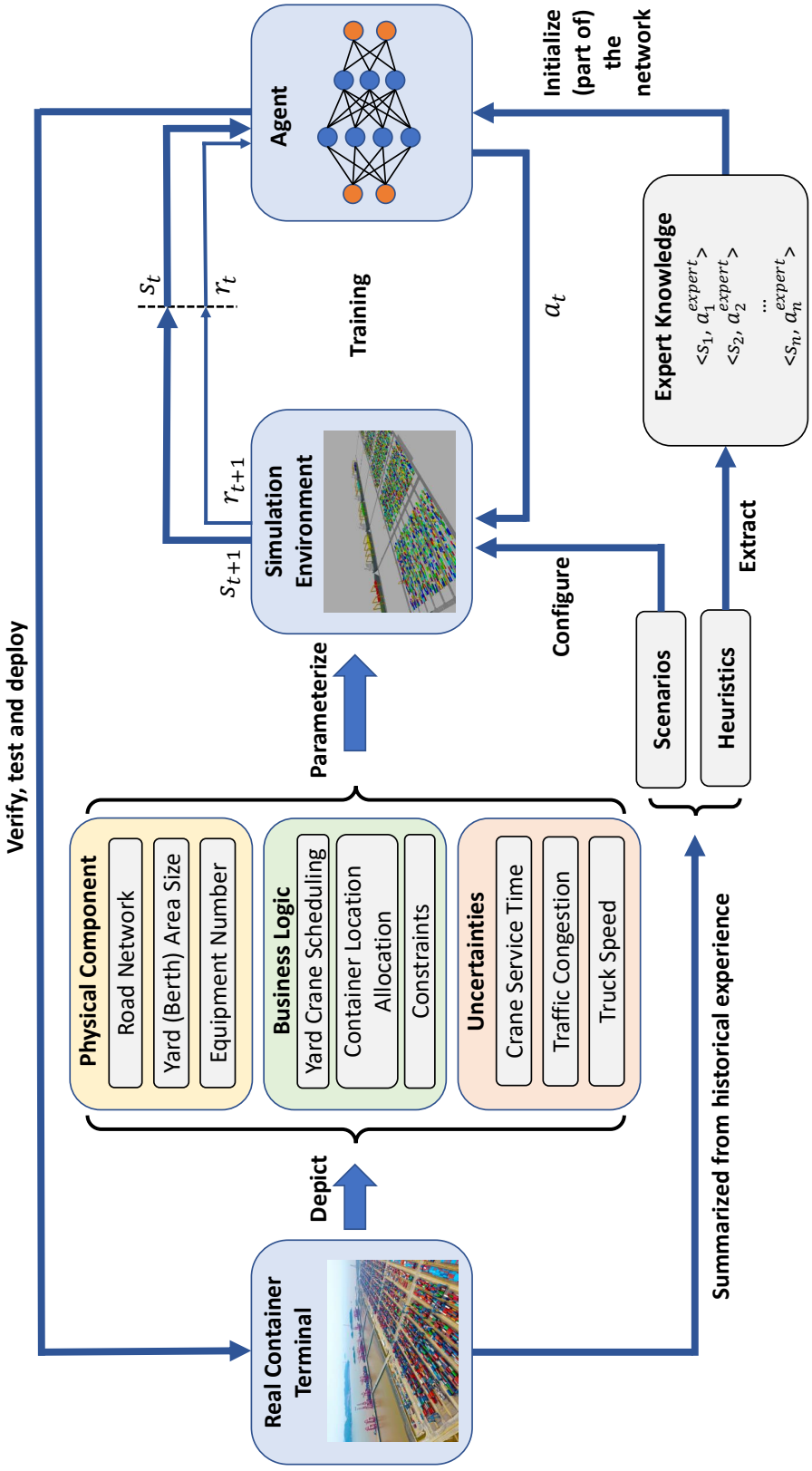


Figure 3.11: The Real2Sim framework of the proposed reinforcement learning environment.

Chapter 4

Methodology for Single-Objective Dispatching

This chapter focuses on the implementation of the methodology for the single-objective container truck task dispatching problem. As the first stage of the research in this thesis, this chapter only considers the single objective of quay crane idle time, which decides the utilization of quay cranes. Firstly, the definition of the Markov decision process (MDP) and how the examined truck dispatching problem is formulated as an MDP is introduced. Several improvements, such as the network structure and learning scheme, are implemented to make the RL algorithm better customized to the examined problem. In this work, the agent's adaptation to the multi-scenarios and the real-world uncertainties is the main concern, and the experimental results of the performance towards such issues are analyzed. In addition, several business insights that are yielded from such methodology are discussed in the end.

4.1 Truck Dispatching Problem as an MDP

Reinforcement Learning (RL) is one branch of the machine learning community. The RL agent (decision-maker) is trained through interactions with the environment (problem) in a trial-and-error manner. Basically, the agent's actions are obtained by a parameterized policy, which is usually represented by a deep neural network, then it executes the given action and receives the feedback (numerical rewards) from the environment. Repeating such a process, the policy of the agent could be improved eventually. RL aims at solving sequential decision-making problems which need to be formalized as the Markov Decision Process (MDP) (Sutton and Barto, 1998).

Generally, an MDP is represented by a tuple $M = (S, A, R, P, \gamma)$, where S indicates the state set. A represents the set of available actions that could be selected by the RL agent. R is the set of immediate rewards. The expression $r(s, a)$ denotes the specific reward obtained after executing the action a at state s and transferring to state s. P stands for the state transition probability, which can be represented as $P(s'|s, a)$, indicating the probability of transition from state s to state s' after executing action a. $\gamma \in [0, 1]$ is called the decay factor that is used to balance the current and future return.

The purpose of an RL agent acting in an MDP is to generate a policy π_θ that could make decisions a on specific state s, which is a mapping between states and available actions in another word. Solving MDP is equivalent to exploring an optimal policy that maximizes the future discounted accumulated rewards.

A finite MDP with a discrete time step is adopted to formulate the examined truck task dispatching problem. The interval between two adjacent time steps is dynamic and relies on the time step that a certain truck just finishes its previous task (i.e. becomes available for dispatching). The details of our formulation are as follows:

4.1.1 State

When a truck just finishes its previous task, a new task will be decided by the dispatcher (RL agent). The observation at this time step is based on the information related to the target truck and each task (the first task of each quay crane). The following information is defined as the state of the MDP model at time step t. The state design fully considers the spatial and temporal features by investigating the examined problem in depth through the implemented Real2Sim system.

- $\mathbf{RE}_t$: The number of remain task of each quay crane.
- $\mathbf{DQ}_t$: The distance between the target truck and each quay crane.
- $\mathbf{DF}_t$: The distance between the target truck and the first operation location of each task. (For loading tasks, the first operation location is a QC. For unloading tasks, the first location is a YC).
- $\mathbf{DS}_t$: The traveling distance of the second task of each QC. If there's no second task remaining, this feature is set to zero.
- $\mathbf{TW}_t$: The number of trucks that are currently working for each QC.
- $\mathbf{TH}_t$: The number of trucks which is currently heading to each QC.
- $\mathbf{QL}_t$: The queue length of each QC.

- $\mathbf{QY}_t$: The queue length of YCs that are dedicated to the first task of each QC.
- **TY** : The type of each QC (loading or unloading), which is represented by one-hot codes.

Denote Q'_t the set of indices of active QCs (the remaining tasks exist) at time step t, which is dynamic in an episode. The shape of **TY** is $|Q'_t| \times 2$ and the shapes of other 8 features are all $|Q'_t| \times 1$. Therefore, the state s_t at time step t for the target truck v is defined as a $|Q'_t| \times 10$ matrix where each row of the matrix is denoted as $[RE_t^q, DQ_t^q, DF_t^q, DS_t^q, TW_t^q, TH_t^q, QL_t^q, QY_t^q, TY^q]$, $q \in Q'_t$.

4.1.2 Actions

Based on a given state s_t, the action space at time step t for the target truck v is defined as $\boldsymbol{a_t} = \{q \mid RE_t^q > 0, q \in Q'_t\}$, where q is the index of the selected active QC. Once q is chosen, the first remaining task of q^{th} QC is assigned to truck v. Notably, the action space at each time step t is not a constant size since the task list could be completed at some time step t.

4.1.3 Reward

In this work, reward design is similar to the principle in the work of Zhang et al. (2022a), which associates QC idle time to each task. The reward related to QC idle time is set to be $-T_1^q$ for task w_1^q and $-\max(D_i^q + T_i^q - D_{i-1}^q - T_{i-1}^q - L_{i-1}^q - O_{i-1}^q, 0)$ for the remaining tasks ($i > 1$), which are the components of the objective function 3.5. Since this is a minimization

problem, negative signs are added to each term. It is noted that the resulting QC idle time $-T_1^q$ or $-\max(D_i^q + T_i^q - D_{i-1}^q - T_{i-1}^q - L_{i-1}^q - O_{i-1}^q, 0)$ are not immediate signals in an episode. Therefore, the reward for each action is computed retrospectively at the end of the episode. For a traditional RL task (usually the video game), there' a high-level purpose such as surviving or winning the game. These high-level purposes usually cannot be described by mathematical language, and achieving them heavily relies on the reward design. There are always issues when the reward design is thoughtless. For example, it is intuitive that setting some small bonus to guide the agent is reasonable, but sometimes the agent may be myopic to these bonuses and ignore the final large reward. In an optimization task, the high-level purpose is to optimize an objective function that is quantifiable. Therefore, it provides a simple way to design the reward, which is to set the reward equivalent to the optimization objective. Such designing is straightforward and makes the training more stable. Actually, the mechanism like reward shaping is also implemented in this project, but the result is not as good as the proposed reward design since too many over-designed extra bonuses sometimes mislead the optimization direction.

4.1.4 State Transition

The state transition between s_t and s_{t+1} is governed by the function: $s_{t+1} = F(s_t, a_t, u_t)$. The transition relies on the action a_t and uncertainties u_t of the environment. In this work, the transitions for $\mathbf{DQ}_t, \mathbf{DF}_t, \mathbf{DS}_t, \mathbf{TW}_t$, $\mathbf{TH}_t, \mathbf{QL}_t, \mathbf{QY}_t$ could be affected by u_t which is introduced in section 3.3 and 3.4. For example, the operation time of each YC and QC is fluctuating, and the traveling speeds of trucks are also not constant, these factors result in the non-deterministic state transitions. Each component of u_t is

implemented in the simulation. There are also some features whose state transition is deterministic, such as the transitions for $\mathbf{RE}_t$ are directly determined by the agent's specific action. In this work, the state transition between two time steps is automatically executed by the simulation environment. The Figure 4.1 explains the state transitions of the examined problem, where the total time step H equals to the total number of tasks in an episode.

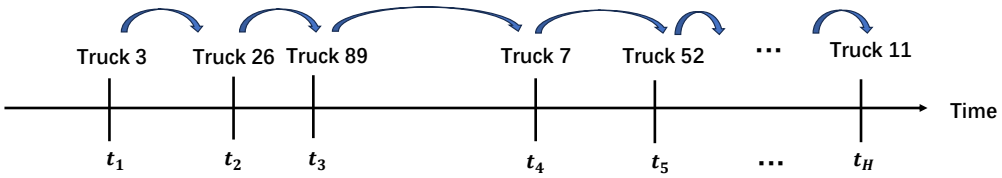


Figure 4.1: An example of the state transitions of the truck dispatching problem.

4.2 Network Structure

The policy network in this work is depicted in Figure 4.2. Given the state described in section 4.1.1, the policy network takes these feature vectors of each QC as the inputs and then outputs a probability distribution which represents the indices of QCs (their first task) to be selected. Firstly, the feature vectors of each QC are fed into a three-layer long short-term memory (LSTM). Next, the hidden states of each LSTM step are fed into an attention layer. Lastly, a probability distribution of actions is obtained after a softmax layer.

Since the number of active QCs is changing in an episode, the policy network treats state vectors as a dynamic set with spatially connected elements (target truck, QCs, tasks, etc.). The spatial-temporal connection between the target truck and each candidate QC is embedded into the state design,

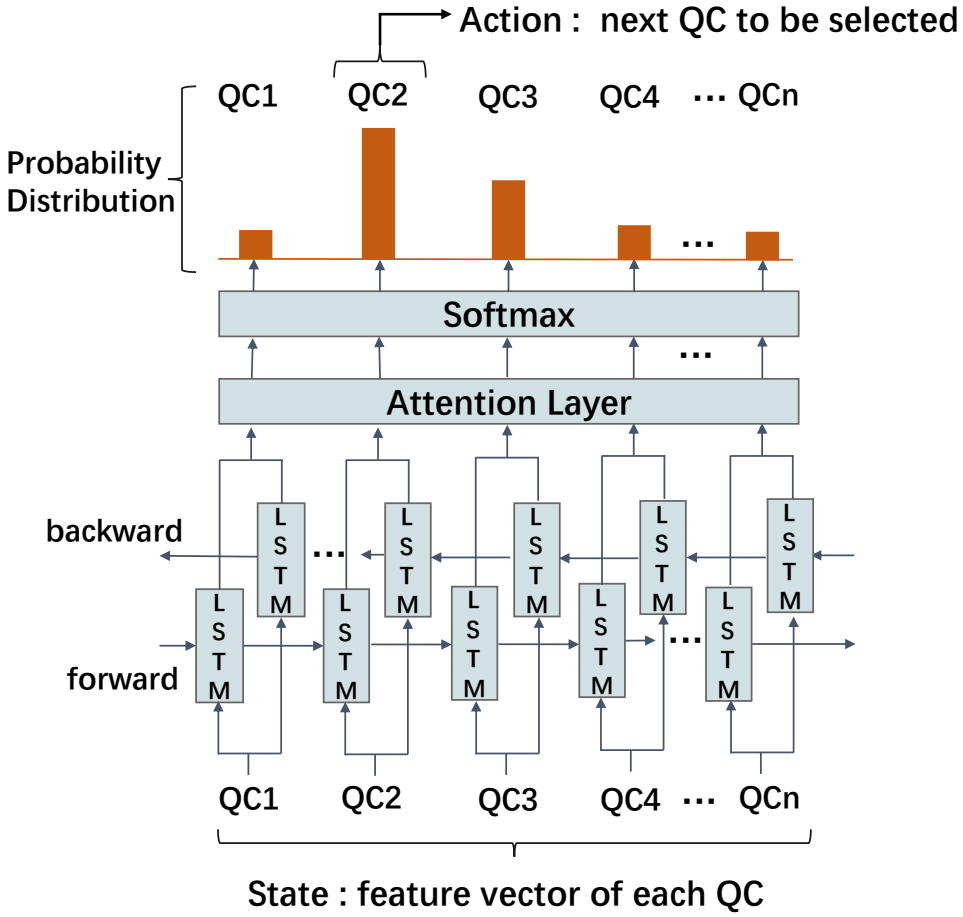


Figure 4.2: Network Structure of the Policy Network.

which is introduced in section 4.1.1. The input data follows the increasing order of QC indexes, which is also the fixed positions in the container terminal (from west to east). Once the task list in a QC is all dispatched, the feature vector of such QC will be eliminated from the input. The purpose of adopting the LSTM layer is to make the network capable of handling the dynamic size of the input candidate QC.

The bidirectional structure ensures the model's awareness of the information of the entire QC sequence at each step. With the help of the attention layer, the network can focus on some specific QCs. For example, if a QC is about to be idle soon, the network may focus on such QC since the idle time of the QC may occur, which gives an impact on the objective value. Moreover, the capability for handling input sequences with different sizes enhances the competitiveness in general scenarios towards structures like a fully connected network. A classic attention mechanism is adopted in this model, which follows the work of Bahdanau et al. (2014).

Actually, the proposed network is quite a classic structure that was once used at the early stage of the natural language processing (NLP) field. The data structure of the examined truck dispatching problem is similar to the NLP since the QC feature vectors in the input sequence are more like the word embeddings in a sentence. This is also the motivation for choosing such a network structure at the early stage of this research.

4.3 Algorithms

Model-free RL aims at obtaining a decision-making policy through a trial-and-error manner by interacting with the environment (Nachum et al.,

2017). In general, model-free RL can be classified into two categories: value-based and policy-based methods. Compared with value-based methods, a policy-based approach directly optimizes the policy (network) that explicitly maps states to the probability distribution of actions in the training process. For complex problems, a policy-based method shows more competitiveness since it can handle the exploration/exploitation trade-offs by training a stochastic policy. In this work, a standard policy-based method, namely, Proximal Policy Optimization (PPO) (Schulman et al., 2017) (see Algorithm 1) is adopted to tackle the examined dispatching problem.

A widely used variant of policy-based methods is called policy gradient with baseline, where a baseline value is used to decrease the variance of gradient estimation while keeping the bias unchanged. An effective baseline design could make the training process more stable, thus accelerating the agent convergence. Usually, such a mechanism is based on the actor-critic framework, which requires training two networks simultaneously. In this work, to simplify such a mechanism, a shared baseline is adopted, which is defined by $b = \frac{\bar{R}}{N}$ where N is the number of episodes within an iteration and $\bar{R}$ is the total return. As introduced in section 4.1.3, the instant reward r_t is unavailable at time step t and is computed at the end of the episode. Recalling the reward design, dispatching a truck to an idle QC would result in a small instant reward. Consequently, the agent is inclined to choose the QC with long queue length rather than the QC with high urgent level, thus, misleading the optimization direction and even running counter to the ultimate purpose. To tackle such an issue, the policy network is trained with a sparse reward manner. The advantage function for each state-action pair is defined as $R^n - b$, where R^n is the sum of rewards collected in an episode. As a result, the advantage function better reflects the quality of

the actions and maintains the gradient updating direction the same as the optimization purpose.

Algorithm 1: PPO for truck dispatching optimization

Input: number of iterations I, steps per episode T, collect N episodes per iteration, update M times per iteration, clipping rate ϵ

Initialize: a differentiable truck dispatch policy parameterization $\pi(a|s, \theta_{old})$;

$\theta = \theta_{old}$;

for *i=1 : I* **do**

  $\bar{R} = 0$;

  Randomly select a problem instance B_i;

  for *n=1 : N* **do**

    Collect an episode $s_0, a_0, r_1, ..., s_{T-1}, a_{T-1}, r_T$ from B_i, following $\pi(\cdot|\cdot, \theta)$;

    Assign each $r_t(a_{t-1}, s_{t-1})$ based on the reward design, $\forall t \in [1, T]$;

    $R^n = \sum_{j=1}^{T} r_j$;

    $\bar{R} = \bar{R} + R^n$;

  end

  $b = \frac{\bar{R}}{N}$;

  Compute advantage function $\mathcal{A}^\theta(s_t^n, a_t^n) = R^n - b, \forall t \in [1, T], \forall n \in [1, N]$;

  for *m=1 : M* **do**

    Compute probability ratios $PR_t^n(\theta) = \frac{\pi(a_t^n|s_t^n, \theta)}{\pi(a_t^n|s_t^n, \theta_{old})}, \forall t \in [1, T], \forall n \in [1, N]$;

    $\mathcal{A}^\theta(s_t^n, a_t^n) = min[PR_t^n(\theta)\mathcal{A}^\theta(s_t^n, a_t^n), clip(PR_t^n(\theta), 1-\epsilon, 1+\epsilon)\mathcal{A}^\theta(s_t^n, a_t^n)]$;

    $\nabla\mathcal{L} = \frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\mathcal{A}^\theta(s_t^n, a_t^n)\nabla \log P_\theta(a_t^n|s_t^n)$;

    $\theta = Adam(\nabla\mathcal{L}, \theta)$;

  end

  $\theta_{old} = \theta$

end

Specifically, the number of iterations I is set to 2000, which ensures the policy will converge. The number of collections per iteration N is 10, which is adequate to estimate the baseline b. M is set to 10, which is an empirical value. clipping rate ϵ is 0.2, which is a common value for PPO settings. Steps per episode T equals the number of tasks in a problem instance. According to the problem instance design, T equals 400 or 800, which will

be introduced in Section 4.4. The optimizer is chosen as the classic Adam (Kingma and Ba, 2014). The algorithm is implemented by Python and Pytorch. The GPU used in the following experiments is a single NVIDIA A100. Each iteration takes about one minute in such an algorithm setting. According to the training process, the policy will sometimes fluctuate sharply but always converge and outperform the benchmark methods.

4.4 Problem Instance and Scenario Design

A set of problem instances with various configurations is designed for RL agent training and testing. In this work, a configuration is defined as a kind of dispatching scenario with specific truck fleet sizes, number of tasks, number of QCs, container storage mode, and yard distribution. Each configuration contains infinite problem instances with different random seeds, which makes the result of an instance reproducible. The configurations are designed based on the scenarios in the real-world container terminal operation environment. In the environment, each vessel is equipped with two loading and two unloading QCs. 60, 80, and 100 are chosen as the number of trucks in the designed problem configuration. Such settings are decided by fully considering the optimization space issue since too many or too few trucks may lead to obvious optimal policies and hence leave no room for optimization. Two stacking modes are defined: `mixed` stacking means the loading or unloading containers are allowed to share the same yard, while `separated` stacking indicates the opposite. Two types of yard distribution are designed as well: `centralized` shows that the loading or unloading containers are stored in no more than two yards, while `distributed` allows to store in more than two yards. The concrete configurations for training

instances are presented in Table 4.1. Notably, one configuration represents a family of problem instances with similar initial conditions caused by uncertainties in the environment. A static problem instance could be created by a fixed configuration and a random seed.

Table 4.1: Training Instance Configurations.

Name	Configuration Items				
	Number of Tasks	# of Trucks	# of QCs	Stack Mode	Yard Distribution
Config 1	400	70	16	Mixed	Centralized
Config 2	400	60	16	Mixed	Distributed
Config 3	800	80	16	Mixed	Centralized
Config 4	800	80	20	Separated	Distributed
Config 5	400	70	20	Separated	Centralized
Config 6	400	60	20	Separated	Distributed

4.5 Benchmarks

A manually crafted heuristic (Chen et al., 2016) is selected as one of the benchmarks for the examined problem. A dispatching rule, which is obtained by genetic programming (GP) (Chen et al., 2020b) is also used as another benchmark since GP is a widely used approach for online optimization problems. The proposed method in this work is also compared with another RL-based approach, deep reinforcement learning-based hyper-heuristic (DRL-HH) (Zhang et al., 2022a) since it is a novel and competitive approach for decision-making optimization. For the sake of fairness, both GP and DRL-HH are trained through the same simulation environment and the same problem instances. Besides these three methods, several commonly considered heuristic dispatching rules (Tao and Qiu, 2015; Nguyen and Kim, 2012; Chen et al., 2022), which are based on various priority factors, are also included in the comparison of this work. All benchmark details are presented in Table 4.2.

Table 4.2: Benchmark Algorithms for Comparison.

Name	Description
Random Dispatch	Dispatch the task randomly.
Dedicated Dispatch	Each QC is only served by fixed group of trucks.
Shortest Queue Length	Dispatch the task with the shortest QC queue length.
Most Task Remain	Dispatch the task with the most QC task remaining.
Shortest Distance	Dispatch the task with the shortest traveling distance.
Most Urgent	Dispatch the task with minimum current QC supply.
Genetic Programming	Dispatching rules generated by genetic programming approach.
Manual Heuristic	A sophisticated heuristic used in real port.
DRL-HH	Policies trained by DRL-based hyper-heuristic approach.

4.6 Experiments and Result Analysis

4.6.1 Comparative Results with Benchmarks

In the training and testing process, instances of 6 different configurations that are introduced in Table 4.1 are used simultaneously. The metric of evaluation is the summation of each QC's total idle periods (in seconds). Since the randomness factors exist in the environment, the result of the same configuration may vary even if the dispatching policy is fixed. Therefore, for each test configuration, the result is based on the average value of 100 runs with different initial random seeds. The numeric comparison results are presented in Table 4.3. The standard deviations of the proposed DRL method for these six configurations are 1463, 955, 1560, 1869, 971, and 1284, which demonstrates that the results in Table 4.3 are stable.

Table 4.3: The performance of the proposed method in comparison with different benchmark algorithms (measured by QC idle time in seconds)

Method	Config 1	Config 2	Config 3	Config 4	Config 5	Config 6	Average
Random Dispatch	37477	26432	50474	57708	32493	37356	40323
Dedicated Dispatch	38289	29306	49036	54367	34795	40236	41005
Shortest Queue Length	38148	30291	58062	62312	36391	40244	44241
Most Task Remain	45207	30847	58486	68198	36477	44630	47308
Shortest Distance	46039	39589	119403	101193	50964	50074	67877
Most Urgent	29963	25336	40226	50397	27512	30982	34069
Manual Heuristic	29470	24586	38915	48199	26693	31051	33152
Genetic Programming	28862	24086	39016	46812	25512	29122	32235
DRL-HH	28438	22840	37903	45210	25732	28784	31484
Ours	**27714**	**21140**	**36930**	**42125**	**24940**	**26428**	**29880**

The results indicate the proposed DRL approach could outperform all other 9 benchmark methods for these six configurations. Six simple dispatching rules, Random Dispatch, Dedicated Dispatch, Shortest Queue Length, Most Task Remain, Shortest Distance, and Most Urgent, fail to obtain competitive results since most of them greedily consider only one single prioritized factor, which cannot evaluate the state as a whole. Dedicated Dispatch could cause a large empty traveling distance, as introduced in chapter 2.2. Intuitively, queue length should be equivalent to the urgency level of the QC. However, the performance of the Shortest Queue Length heuristic is far worse than Most Urgent, since the urgency level of QC should depend on the supply in a period, which is also closely related to the task distance. It is obvious that most of the simple heuristic rules are even worth than Random Dispatch, which is caused by their greedy nature.

Manual heuristic outperforms these six simple dispatching heuristics because it fully considers the spatial-temporal related observations, and the supply-demand mechanism better reflects the urgency level of QCs. Dispatching rules obtained by the genetic programming (GP) method further improve the results of a manually designed heuristic due to its ability to construct a sophisticated dispatching rule based on all combinations of the observed states and repeatedly refine the policy in an evolutionary manner. DRL-HH slightly outperformed GP-based methods since DRL-HH introduced prior expert knowledge, and its RL agent aims at learning the scheduling of several low-level, properly designed heuristics, which further enhance the ability of the method. Although DRL-HH demonstrated its competitive performance, its ability may still heavily rely on the performance of low-level heuristics. In addition, the action space of DRL-HH is too restrictive since these low-level heuristics usually fail to cover the entire available action space, which may cause the model to miss out on

some promising solution regions that deserve to be explored. In contrast, our proposed method is more effective since it can directly search for specific actions. In general, our proposed DRL approach achieved a significant improvement (5.46%) towards DRL-HH. For the subsequent experiments, the manual heuristic (Chen et al., 2016), GP-based approach (Chen et al., 2020b), and DRL-HH (Zhang et al., 2022a) are selected as three representative benchmarks for further experiments.

4.6.2 Generalization Performance of Proposed DRL Approach

To demonstrate the generalization performance of our proposed DRL approach, several customized instances with various configurations are designed for testing in this experiment. These instances are more abundant, and most of them are unavailable during the training process. The design of testing configurations follows the controlled variables principle. In another word, if one of the variables (number of trucks, tasks, QCs, stacking mode, and yard distribution) changes, the others keep the same. All other configurations are derived from the base configuration (Config 2: 60 trucks, 400 tasks, 16 QCs, mixed storage, and distributed yard). The results are reported in Table 4.4.

Similarly, our proposed DRL-based method outperforms the benchmarks, manually designed heuristic, GP, and DRL-HH approaches in all testing configurations. The results demonstrated the generalization performance of the proposed DRL approach to handle various real-world scenarios.

Several phenomena through experimental results deserve discussion. Over-

Table 4.4: The performance of the proposed method in comparison with the manual heuristic and a DRL-HH method under different instance configurations (measured by QC idle time in seconds)

Configuration	Heuristic Method (Benchmark)	Obj value / Imp (%) against benchmark		
		GP	DRL-HH	Ours
60 Trucks	24390	23269 4.6%	22641 7.2%	21428 **12.1**%
80 Trucks	16716	15980 4.4%	15946 4.6%	15032 **10.1**%
100 Trucks	13658	13423 1.7%	13246 3.0%	12891 **5.6**%
400 Tasks	24390	23269 4.6%	22641 7.2%	21428 **12.1**%
800 Tasks	46628	44583 4.4%	43425 6.9%	41032 **12.0**%
1200 Tasks	67735	65201 3.7%	64280 5.1%	64107 **5.4**%
8 QCs	7786	7541 3.1%	7412 4.8%	7298 **6.3**%
12 QCs	13972	13523 3.2%	13012 6.9%	12987 **7.0**%
16 QCs	24390	23269 4.6%	22641 7.2%	21428 **12.1**%
20 QCs	30850	28912 6.3%	28592 7.3%	26331 **14.6**%
24 QCs	40145	38123 5.0%	37816 5.8%	35489 **11.6**%
28 QCs	49700	48325 2.8%	47431 4.6%	45092 **9.3**%
Mixed Storage	24390	23269 4.6%	21132 7.2%	21428 **12.1**%
Separate Storage	22597	21332 5.6%	21376 5.4%	19802 **12.4**%
Centralized Yard	26920	26230 2.6%	25816 4.1%	25163 **6.5**%
Distributed Yard	24390	23269 4.6%	22641 7.2%	21428 **12.1**%

all, the results of the dispatching algorithms heavily rely on the truck-QC ratio. A higher truck-QC ratio makes QC's operation more continuous, thus reducing the QC idle time. Such a phenomenon is confirmed by the positive correlation between QC number and the objective value, and the negative correlation between truck number and the objective value, respectively. Except for objective value, the improvements over the benchmarks are also negatively correlated to truck numbers. This is caused by the optimization space issue. The examined truck dispatching problem is sensitive to the number of trucks and a high truck-QC ratio over limits the space for improvement of the proposed approach.

The improvement ratio of the proposed method over benchmarks drops significantly from the configuration of 800 tasks to 1200 tasks. This should be illustrated from the perspective of the QC operation logic. The QC idle time is reduced by keeping all QCs as busy as possible by dispatching the proper tasks to trucks. However, configurations with more task lists are vulnerable to a relatively poor dispatching policy. For example, unreasonable decisions are more likely to be made in such a configuration, therefore, the idle time caused by such decisions could be accumulated. Such a case is also one of the experiments where the performance of DRL-HH is closest to our method, and DRL-HH presents fairly robustness towards changing of task number.

The improvement ratio in mixed and separated storage mode configurations is basically the same. The objective value in mixed storage is significantly higher than the separate storage mode, since relatively high yard congestion levels and more container relocation operations tend to occur in cases of mixed storage mode. Our method performed better in cases of distributed yards than in cases of centralized yards in terms of both improvement and

objective value, which could be explained by the same reason.

To further verify the generalization ability of our proposed DRL-based method, a dispatching policy is trained only by the base configuration (Config 2: 60 trucks, 400 tasks, 16 QCs, mixed storage, and distributed yard) and is tested by a series of unseen configurations. The results demonstrate that the policy trained by one single configuration is able to be generalized to these unseen problem configurations while maintaining fair performance. DRL-HH is excluded in this experiment since its input size is fixed, which makes it unable to handle instances with various QC numbers.

Table 4.5 reports the generalization performance of the proposed DRL method and GP in comparison with the benchmark. The benchmark in Table 4.5 represents the work (Chen et al., 2016). Overall, the policy trained by the base configuration outperformed both GP and the benchmark method in all unseen test configurations. It is obvious that our method shows slighter over-fitting effects towards GP. This benefits from the practical state design, which makes the RL agent sensitive to environmental changes in the unseen configurations, and also benefits from the design of the network structure, which could cope with various QC numbers.

Several key factors of truck dispatching policy are revealed in the state feature design. Features like the number of trucks working for each QC, the number of trucks heading to each QC, and the queue length of each QC indicate the QC's urgency level of truck supply in a future period. Features like the truck traveling distance to QC could estimate the traveling time for trucks to reach target QCs. QC type is considered as crucial information since the supply forms for loading and unloading QC vary a lot. Target

Table 4.5: The generalization performance of the proposed method in comparison with the manual heuristic (measured by QC idle time in seconds)

Configuration	Heuristic Method (Benchmark)	Obj value / Imp (%) against benchmark	
		GP	Ours
8 QCs	7786	7568 2.8%	7493 **3.8**%
12 QCs	13972	13228 5.3%	12237 **12.4**%
16 QCs (base)	24390	21032 13.8%	20793 **14.7**%
20 QCs	30850	28903 6.3%	27983 **9.3**%
24 QCs	40145	38625 3.8%	37896 **5.6**%
60 trucks (base)	24390	21032 13.8%	20793 **14.7**%
70 trucks	20762	19035 8.3%	18663 **10.1**%
80 trucks	16716	16035 4.1%	15438 **7.6**%
90 trucks	14632	14345 2.0%	13312 **9.0**%
100 trucks	13658	13758 -0.7%	13209 **3.3**%

yard queue length could be used to infer yard congestion level at some time steps. The second task's QC-yard distance exposes some key information about QC's future tasks, which affects the sensitivity to truck supply of QCs. The ability of the network structure to handle various QC numbers also helps the RL agent get rid of the influence of inactive QCs (the empty QCs).

4.6.3 Comparative Results with Offline Solution

We selected one fixed instance (instance with fixed random seed) for each configuration and solved them in an offline manner. In this case, the result could be reproducible for a fixed dispatching decision sequence. Consequently, the problem of each configuration could be considered as a static problem instance. Therefore, the high-quality solutions could be obtained with the help of local search-based heuristics. The purpose of doing so is to further explore the gap in the solutions obtained by our proposed DRL approach towards their estimated upper bounds. In this experiment, a multi-start local search heuristic with a two-element swap operator is adopted to solve static problems. The optimization time for each instance is set to 72 hours. The results obtained in such a way are viewed as the upper bounds of each specific instance. The result is reported in Table 4.6

Table 4.6: Comparative results of the proposed method with estimated upper bound (measured by QC idle time in seconds)

	Estimated Upper Bound	Our Method	Gap(%)
Instance 1	25832	27714	7.29%
Instance 2	19932	21140	6.06%
Instance 3	34241	36930	7.85%
Instance 4	39895	42125	5.59%
Instance 5	22839	24940	9.2%
Instance 6	24221	26428	9.11%

Basically, the average gap between the solutions obtained by our proposed DRL approach and the estimated upper bounds is around 7.38%. The results further demonstrate the competitiveness and potential of our proposed approach.

4.6.4 Managerial Insights

Except for the competitiveness and adaptiveness performance of the proposed approach in multi-scenario cases, several useful insights for the container terminal operational management are also revealed in these experiments. Firstly, it is recognized that the optimization space is highly related to this study. The concept of optimization space could be intuitively defined as the improvement of any sophisticated dispatching algorithms compared with a base dispatching policy (such as the percentage in Tables 4.4 and 4.5). For a given QC number, if a great number of trucks are deployed, the optimization space is tiny since each QC is over-supplied, and the performance of any dispatching policies would be close to random dispatching. On the contrary, if the truck numbers are too few, dispatching policies are easier to have a great improvement ratio, but the objective value (QC idle time) would be worse because the trucks are not adequate. Therefore, the Truck-QC ratio could be viewed as a sensible indicator since it not only affects QC utilization but also limits the optimization space for dispatching algorithms.

Our experiments empirically proved that for a given QC number, fewer trucks decrease the objective value but make a larger optimization space. Ideally, an acceptable objective threshold should be set, and some optimization space is supposed to be left for the dispatching policy to play

a role in it. In this manner, the operational cost could be saved by less truck deployment while still maintaining the QC utilization. Also, fewer trucks help to alleviate the traffic congestion issue. The optimal truck-QC ratio for each configuration setting or concrete problem instance should be dynamic, and there is an underlying mapping between it and different scenarios. For the base configuration setting in section 4.6.2, this ratio is empirically confirmed within the range [4.5, 5.2], which achieved a relatively fair QC utilization level while maintaining an adequate optimization space for the RL agent.

Moreover, it is detected that the performance of the dispatching policy is sensitive to some spatial-related factors of containers. According to the experimental results, the stack mode shows little impact on policy performance since there is no outer-truck disturbance. Distributed yard mode offered more optimization space for the dispatching policy than the centralized yard mode. Except for the configuration settings, the relative locations of the involved yards and task sharing schemes among QCs for the same vessels are also crucial. This is related to the container storage space allocation problem, which is also a widely concerned COP in container terminals. In this work, these spatial-related factors are partially embedded into the configuration design for training and testing of the RL agent. Actually, the configuration design could be viewed as another decision-making problem and also affects the target objective. Therefore, the container terminal operation efficiency could also be facilitated by a sophisticated design of configuration, together with a fine-tuned RL dispatching policy.

The quay crane or truck deployment could be viewed as an important decision for container terminal management. Figures 4.3 and 4.4 display the generalization performance of the proposed DRL method compared with

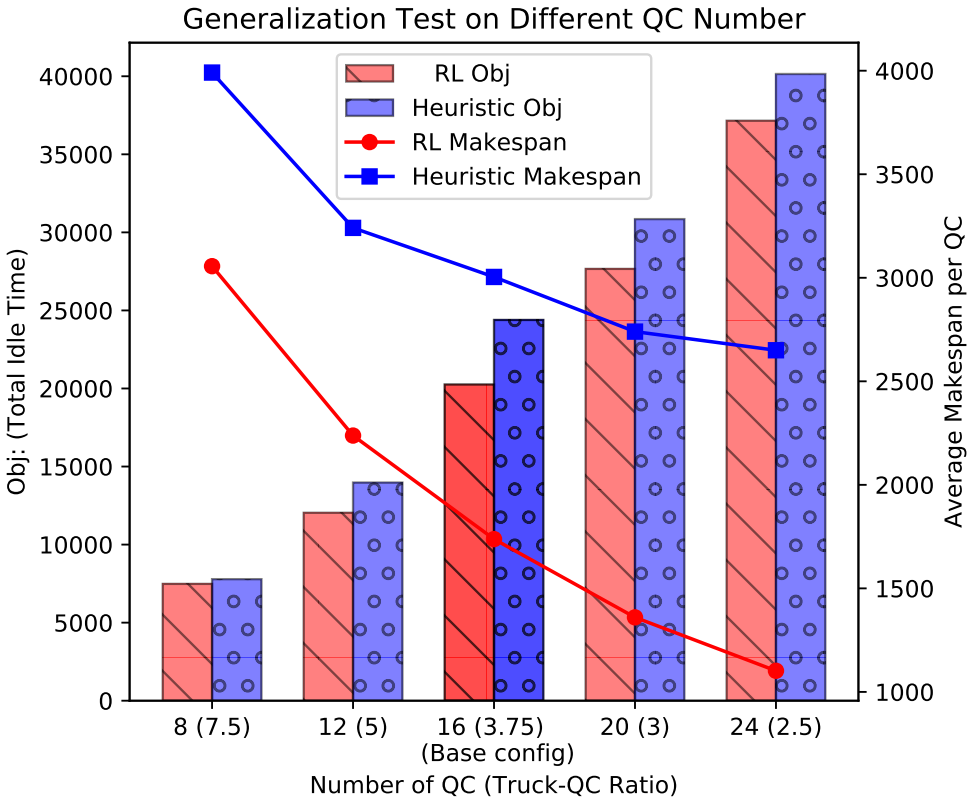


Figure 4.3: The performance of the proposed method trained on the single configuration in comparison with the benchmark under different QC amounts.

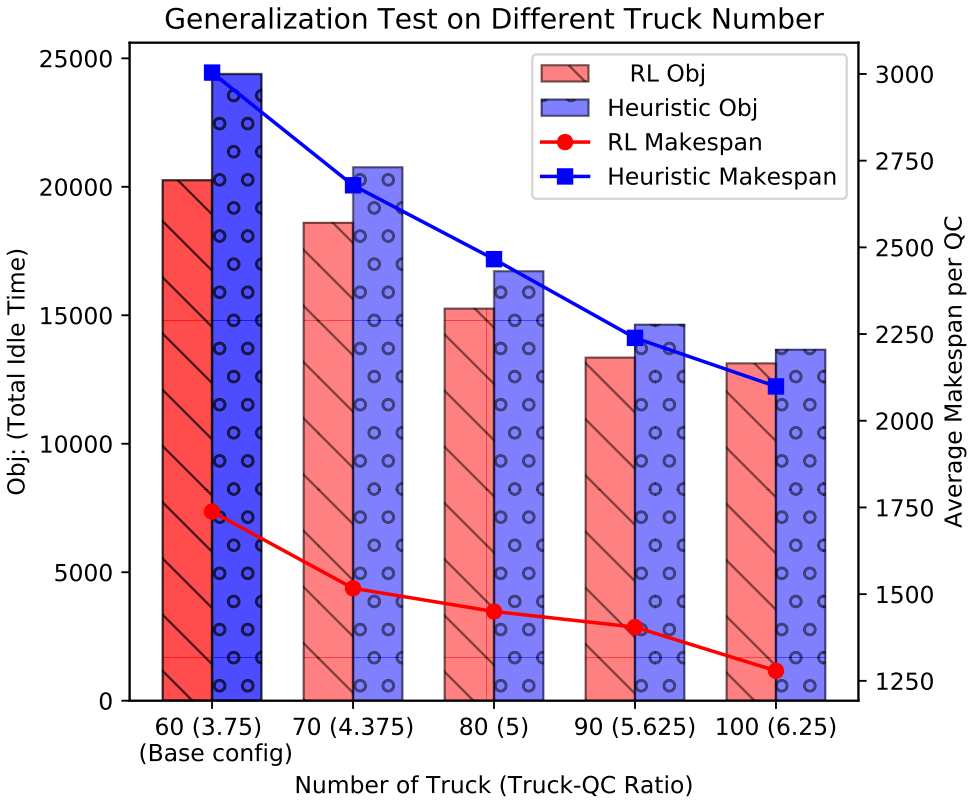


Figure 4.4: The performance of the proposed method trained on the single configuration in comparison with the benchmark under different truck amounts.

the benchmark on two metrics, total idle time (left Y-axis) and average makespan per QC (right Y-axis). Total idle time is the exact objective of the examined problem, and makespan indicates the time duration the QC takes to execute all its tasks. In experiments with different QC numbers (Figure 4.3), it could be found that a lower QC number makes a better objective (idle time) on both methods, since it increases the truck-QC ratio. However, the average QC makespan is worse because of more tasks per QC. Therefore, trade-offs between idle time and makespan of QCs are encountered when deciding the number of QCs to be deployed. In experiments with various truck numbers (Figure 4.4), more truck deployments make a reduction for both metrics at the expense of operational cost (deployment of trucks, fuel, labour cost, etc.).

This experiment further indicates several trade-offs among various equipment deployments. It is always strategic to balance cost and efficiency in daily port operational management. Our proposed methodology, along with the developed simulation system, is able to conduct exploration for various optimal equipment deployment strategies under diverse scenarios, which provides reliable conclusions for terminal authorities to schedule different resources, thus promoting the entire container terminal's efficiency.

4.7 Summary

In this work, we proposed a methodology that utilizes a deep reinforcement learning technique to solve a container truck task dispatching optimization problem in the real-world maritime container terminal. The truck dispatching process is considered as a sequential decision-making problem, and the corresponding MDP for the examined problem is formulated. Several im-

provements of the methodology implementation towards our previous work (Zhang et al., 2022a) are made. Firstly, the design of the state considers more spatial-temporal related features which better reflects the emergency level of quay cranes and is sensitive to the changing of problem configurations. Secondly, the action space covers the entire decision space that avoids the reachability problem discussed in the work of DRL-HH. Moreover, the sparse reward mechanism is adopted in the training process that fixes the myopic and misleading problem of the RL agent, which are discussed in Section 4.1.3 and 4.3. Finally, it has taken great efforts to design a tailor-made neural network structure that makes it more compatible with this examined problem. The network leverages LSTM and attention mechanisms in order to mask the invalid actions and adapts to the scenarios with various QC numbers.

In the process of building the problem instances, several real-world factors that could distinguish different scenarios have been taken into account. Also, we fully considered various types of uncertainties that stem from the real-world port operations in the training environment. These efforts make the DRL-generated policy more practical and robust to multi-scenario issues. Significant performance gains have been achieved towards DRL-HH, GP, and heuristic approaches. Apart from the crosswise contrast results, our proposed DRL approach also shows the competitiveness of the adaptiveness and generalization ability in relevant experiments. In addition, by analyzing the quantitative results, several insights that are practical and could promote the container terminal management are given.

Overall, our methodology is data-driven and barely relies on any exogenous forecasts. It respects the dynamic and uncertain nature of the examined problem. Furthermore, such a method is able to provide reliable references

for container terminal authorities to better arrange relevant resources under various operation scenarios.

Chapter 5

Methodology for Multi-Objective Dispatching

The chapter extends the previous truck dispatching problem to a multi-objective version. In this work, both the idle time of quay cranes and the truck empty traveling distance are considered. A novel methodology called preference agile multi-objective optimization (PAMOO) is proposed. The method aims at obtaining a uniform dispatching policy that could be able to handle arbitrary preferences among the objectives.

The extended problem is formulated as a multi-objective Markov decision process (MOMDP), and the inner-loop multi-objective reinforcement learning is utilized to design the methodology. A customized neural network for MOO is designed for the examined problem. Consequently, the obtained MOO policy set could be considered as the policies with arbitrary preference requirements that properly satisfy the diversity of the MOO concerns. Benefited by the generalization attributes of the proposed method, a preference calibration method is proposed that is able to further enhance the

quality of the approximated Pareto front.

5.1 Preference Agile Multi-Objective Optimization

This section briefly introduces the framework of the proposed preference agile multi-objective optimization (PAMOO) and its advantages towards the real-world complex online decision-making problems. Such methodology aims at only using one single uniform dispatching model to handle arbitrary preference weights among different objectives while satisfying the special requirements of the examined container truck task dispatching problem, such as short response time, robustness to uncertainties, and diversity of the trade-off policies for users' selection. The proposed method is still based on deep reinforcement learning techniques. Specifically, the RL agent takes a concrete preference weight as one of the inputs and then outputs the proper dispatching decision based on this weight. Because of the generalization ability of deep neural networks, the RL agent, once fully trained, is able to generate any dispatching decisions based on arbitrary preference weights. Benefit from such advantages, the proposed method also allows users to dynamically adjust the preference weight, thus, to revise their purposes. A special preference calibration method is also proposed which could be used to further refine a given policy set on both policy quality and diversity.

Figure 5.1 illustrates the mechanism of the proposed method. According to the left part of Figure 5.1, at time step t, one of the idle trucks needs to be dispatched a new task (dedicated to different QCs). There

are three choices, QC1, QC2, and QC3, with incremental queue lengths and decremental empty travel distances (indicated by three red lines in the left figure). Dispatching decisions could be made by the dispatcher network according to the current state (information of each QC) and a set of user preferences p_1, p_2, and p_3. Three corresponding Pareto optimal policies, which lead to selecting QC1, QC2, and QC3 by their probability distributions, are generated by the model, which maps to the points at the approximated Pareto front indicated by the right figure. $\pi(a|s_t, p1)$ and $\pi(a|s_t, p3)$ are policies that prefer minimizing QC idle time and truck empty travel distance separately, and $\pi(a|s_t, p2)$ is a utopia policy that balances both objectives ($p2$ is a preference like (0.5,0.5)).

5.2 Truck Dispatching Problem as an MOMDP

The examined container truck task dispatching process could be formulated as a Markov decision process (MDP). Recall the MDP formulation, a single objective MDP is described by the tuple (S, A, R, P, γ), which is introduced in section 4.1. The RL agent aims at maximizing the accumulated discounted reward at each time step t, which is denoted as $G_t = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$, where T is the total time step. When γ approaches 1, the agent treats immediate rewards and the possible rewards in the future equally, while prioritizing myopic decisions if γ is close to 0.

For a given state s, the state-value function $V^\pi(s)$ indicates the expectation of accumulated discounted reward in the future following policy π, which is defined by Equation (5.1). A state-independent value function V^π is defined by Equation (5.2), which evaluates a given policy π.

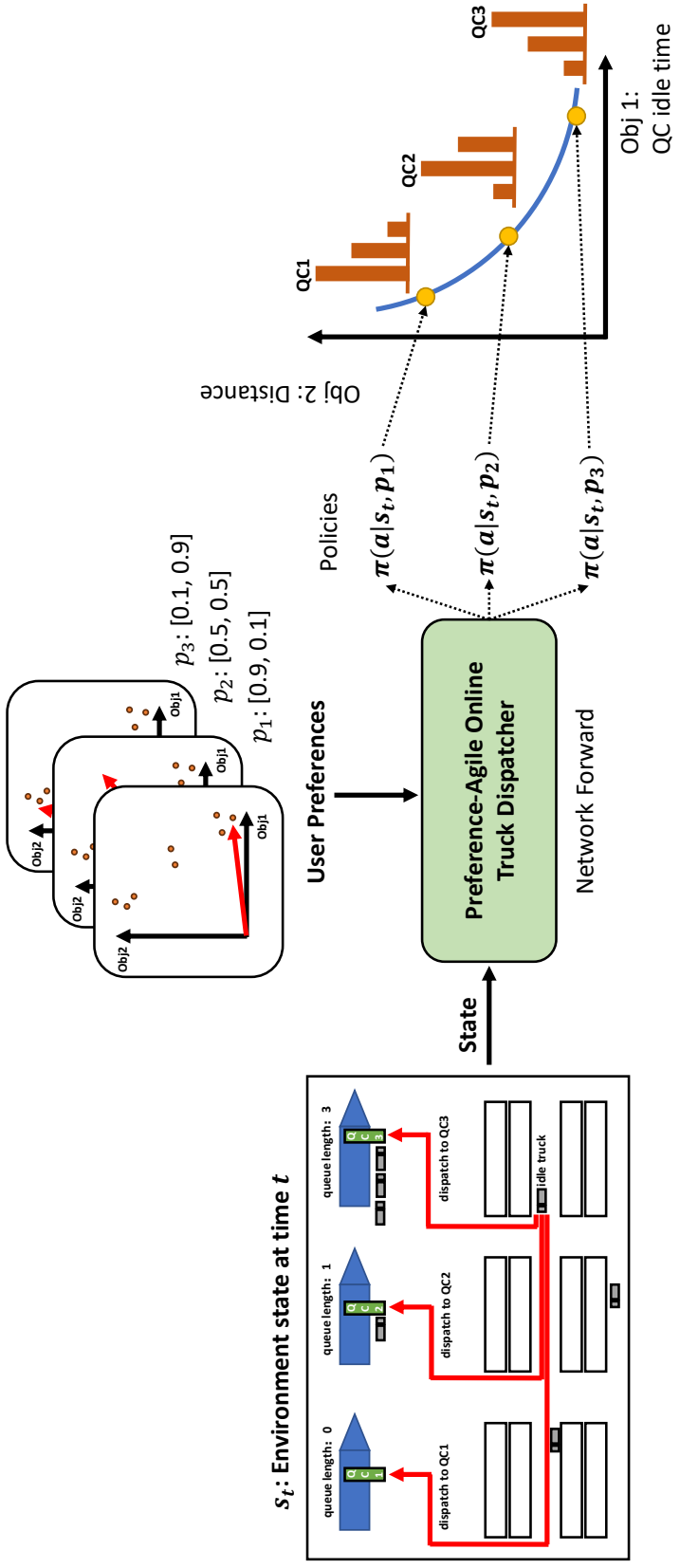


Figure 5.1: An example to illustrate the proposed methodology of preference agile multi-objective optimization for online container truck task dispatching.

$$V^{\pi}(s) = E(G_t|S_t = s) \tag{5.1}$$

$$V^{\pi} = E_s(V^{\pi}(s)) \tag{5.2}$$

The purpose of the RL agent is to find a policy $\pi(a|s,\theta)$, where the θ is the parameters of the policy, that could map the possible state vectors s to proper actions a in order to maximize $E_s(V^{\pi}(s))$. As usual in policy-based RL methods, the policy $\pi(a|s,\theta)$ is defined as a deep neural network with parameters θ. The θ is learned during considerable interactions with the environment.

The multi-objective reinforcement learning (MORL) could be formulated as a multi-objective Markov decision process (MOMDP), which is denoted as a tuple $(S, A, \boldsymbol{R}, P, \gamma)$. Compared with the single objective MDP, the reward function $\boldsymbol{R}(s, a, s')$ is a vector $\boldsymbol{r} \in \mathbb{R}^d$, where d is the number of objectives. Rather than a scalar reward in a single objective MDP, the vector-valued reward $\boldsymbol{r}$ indicates the immediate reward signals for all objectives at some time step t. Therefore, $\boldsymbol{V}^{\pi}$ denotes the state-independent value function vector of a given policy π in the case of MOMDP, and V_i^{π} is the i^{th} objective.

To evaluate a policy π in MOMDP, several concepts about multi-objective RL (Hayes et al., 2022) are introduced and defined as follows.

Definition 1 (Pareto Dominance) *A policy π is said to Pareto dominate another policy π' ($\pi \succ \pi'$) if and only if π's value vector is at least as high π''s in all objectives and is strictly higher in at least one objective, i.e.* $\pi \succ \pi' \Leftrightarrow \forall i, V_i^{\pi} \geq V_i^{\pi'} \wedge \exists i, V_i^{\pi} > V_i^{\pi'}$

Definition 2 (Pareto Optimality) *A policy π is said to Pareto optimal if there is no policy π' such that $\pi' \succ \pi$.*

Definition 3 (Pareto Set) *A Pareto set is defined as the set of all possible Pareto optimal policies.*

Definition 4 (Pareto Front) *A Pareto front is the visualization of the Pareto set in the objective space.*

In the field of multi-objective optimization, the examined objectives are usually conflicting, therefore, MORL aims to learn a set of Pareto optimal policies rather than to obtain a policy that generates an optimal solution with one single objective.

A common approach of MORL is to convert multi-objectives to single objective by using scalarization functions (also called utility functions in literature) $g(\boldsymbol{p}, \boldsymbol{o})$, where $\boldsymbol{p}$ is the given preference weight among different objectives which satisfies $\sum_{i=1}^{|\boldsymbol{p}|} p_i = 1$ and $\boldsymbol{o}$ is the vector of the specific objective values. For example, the Weighted-Sum aggregation $g_{ws}(\boldsymbol{p}, \boldsymbol{o}) = \boldsymbol{p}^T \boldsymbol{o}$ is the most commonly used scalarisation function. In this project, the Weighted-Tchebycheff aggregation is adopted and defined by (5.3), where the z_i^* is the ideal value that satisfies $z_i^* < min(o_i)$. The reason for using it is the ability to find any Pareto optimal policies with an adequate number of preference weights. It is proved that any Pareto optimal solution could be converted to an optimal solution of Equation (5.3) with a specific preference $\boldsymbol{p}$ (Choo and Atkins, 1983).

$$g_{wt}(\boldsymbol{p}, \boldsymbol{o}) = \max_{1 \leq i \leq d} \{p_i | o_i - z_i^* |\} \tag{5.3}$$

As mentioned in section 2.4, our methodology belongs to a multi-policy inner loop RL, which aims to obtain a set of Pareto optimal policies $\pi_{\boldsymbol{w}}(a|s,\theta)$ for any given preference weight vector $\boldsymbol{w}$ in a single run. To achieve this, the preference weight is considered as a special part of the state s and a uniform policy is defined as $\pi(a|s,\boldsymbol{w},\theta)$. A well-trained agent must make the most suitable action based on any combinations of s and $\boldsymbol{w}$ that minimize the scalarization objective $g(\boldsymbol{w},\boldsymbol{o})$.

5.3 Network Structure

Similar to most of the RL approaches, the truck dispatch policy $\pi(a|s,\theta)$ is approximated by a customized deep neural network with parameters θ. The structure of the policy network used in this research is depicted in Fig. 5.2. It can be seen that the proposed network takes both the raw observation of the state features described above and the user preference weight vector as the inputs and then outputs the action probability distribution across all candidate actions available for this given preference.

Firstly, the raw observations are fed into a feed forward layer to generate a 128-dimensional feature vector for each QC. Then, a multi-head attention block (following the same structure in (Vaswani et al., 2017)) is adopted to generate neighborhood-aware QC feature vectors. Then, each QC vector is combined with the preference vector generated by the preference embedding layer (a two-layer fully connected block) by taking the preference weights as the input. After this process, the QC feature vectors that incorporate preference information go through a feed forward layer to map each QC vector to a scalar, and together with a softmax layer to generate a probability distribution.

Unlike most other multi-objective reinforcement learning research studies, our method does not consider the preference as a homogeneous feature component in the agent's observation but deals with user-defined preference and state observations separately in the network. The preference information is merged into the QC vectors through Hadamard product, which is a kind of feature crossing operation, where the feature crossing is a common operation in recommendation system related networks. The reason for doing so is to make sure the preference weights play a more prominent role, such that the dispatching policies are sufficiently sensitive to various preference vectors. If the preference weights are concatenated to the raw state observations, based on our initial experiments, the network tends to ignore this part in the training, or is insensitive to the changes of preferences, which must be prevented. In contrast, Hadamard product operation transforms the QC feature vectors integrally and significantly. Furthermore, such an operation could also selectively highlight some parts in a QC feature vector since the agent may focus on different features when faced with various preference weights.

Apart from the capability to handle multi-objective preference weights, the network treats the inputs as a dynamic set of QC feature vectors. When a QC's task list becomes empty, the feature vector of this QC needs to be eliminated from the input so that the corresponding QC is excluded from the candidate actions. This issue is the same as that introduced in Section 4.2. The multi-head attention block is used to leverage this thanks to its abilities to properly handle the variable input length and its great perception performance of different parts of the features.

Generally, this proposed network structure is motivated by some common methodologies of natural language processing, recommendation systems,

and the works of single-objective optimization in Chapter 4.

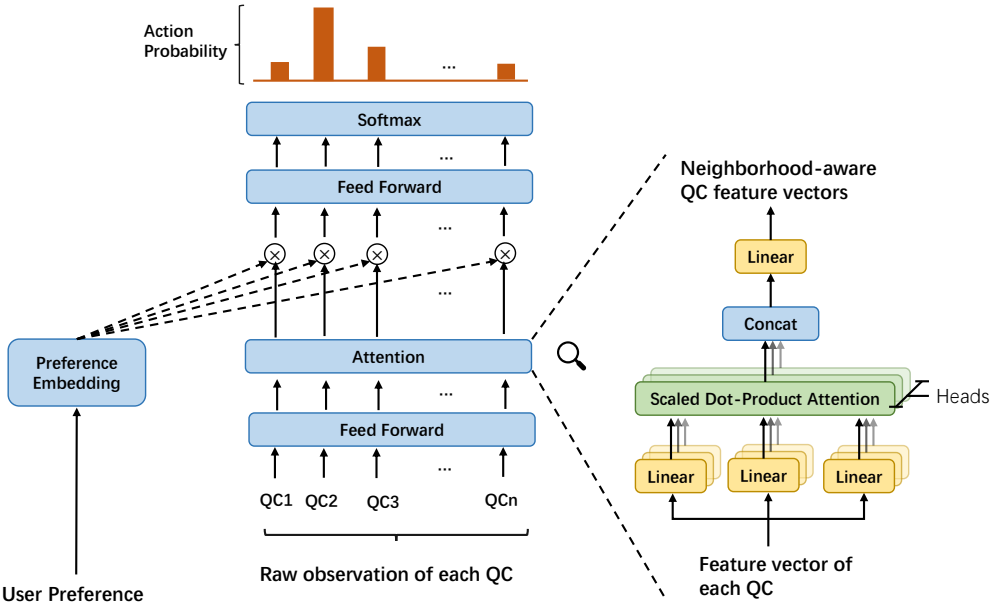


Figure 5.2: The network structure of preference-agile online truck dispatcher.

5.4 Algorithms

The well-known proximal policy optimization (PPO) (Schulman et al., 2017) is adopted to train the dispatching agent. PPO has demonstrated its superior performance on convergence and sample efficiency due to the mechanism of reusing the sampling data. The general process of the PPO is basically the same as the PPO used in Section 4.3 except for some details related to MOO mechanisms.

Denote $s_0^k, a_0^k, \boldsymbol{r_1^k}, ..., s_{H-1}^k, a_{H-1}^k, \boldsymbol{r_H^k}$ a trajectory of the agent's exploration in the k^{th} episode in training and K is the total number of episodes used in training. $\boldsymbol{r_t^k}$ indicates the reward vector for all objectives at time step t of episode k, and H is the total steps in an episode, which equals the total number of tasks of the episode. The total scalar reward with preference

weight $\boldsymbol{p}$ is defined by Equation (5.4) based on the Weighted-Tchebycheff scalarization function.

$$R^k = \sum_{t=1}^{H} g_{wt}(\boldsymbol{p}, \boldsymbol{o})^t \tag{5.4}$$

where $g_{wt}(\boldsymbol{p}, \boldsymbol{o})^t$ is a weighted-Tchebycheff function (see Equation (5.3)) at time step t. To enhance the stability of the training process, a shared baseline is used to compute the advantage function value for each s_t^k, a_t^k pair as follows:

$$A(s_t^k, a_t^k) = R^k - \frac{1}{K} \sum_{k=1}^{K} R^k \tag{5.5}$$

The probability ratio is defined by Equation (5.6), where the θ_{old} are the policy network parameters before updating.

$$Ratio_t^n(\theta) = \frac{\pi(a_t^k | s_t^k, \theta)}{\pi(a_t^k | s_t^k, \theta_{old})} \tag{5.6}$$

PPO optimized a surrogate objective with a clipped probability ratio. A clipped advantage function is defined by Equation (5.7), where ϵ is a hyper-parameter ($0 < \epsilon < 1$).

$$A^{clip}(s_t^k, a_t^k) = min[Ratio_t^k(\theta) A(s_t^k, a_t^k), clip(Ratio_t^k(\theta), 1-\epsilon, 1+\epsilon) A(s_t^k, a_t^k)] \tag{5.7}$$

Each update of the policy network parameters uses a sampled mini-batch

of data (s_t^k, a_t^k) pairs and their advantage function values with batch size B. Each update optimizes the policy with the same preference weight $\boldsymbol{p}$. The estimated gradient of PPO loss for a particular preference weight is indicated by Equation (5.8). The entire training process is presented by the Algorithm 2. In each iteration, preference weight $\boldsymbol{p}$ is selected randomly, and each preference will gain adequate learning after enough number of iterations. Also, $\boldsymbol{p}$ could also be selected in order in each generation, and the result is basically the same according to the experiment implementation.

$$\nabla \mathcal{L} = \frac{1}{KH} \sum_{k=1}^{K} \sum_{t=1}^{H} A^{clip}(s_t^k, a_t^k) \nabla \log P_\theta(a_t^k | s_t^k, \boldsymbol{p}) \tag{5.8}$$

Algorithm 2: PPO for preference-agile multi-objective optimization

Input: number of iteration K, collect N episodes per iteration, steps per episode T, M epochs per iteration, clipping rate ϵ, batch size $B(B < NT)$, preference set $\mathcal{P}$

Initialize: a differentiable truck dispatch policy parameterization $\pi(a|s, \boldsymbol{p}, \theta)$;

for *i=1 : K* **do**

  Randomly select a preference weight $\boldsymbol{p}$ from $\mathcal{P}$;

  for *n=1 : N* **do**

    Collect an episode $s_0, a_0, r_1, ..., s_{T-1}, a_{T-1}, r_T$, following $\pi(\cdot|\cdot, \boldsymbol{p}, \theta)$;

    Compute each $r_t(a_{t-1}, s_{t-1})$ based on the reward design, $\forall t \in [1, T]$;

    Compute R^n for preference $\boldsymbol{p}$ based on Equation 5.4;

  end

  Compute $A(s_t^n, a_t^n)$ through Equation 5.5, $\forall t \in [0, T-1], \forall n \in [1, N]$;

  Compute PPO loss $\mathcal{L}$ with clipping rate ϵ, optimize the parameters θ with M epochs and batch size B

end

Benefiting from the generalization ability of deep neural networks, a well-trained dispatching policy is able to make proper dispatching decisions

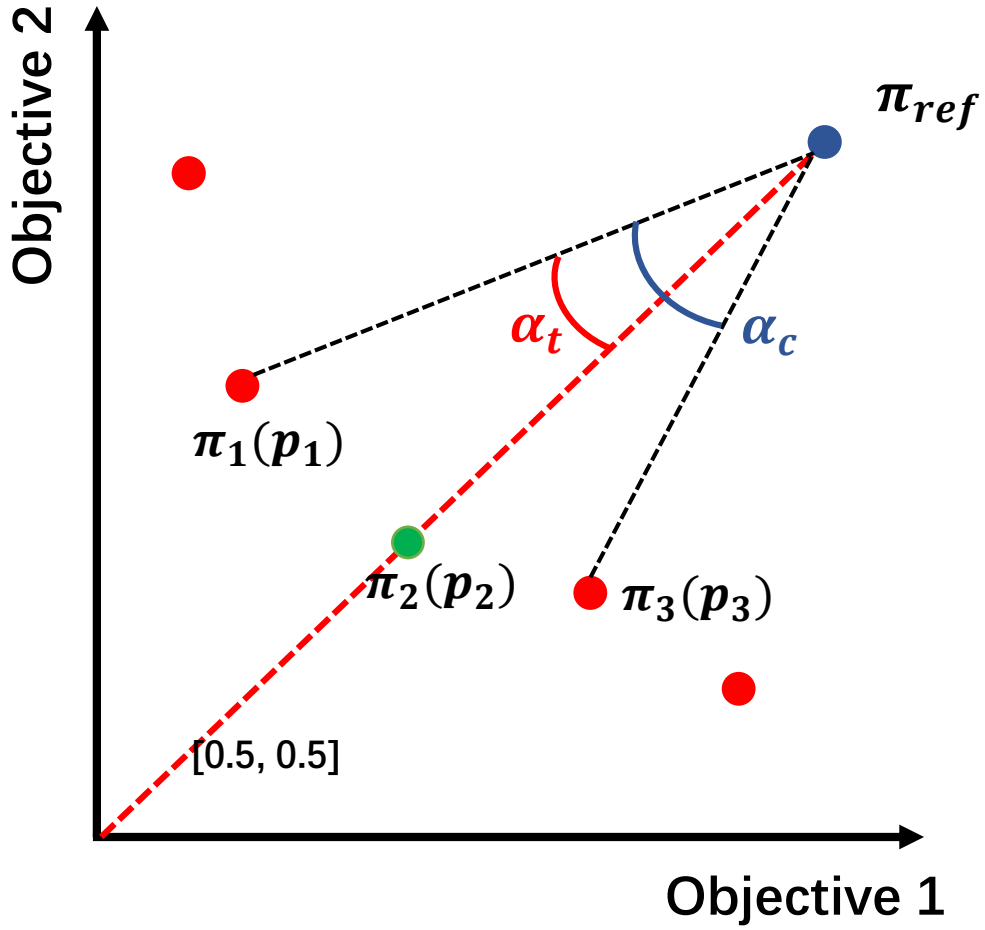


Figure 5.3: Interpretation of the preference calibration method.

under arbitrary (including unseen) preferences. Intuitively, a set of evenly distributed preference weights would be used to generate an even and dense Pareto front. However, for numerous real-world applications, evenly distributed preferences always fail to generate a regular shape of the approximate Pareto front. To tackle such an issue, we proposed a heuristic approach to calibrate the preference weights based on the relative locations of a set of policies in the objective space. The idea of the preference calibration method is illustrated in Figure 5.3, where a set of non-dominant policies (indicated by red points) generated by even distributed preferences are given. Suppose a policy with preference (0.5, 0.5) which should ideally locate at the direction vector of (0.5, 0.5) (indicated by red dashed line), but the actual location drifts off to some extent. Therefore, this preference needs to be adjusted to make the new policy (a possible policy is π_2 with preference p_2, which is indicated by green points) locate in the correct direction. Firstly, two policies π_1 and π_3 with preferences p_1 and p_3, which are closest to the target direction (0.5, 0.5) are selected. The area between $\pi_1(p_1)$ and $\pi_3(p_3)$ is considered as the neighborhood of the target direction (0.5, 0.5). The ratio of the angles α_t and α_c is computed. Then, the calibrated preference p_2 could be obtained by Equation (5.9).

$$\boldsymbol{p_2} = \boldsymbol{p_1} + \frac{\alpha_t}{\alpha_c}(\boldsymbol{p_3} - \boldsymbol{p_1}) \tag{5.9}$$

This calibration method estimates the preference adjustment based on the change of the angles of two policies located in the neighborhood area of a particular direction. This preference calibration method could make the entire Pareto front more even, thus improving the quality of the approximated Pareto set policies as a whole.

5.5 Problem Instances

Each of the instances used in this study consists of 28 QCs, 100 yards, and 700 tasks. The number of trucks varies between different instances, and the impact of this variation shall be analyzed shortly. Yard blocks for import and export containers are fixed. In order to enhance the degree of mutual exclusiveness of the two objectives, the import and export yards are both evenly distributed among the entire container terminal based on some initial investigation. Due to the dynamic nature and the high-level uncertainties in the environment, the same dispatching policy cannot guarantee the same results in terms of the two objectives, but it fluctuates within a range instead. Therefore, the i^{th} objective of a policy $\boldsymbol{V}^{\pi}$ is evaluated by C times average implemented by π, which is shown in Eq. (5.10).

$$V_i^{\pi} = \frac{1}{C} \sum_{n=1}^{C} \sum_{t=1}^{H} r_{t,(i)}^{n} \tag{5.10}$$

C is set to 128 in this work. That is, in the testing stage, 128 problem instances (instances generated with fixed random seeds, which could ensure a reproducible result for the same dispatching policy) are used for evaluation.

5.6 Evaluation Metrics

The hypervolume (HV) is selected as the performance indicator to measure each method. A reference policy whose value $\boldsymbol{V}^{ref}$ is dominated by all other non-dominated policies is first needed. For bi-objective optimization, HV measures the space area covered by the non-dominated policies and

reference policy over the policy value space. Fig. 5.4 illustrates the HV indicator, where the $\pi_1, \pi_2, \pi_3, \pi_4$ indicate the non-dominated policies, π_{ref} is the reference policy, and the size of the grey area is the value of HV. HV is a common indicator to evaluate multi-objective optimization methods. Generally, a higher HV value implies a better non-dominated policy set as a whole, which indicates better algorithm performance. In the field of multi-objective optimization, HV is the most commonly used indicator.

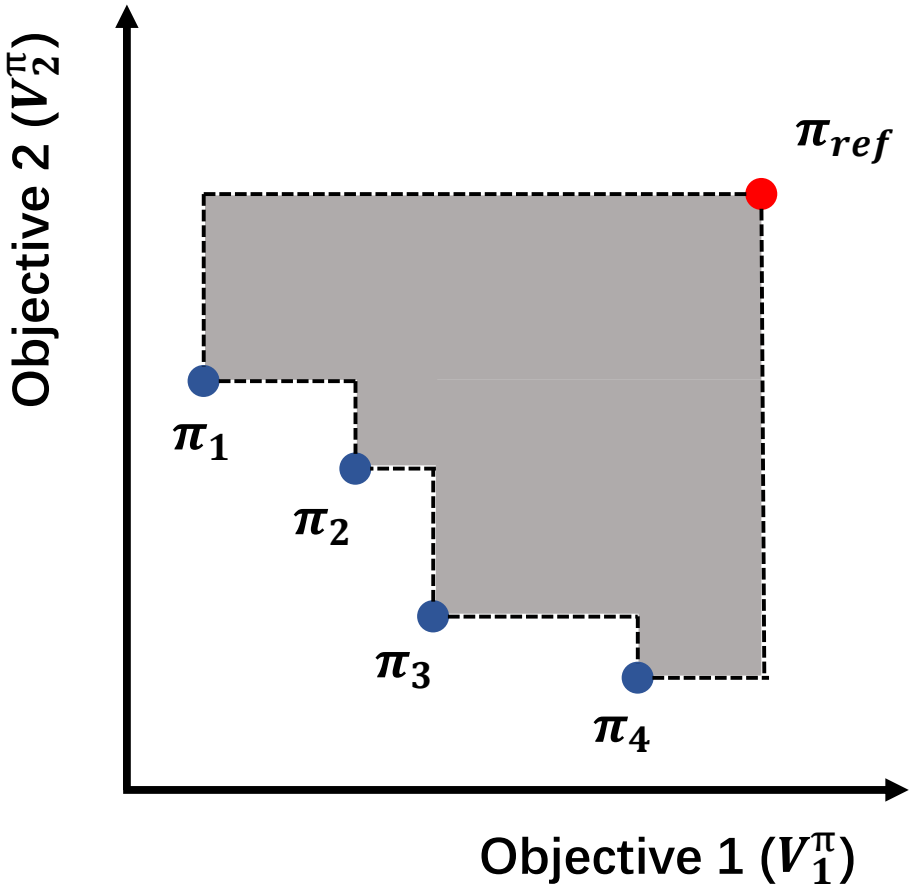


Figure 5.4: Hypervolume indicator.

Apart from the HV, the sparsity is also adopted for the evaluation. The sparsity of a given policy set $\mathcal{S}$ with m objectives is defined by Eq. (5.11), where $\widetilde{\mathcal{S}}_j(i)$ is the i^{th} objective value in a partial ordering of these policies sorted by the j^{th} objective. Sparsity evaluates the degree of uniformity of a policy set. A lower sparsity value indicates a more even distribution of

policies spread among the entire approximate Pareto front. When evaluating, HV is considered as the primary focus, and when values of HV are similar, the policy set with lower sparsity is preferred. Sparsity is another important measurement of MOO policy set (Roijers et al., 2013).

$$Sparsity(\mathcal{S}) = \frac{1}{|\mathcal{S}| - 1} \sum_{j=1}^{m} \sum_{i=1}^{|\mathcal{S}|-1} (\widetilde{\mathcal{S}}_j(i) - \widetilde{\mathcal{S}}_j(i+1))^2 \tag{5.11}$$

Due to the difference in the measurement units of different objectives, the value scale of an objective may vary a lot from the other, which would cause bias for evaluating the Pareto front. To eliminate such effect, both objective values are scaled to range of [0, 1] by using min-max normalization which is defined by Eq. (5.12), where x is some objective value of a policy and X is the set of objective values generated by all methods to be compared. When computing HV, the value of reference policy $\boldsymbol{V}^{ref}$ is naturally selected as [1, 1]. Therefore, the HV became a value in the range of [0, 1].

$$z = \frac{x - min(X)}{max(X) - min(X)} \tag{5.12}$$

5.7 Benchmarks

To demonstrate the performance of the proposed methodology, two evolutionary algorithms, namely, multi-objective genetic programming (GP) based on paradigms of NSGA-II (Deb et al., 2002) and MOEA/D (Zhang and Li, 2007) are implemented respectively. The reason for choosing these two benchmarks is that they are two quite classic algorithms in the field of multi-objective optimization, and most of the other new proposed methods

are either derived from one of these two or combine the mechanisms of these two methods.

Traditionally, both NSGA-II and MOEA/D solve the deterministic problems whose solutions could be encoded as fixed-length chromosome vectors (routes, plans, etc) and the operators of genetic algorithm could be adopted during the evolution. Since the examined problem is an online one, the solutions in this work must be a dispatching rule, with which a dispatching decision can be generated. Therefore, both benchmark algorithms are implemented by us. In our proposed method, such a rule is the actor network, while in the GP benchmark algorithms, we make it an arithmetic tree (Chen et al., 2020b) to rank different candidate actions. For both algorithms, the population size is set to 1000. The maximum generation is set to 500, 1000, 2000, respectively, in the corresponding experiments. The probabilities for both crossover and mutation operators are settled at 50% after some initial trials. For the multi-objective related mechanisms, we followed the original works of NSGA-II (Deb et al., 2002) and MOEA/D (Zhang and Li, 2007).

5.8 Experiments and Result Analysis

5.8.1 Comparative Results with Benchmarks

Table 5.1 summarizes the comparative results between the proposed PAMOO method with two main-stream MOO methods. The performance of HV and sparsity (united by 10^{-4}) are presented. The gap is measured against the algorithm with the highest HV. For evolutionary algorithms, results after 500, 1000, and 2000 generations of evolution are reported. For PAMOO,

results of 11, 51, and 101 numbers of evenly distributed preferences are selected as the corresponding comparison. Each preference is evaluated by running 128 times to guarantee the robustness. According to the numerical results, PAMOO with 101 preset preference vectors outperforms the other two methods on both HV and sparsity. Overall, the best results (2000 generations) of the two evolutionary algorithms have an average gap of 9.74% towards our method (with 101 preferences) in terms of HV. The performance of NSGA-II and MOEA/D are comparable with each other in general. The HV of MOEA/D is slightly higher than NSGA-II, while the sparsity of NSGA-II is better than MOEA/D. PAMOO with much fewer preferences (11 and 51) have average gaps of 0.88% and 0.41% respectively, which demonstrates the excellent ability of the proposed method in that even a smaller number of preset preferences can approximate the Pareto Frontier very well.

The visualization of the obtained approximated Pareto frontiers is presented in Fig. 5.5. It can be seen that the approximate Pareto frontiers obtained by our method could completely cover those by two evolutionary algorithms. In cases of 2000 generations, merely a few policies generated by evolutionary algorithms are close or at the same level as our method (see Figure 5.5 (c) and (i)). Such outstanding results demonstrate the great potential of RL-based methods for online MOO problems. Compared to evolutionary algorithms, RL seems to have a better perception of the dynamics of the environment. This is probably due to the power of the deep neural network, which can better handle high-dimensional features and their temporal and spatial relationships, while genetic programming-based methods are constrained by the limited set of arithmetic operators, which make far less use of feature data.

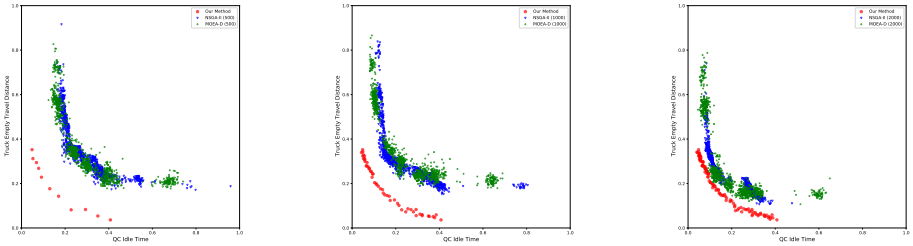
Table 5.1: Comparison Result to NSGA-II and MOEA/D.

Method	80 Trucks			100 Trucks			120 Trucks		
	HV	Sparsity	Gap(%)	HV	Sparsity	Gap(%)	HV	Sparsity	Gap(%)
NSGA-II (500)	0.669	91.86	25.0%	0.674	31.68	23.5%	0.708	21.73	20.45%
NSGA-II (1000)	0.729	19.44	18.27%	0.728	20.1	17.37%	0.759	16.1	14.72%
NSGA-II (2000)	0.807	4.01	9.53%	0.784	23.8	11.01%	0.798	9.31	10.34%
MOEA/D (500)	0.682	31.94	23.54%	0.675	75.3	23.38%	0.693	19.75	22.13%
MOEA/D (1000)	0.72	35.1	19.28%	0.746	56.85	15.32%	0.751	53.66	15.62%
MOEA/D (2000)	0.819	27.85	8.18%	0.798	35.58	9.42%	0.801	27.71	10.0%
Ours (11 pref.)	0.88	42.38	1.35%	0.857	41.88	2.72%	0.876	53.13	1.57%
Ours (51 pref.)	0.89	3.67	0.22%	0.873	4.21	0.91%	0.889	4.22	0.11%
Ours (101 pref.)	0.892	2.97	0.0%	0.881	2.58	0.0%	0.89	1.67	0.0%

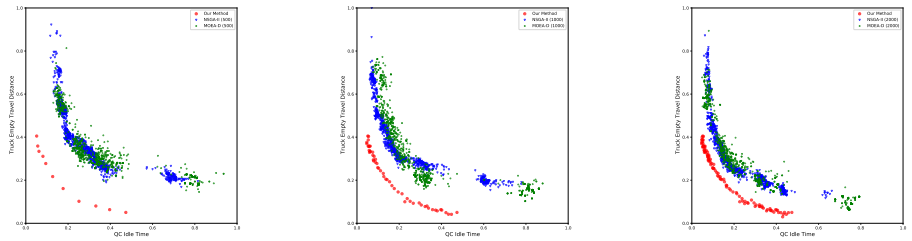
Apart from HV, the proposed PAMOO method also shows significant performance gains in terms of smoothness and spread of the Pareto frontiers. The Pareto frontiers by the two evolutionary algorithms usually have large gaps (See Figure 5.5 (d),(e), and (f)). A closer investigation reveals that a likely reason for this is the lack of effective fine-tuning mechanisms to accurately respond to small changes in preferences. Usually, a policy presented by a tree-based structure leads to large changes in its dispatching logic when modified by genetic operators (crossover, mutation). Such characteristics make the policies tend to be the same for similar preferences, thus forming the non-continuous intervals among the Pareto frontiers. In contrast, our method could still make the correct perception even though the preference is slightly changed because the preference embedding layer maps the preference to a longer feature vector (128 dimensions, same as the QC feature vector), which is adequate to discriminate similar preferences and influence the neural network. That is the reason why our method could generate much more continuous and even Pareto fronts compared with the benchmarks. In addition, the problem of "equivalent trees" (different trees have the same arithmetic logic) also exists for evolutionary algorithms and makes considerable policies overlap or gather in the objective space, which further worsens the discontinuity of the Pareto front.

5.8.2 Performance of Generalization

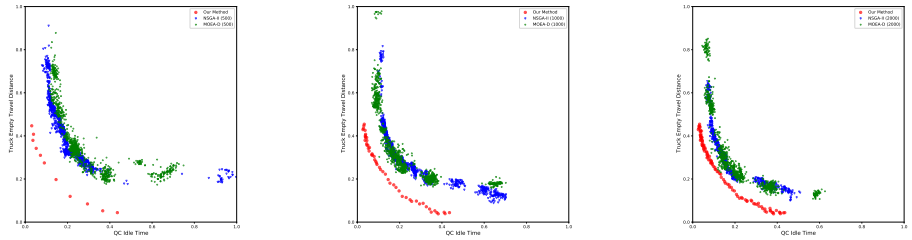
We also conducted PAMOO's generalization experiment by using merely 11 guidance preferences on instances with 120 trucks for training. The results are evaluated against the PAMOO with 101 preferences. The agent yields 40 non-dominated policies out of 101 input preferences. Figure 5.6 visualizes the approximated Pareto front. The numerical results of poli-



(a) 80 Trucks, 500 Generations(b) 80 Trucks, 1000 Generations(c) 80 Trucks, 2000 Generations



(d) 100 Trucks, 500 Generations(e) 100 Trucks, 1000 Generations(f) 100 Trucks, 2000 Generations



(g) 120 Trucks, 500 Generations(h) 120 Trucks, 1000 Generations(i) 120 Trucks, 2000 Generations

Figure 5.5: Approximate Pareto front obtained by our method and benchmarks on instances of different number of trucks.

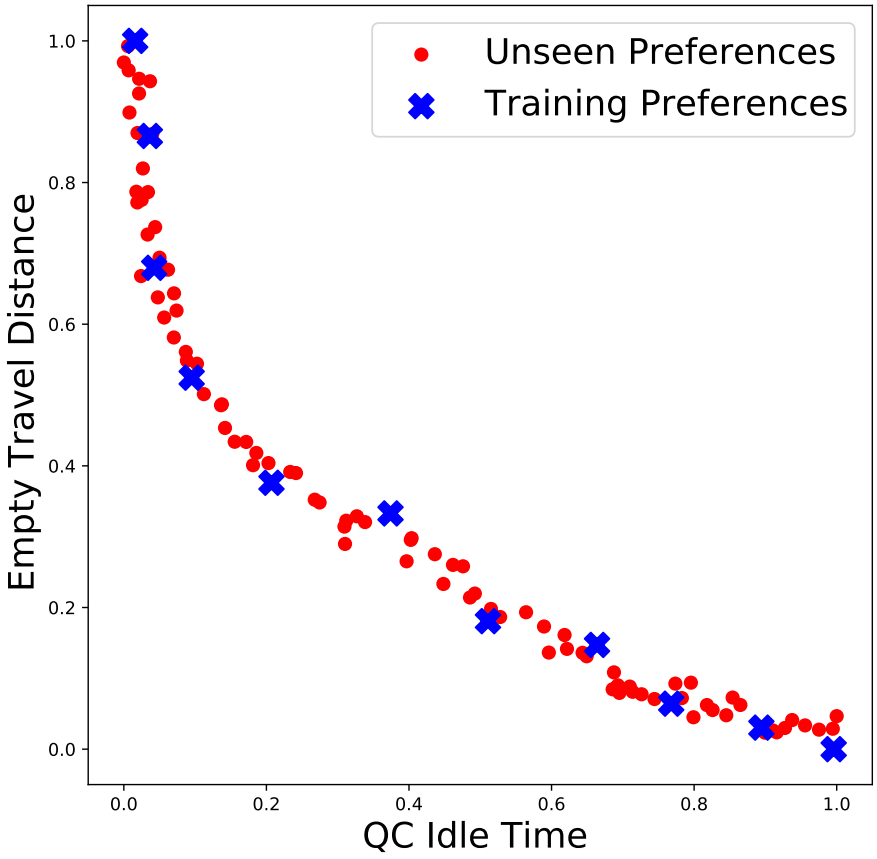


Figure 5.6: Pareto frontiers generated by the proposed method trained by a small number of preferences.

cies with training and unseen preferences are reported in Table 5.2 and 5.3 separately. The numerical results of the approximated Pareto front by these two groups of preferences are presented in Table 5.4. According to these results, the generalized policies could fill up the intervals among the policies with training preferences over the objective space, and all policies approximate a relatively dense Pareto front. Like most of the machine learning approaches, less training samples (preferences in this case) lead to certain degrees of performance drops. The agent is able to generate 11 non-dominant policies out of 11 input preferences if the evaluating preferences are the same as the training instances, but when 90 unseen evaluating instances are added, the policy can only generate 29 more non-dominated policies. Such a problem could be alleviated by taking more preferences into the training process and increasing the network size. However, the generalized non-dominated policies also help to enhance the approximated Pareto front. According to the Table 5.4, with the contributions of the generalized policies, the approximated Pareto front obtained a significant improvement on HV. In general, the generalization performance is crucial for the proposed methodology as it makes it possible for the agent to dispatch under arbitrary preferences and allows users to interactively adjust the preferences in real-world cases.

5.8.3 Comparative Results with Outer-Loop Method

We compare the performance of PAMMO with an outer loop method. As introduced in the literature, the outer loop methods consider multi-objective optimization problems as several single-objective optimization problems with different preferences. In other words, an outer loop method needs to train separate dispatching policies for different preferences. In this

Table 5.2: Numerical Result of Policies with Training Preferences.

Preferences	QC Idle Time (min/QC)	Empty Travel Distance (m/task)
(1.0, 0.0)	13.04	799.11
(0.9, 0.1)	13.21	759.0
(0.8, 0.2)	13.26	703.38
(0.7, 0.3)	13.69	657.03
(0.6, 0.4)	14.6	612.71
(0.5, 0.5)	15.97	599.77
(0.4, 0.6)	17.09	554.33
(0.3, 0.7)	18.34	544.31
(0.2, 0.8)	19.19	519.52
(0.1, 0.9)	20.23	509.42
(0.0, 1.0)	21.06	500.34

experiment, the same network structure, except for the eliminated preference embedding layer, is used for training single-objective policies. Since training a single policy from scratch is time-consuming, only 11 policies are trained separately for comparison in this experiment. Figure 5.7 shows the approximated Pareto frontiers on 11 evenly distributed preference weights. According to the Table 5.6, the outer loop method generates comparative Pareto frontiers towards PAMOO in terms of HV (0.786 and 0.784, respectively). The gap is merely 0.25%. Bear in mind that our proposed PAMOO method only uses a single uniform model. However, the outer loop method has to be trained separately for different preferences which is extremely time costly.

Visually, the approximated Pareto front obtained by the outer loop method covers a broader space. This property is further analyzed by numerical results of each preference, which are presented in Table 5.5. In extreme cases (preferences of (1.0, 0.0) or (0.0, 1.0)), the examined problem degenerates into a single-objective problem, and the outer loop method shows better performance. Numerically, the outer loop method achieved the ranges of 15.08 min/QC and 393.86 m/task for both objectives, while PAMOO ob-

Table 5.3: Numerical Result of Policies with Unseen Preferences.

Preferences	QC Idle Time (min/QC)	Empty Travel Distance (m/task)
(0.98, 0.02)	12.96	786.7
(0.97, 0.03)	12.91	789.99
(0.93, 0.07)	12.97	768.88
(0.88, 0.12)	13.05	735.47
(0.85, 0.15)	13.06	730.95
(0.81, 0.19)	13.1	699.9
(0.78, 0.22)	13.3	690.91
(0.76, 0.24)	13.37	682.41
(0.74, 0.26)	13.48	674.0
(0.73, 0.27)	13.62	667.91
(0.72, 0.28)	13.63	664.25
(0.71, 0.29)	13.75	662.92
(0.69, 0.31)	13.83	650.1
(0.68, 0.32)	14.02	645.41
(0.66, 0.34)	14.07	635.84
(0.65, 0.35)	14.18	630.01
(0.64, 0.36)	14.31	629.93
(0.63, 0.37)	14.39	620.12
(0.59, 0.41)	14.82	617.27
(0.58, 0.42)	14.88	616.81
(0.57, 0.43)	15.1	605.57
(0.56, 0.44)	15.16	604.39
(0.54, 0.46)	15.44	594.21
(0.52, 0.48)	15.45	586.91
(0.47, 0.53)	16.15	579.58
(0.44, 0.56)	16.58	570.03
(0.42, 0.58)	16.88	564.3
(0.39, 0.61)	17.12	559.48
(0.38, 0.62)	17.23	556.08
(0.36, 0.64)	17.73	552.08
(0.35, 0.65)	17.79	541.08
(0.32, 0.68)	18.18	540.98
(0.31, 0.69)	18.22	539.52
(0.27, 0.73)	18.52	525.59
(0.25, 0.75)	18.6	524.07
(0.22, 0.78)	18.85	523.49
(0.21, 0.79)	19.0	521.48
(0.16, 0.84)	19.45	513.79
(0.09, 0.91)	20.27	507.28

Table 5.4: Comparison of approximated Pareto fronts obtained by various groups of preferences.

Methods	HV	|Sols|	Sparsity (10^{-4})	Gap(%)
Training preferences	0.701	11	256.68	6.53%
Generalized preferences	0.748	39	22.99	0.27%
All preferences	0.75	40	24.81	0%

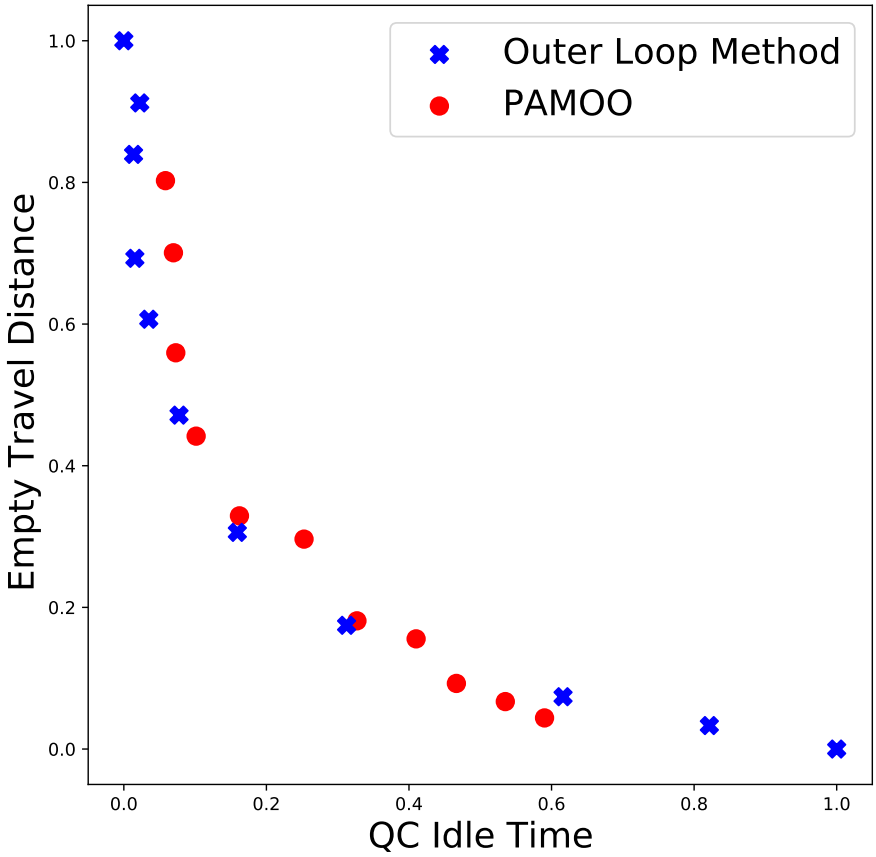


Figure 5.7: Pareto frontiers generated by PAMOO and outer loop method on the instance of 120 trucks.

tained the ranges of 8.02 min/QC and 298.77 m/task respectively, having the gaps of 46.8% and 24.1%. In general, in extreme cases, one of the objectives is further optimized to a small degree at the expense of a dramatic performance drop for the other objective.

Another advantage of the proposed methodology is the higher sample efficiency compared with the outer loop method and evolutionary algorithms. A sample indicates one single run for a particular problem instance and is equivalent to an episode in the RL training process. The Figure 5.8 shows the total number of samples (episodes) that are required for these algorithms along with the increase of HV. For both evolutionary algorithms, their population sizes are large, and numerous samples are required to evaluate the fitness of individuals reliably and avoid over-fitting and performance drop for testing. For RL-based approaches, our proposed method requires less than half of the episodes compared with the outer loop method. This is probably because the features extracted by the network at different preferences are correlated, and hence learning is transferred across different preferences. For example, the dispatching policy under preferences of (0.5, 0.5) must be somehow similar to the policies under preferences of (0.4, 0.6) or (0.6, 0.4). Therefore, when the network is trained for preference (0.5, 0.5), it is also partially trained for the preferences (0.4, 0.6) and (0.6, 0.4). More precisely, it is believed that when the network is trained for one particular preference, it is actually trained for all other possible preferences simultaneously to some extent. This also explains why our proposed method can produce solutions at preferences that are not seen in the training.

Table 5.5: Numerical Result of Comparison to Outer Loop Method.

Preferences	Our Method		Outer Loop Method	
	QC Idle Time (min/QC)	Empty Travel Distance (m/task)	QC Idle Time (min/QC)	Empty Travel Distance (m/task)
(1.0, 0.0)	13.04	799.11	12.16	876.9
(0.9, 0.1)	13.21	759.0	12.49	842.33
(0.8, 0.2)	13.26	703.38	12.36	813.71
(0.7, 0.3)	13.69	657.03	12.39	755.92
(0.6, 0.4)	14.6	612.71	12.68	722.04
(0.5, 0.5)	15.97	599.77	13.32	668.71
(0.4, 0.6)	17.09	554.33	14.56	603.52
(0.3, 0.7)	18.34	544.31	16.87	551.84
(0.2, 0.8)	19.19	519.52	21.45	512.14
(0.1, 0.9)	20.23	509.42	24.55	496.1
(0.0, 1.0)	21.06	500.34	27.24	483.04

Table 5.6: Comparison of approximated Pareto fronts obtained by PAMOO and outer loop method.

Methods	HV	|Sols|	Sparsity (10^{-4})	Gap(%)
PAMOO	0.784	11	113.19	0.25%
Outer Loop Method	0.786	10	365.92	0%

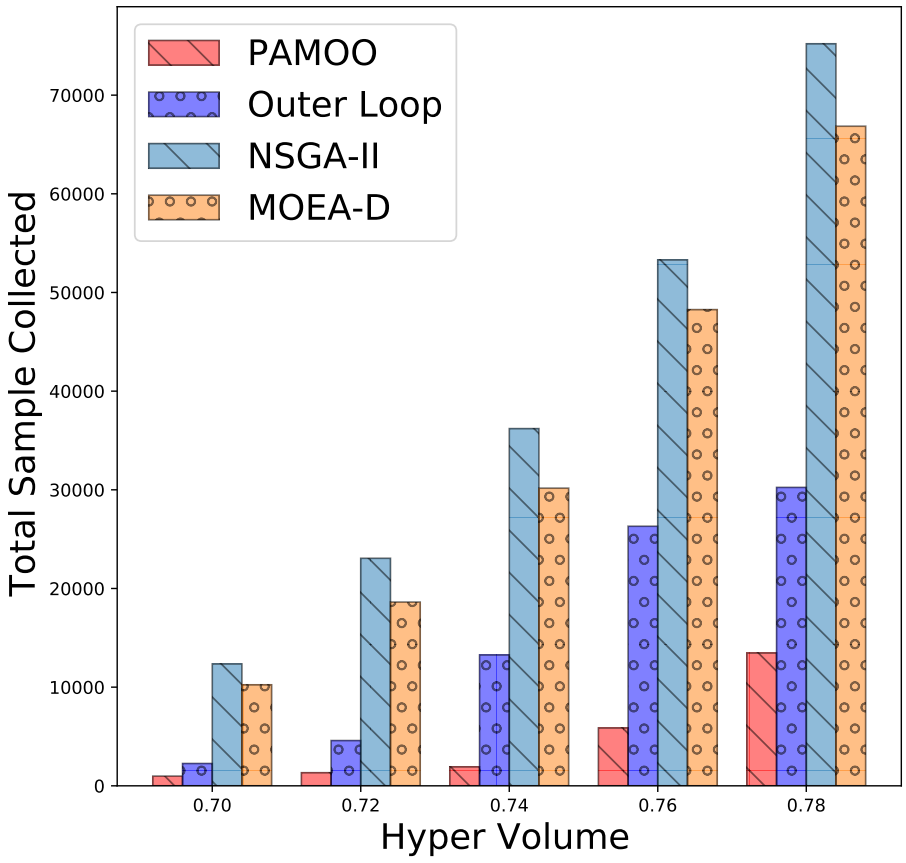


Figure 5.8: Sample efficiency of inner-loop (PAMOO) and outer-loop methods compared with NSGA-II and MOEA-D.

5.8.4 Result of Preference Calibrations

The experiment of preference calibration is conducted on instances with 80, 100, and 120 trucks. The specific adjusting method based on a set of existing Pareto policies is described in section 5.4. Firstly, a set of Pareto policies is generated by a list of evenly distributed preferences. Then, each preference is adjusted by using the aforementioned preference calibration method based on the relative locations of these policies in the objective space. Table 5.7 shows the PAMOO results with and without calibrations. In our experiments, the number of preferences is set to different values (11, 21, and 51) to evaluate the trends of the performance gains through sampling. It can be seen that, at the same number of preferences, PAMOO with calibration achieves better HV scores than PAMOO without calibration. It is axiomatic that more preferences contribute to higher HV as well. When the number of training preferences reaches 51, PAMOO with calibration obtains the best results among all.

According to the numerical result, PAMOO with preference calibration can improve the results between 0.4% and 1.79%. Calibration gives smaller improvements when the number of preferences is relatively large, since the optimization space for adjusting preferences is quite limited. Despite little improvement for HV, calibration is still valuable for the improvement on sparsity since the preference calibration makes the Pareto frontier more even.

The reason why there's a space for improving HV by using the preference calibration method is that the approximated Pareto front is not evenly explored by evenly distributed preferences. For the examined truck dispatching problem, the direct reason for leading to an uneven and irregular

Pareto front is non-equivalent sensitivities to the same degree of preference change for both objectives. According to our investigation, the objective QC idle time's responses are not implicit when its preference weight is high. Such an issue is also verified by Table 5.5. For PAMOO, QC idle time only changes 0.65 along with the preference changing from (1.0, 0.0) to (0.7, 0.3), but when the preference weight for it is low (from (0.3, 0.7) to (0.0, 1.0)), the objective range is 2.72. For the outer loop method, these two values are 0.23 and 10.37, where the issue is more serious.

The basic reason that causes such an effect may be underlying and related to many factors of problem instances or the environment. For example, the number of trucks, task distribution among QCs, and the relative locations of import-export yards, all of these factors could lead to the effect that optimizing one of the objectives is harder, thus causing the final approximated Pareto front to be uneven. It is interesting to conduct a systematic sensitivity analysis of these factors and the connection between these factors in the problem environment, and the uneven degree of the final Pareto front is worth investigating.

In addition to the examined truck dispatching problem, for most of the real-world multi-objective optimization, the evenly distributed preferences usually do not have a straightforward connection to a regular and dense Pareto front. Increasing the number of preferences at the evaluation stage may alleviate such issues to some extent, but will lead to unrealistic computational cost. Not to mention some evolutionary approaches that require re-training (re-run) the entire algorithms to gain more Pareto solutions and are not allowed any preference adjustment after the training process. By contrast, our proposed methodology shows significant advantages for its flexibility to generate different numbers of trade-off policies under arbitrary

Table 5.7: Comparison Result between Even and Adjusted Preferences.

Method	80 Trucks			100 Trucks			120 Trucks		
	HV	Sparsity	Gap(%)	HV	Sparsity	Gap(%)	HV	Sparsity	Gap(%)
Evenly Preferences (11 Pref.)	0.684	253.03	5.79%	0.676	260.71	7.02%	0.679	267.98	7.24%
Evenly Preferences (21 Pref.)	0.709	64.48	2.34%	0.705	66.14	3.03%	0.709	68.91	3.14%
Evenly Preferences (51 Pref.)	0.723	11.42	0.41%	0.721	11.8	0.83%	0.724	11.06	1.09%
Adjusted Preferences (11 Pref.)	0.694	240.93	4.41%	0.683	240.13	6.05%	0.685	241.41	6.42%
Adjusted Preferences (21 Pref.)	0.716	58.29	1.38%	0.718	60.55	1.24%	0.719	60.9	1.78%
Adjusted Preferences (51 Pref.)	0.726	10.28	0.0%	0.727	10.29	0.0%	0.732	10.07	0.0%

preferences to approximate dense and even Pareto fronts and could further improve the Pareto front quality by adopting the preference calibration method.

5.9 Summary

In this work, we proposed a novel methodology for the multi-objective version container truck task dispatching problem in marine container terminals. The examined problem is solved by formulating it as a multi-objective Markov decision process (MOMDP) and introducing multi-objective deep reinforcement learning approaches. Benefited by the innovative network structure design, which adopts feature crossing operation to combine preference and state information, users are able to make online decisions under arbitrary trade-off preferences. Polices could be generated rapidly to explore the entire Pareto front.

Following the principles of traditional evolutionary multi-objective algorithms, the genetic programming-based NSGA-II and MOEA/D are developed to benchmark our proposed method. The comparative experiments towards the benchmarks demonstrated that our proposed method could outperform the benchmarks on solution quality, diversity, and sample efficiency.

In addition, the experiment also shows that the policy trained with only a small number of preferences could still generalize to other, more unseen preference cases and span a relatively dense and smooth approximated Pareto front. Based on these attributes of the methodology, a preference calibration method is proposed, which could rearrange the policy set to

obtain a more even approximated Pareto front by adjusting the preferences. By adopting the preference calibration method, the effect caused by unequal sensitivities towards different objectives could be eliminated to a large extent, thus further improving the quality of the approximated Pareto front.

Chapter 6

Mechanisms for Prior Expert Knowledge Augmentation

Based on the implementations of the previous research work, the issue of training efficiency has emerged. For the single-objective dispatching problem, the proposed method is faced with the issue of multi-scenarios which require the policy to be versatile to adapt to various cases. As for the MOO problem, an inner-loop method needs enough data samples of various preferences for an agent to generalize to unseen cases. In both cases, enough data records are needed in the training process to cover these various complex situations, which require numerous interactions with the environment. Therefore, the methods that utilize the pre-trained blocks are visited. Such methodology could be considered as an approach to introduce prior expert knowledge. Following such an idea, we proposed the expert network-assisted dispatching model and policy fusion approach, both for single and multi-objective optimization, which could alleviate the training efficiency issues to a large extent.

6.1 Motivations of Prior Expert Knowledge

This section briefly introduces the purpose and motivation of introducing the mechanisms of prior expert knowledge augmentation. For complex real-life decision-making problems, great effort may be taken for the RL agent to learn a sophisticated policy from scratch. Also, the hardship of training may boost along with the increasing state or action spaces. Therefore, mechanisms of domain expert knowledge are leveraged to alleviate such issues. Deep reinforcement learning-based hyper-heuristic (DRL-HH) approach (Zhang et al., 2022a) is one of the most typical methodologies. Specifically, the action space is not all the available dispatching tasks but a series of pre-designed low-level heuristics as mentioned in section 2.3. The overall framework of DRL-HH is described in Figure 6.1, where the RL agent selects the heuristic 1 as the action and then the heuristic outputs the specific task 3 in this case. By this means, the RL agent starts to be trained from a human-dispatching level because each heuristic is fine-tuned and the agent could still perform well even randomly selects actions. In addition, the DRL-HH accelerates the agent's subsequent learning process by reducing the action space, thus making it easy to learn a high-quality mapping between states and specific tasks. Indeed, the methodology of DRL-HH is significant, especially for real-world applications, because the performance of the agent could be enhanced by repeatedly introducing the newly designed dispatching heuristic based on practical experience, even though there's no breakthrough at the algorithmic level.

Despite the performance and practical effect of DRL-HH, such methodology still has its bottlenecks. First of all, the performance of DRL-HH highly relies on these low-level heuristics because it is hard to be improved by subsequent training with a poor-quality low-level heuristic. In other words,

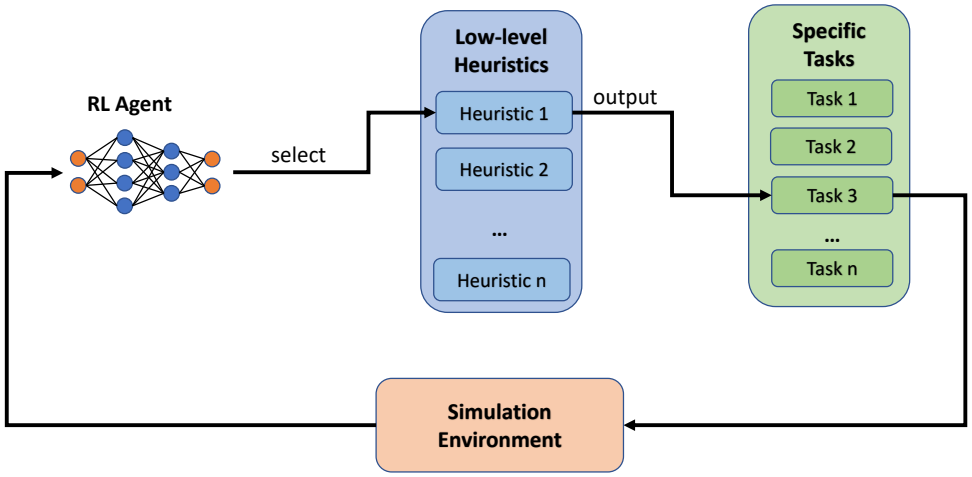


Figure 6.1: The illustration of deep reinforcement learning hyper-heuristic framework.

such an approach is limited by expert knowledge by introducing the expert knowledge. The second bottleneck is called the reachability issue. It means the decisions yielded by low-level heuristics usually cannot cover the entire decision space (space of all available tasks), which is illustrated by Figure 6.2. Ideally, each specific decision (task) should be reached by at least one low-level heuristic, but such an attribute is hard to achieve. Otherwise, the decision space is reduced at each decision-making step, and the policy obtained through such a manner is a lower or upper bound theoretically.

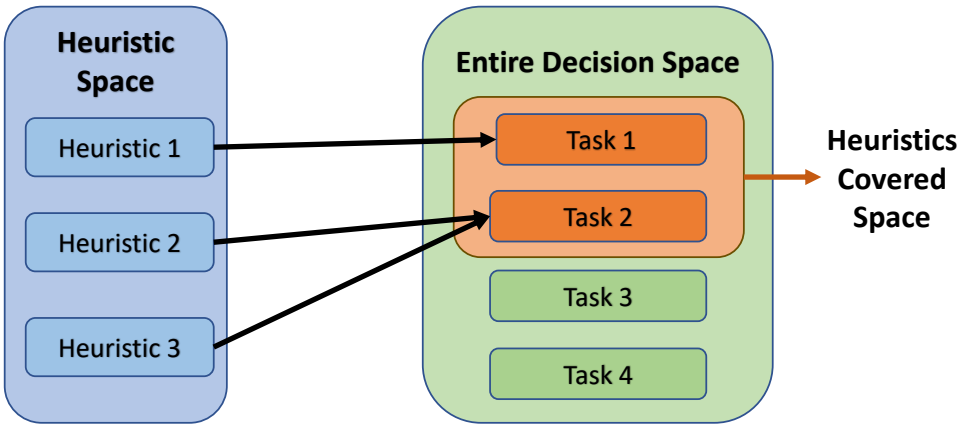


Figure 6.2: An example that explains the reachability issue of hyper-heuristics methodology.

The idea of introducing the domain knowledge has been widely explored in deep learning community. One of the classic approaches is to use pre-trained blocks in deep neural networks (Qiu et al., 2020). Pre-trained model indicates the network model that trained by large datasets of generic task in advance and the learned features could be transferred to other domain-specific tasks. The general concept of pre-trained model is to utilize large-scale datasets to initiate (a part of) parameters of the network then adopt the model to other tasks by fine-tuning or transfer learning. Several representative applications of pre-trained models include bidirectional encoder representation of transformer (BERT) (Kenton and Toutanova, 2019) and object detection techniques (YOLO: you only look once) (Redmon, 2016). However, most of these works focus on natural language processing (NLP) or computer vision (CV) domains, and research that studies the utilization of pre-trained models in the field of operations research is quite limited. Following such an idea, a pre-trained block obtained through imitation learning is introduced to solve the examined truck dispatching problem in this work. The purpose of this methodology is to maintain the augmentation of the prior expert knowledge in the model but avoid these two drawbacks of DRL-HH that are introduced above.

In the case of multi-objective optimization of the examined dispatching problem, training time is boosted even for one single scenario since the agent actually learn all the cases with different preference weights. Considering the multi-scenarios issue in MOO problem could be more challenging. Therefore, the idea of prior expert knowledge augmentation is also introduced to the MOO field. In this work, the polices that trained through single objective optimization environment are treated as expert knowledge. These two mechanisms for both single and multi-objective cases are described in the following sections.

6.2 Imitation Learning

In the case of approximating decisions, Imitation Learning (IL) is often used to learn a policy from demonstrations of expert behaviors (Bengio et al., 2021). In IL, the agent is not trained to maximize the reward, but to blindly mimic the expert through learning a mapping between observations and actions. Researchers have tried to combine IL with DRL to reduce the exploration costs of the agents (Silver et al., 2016). In this work, we also adopt IL in our framework in order to speed up the convergence of the RL agent and obtain some basic prior knowledge of the examined problem.

The parameters of the expert network are learned through a simple on-policy iterative supervised learning algorithm (as shown in Algorithm 3) where the expert policy is provided by a heuristic dispatching rule (Chen et al., 2016). Such a heuristic dispatching policy is scenario-independent and insensitive to environment uncertainties and thus should be suitable as prior knowledge for an RL agent. The loss is defined by the difference between the expert network output and the label (decision of the manual heuristic). For example, for a given state observation at some time step t, a manual heuristic would provide a unique answer for which task to select for this state (input). The expert network with the same structure in 4.2 outputs a probability distribution of the actions. Then the cross entropy between the network output (probability distribution) and the label (one-hot vector form) could be computed through the Function (6.1). The loss could be used to update the expert network parameters by backpropagation accordingly. In each iteration of imitation learning, the problem instance is randomly selected. Random selection makes the expert network cover more problem scenarios, thus making the embeddings extracted from the expert network more informative. The imitation mechanism could make

the agent converge to the heuristic level in a relatively short time period and consequently make the RL training process more stable. The detailed experimental results are presented in section 6.5.1 and 6.5.2 .

$$cross\_entropy(\boldsymbol{a}, \boldsymbol{p}) = -\sum_i a_i \log p_i \tag{6.1}$$

Algorithm 3: Imitation learning for truck dispatching optimization

Input: number of iterations I, steps per episode T
Initialize: a differentiable truck dispatch policy with random parameterization $\pi(a|s, \theta)$;
for *i=1 : I* **do**
  Randomly select a problem instance B_i;
  Collect an episode $s_0, p_0, s_1, p_1, \ldots, s_T, p_T$ from B_i, where p_t is probability distribution of the network output and s_t is the state representation, at time step t, following $\pi(\cdot|\cdot, \theta)$;
  Collect the actions $a_0, a_1, \ldots, a_T$ of each state $s_0, s_1, \ldots, s_T$ accordingly based on heuristic dispatching rule using one-hot representation;
  Calculate the loss as: $\mathcal{L} = \sum_{t=0}^{T} cross\_entropy(a_t, p_t)$;
  Update expert network parameters as: $\theta = Adam(\nabla\mathcal{L}, \theta)$;
end

6.3 Expert Network Assisted Dispatching Model

The policy network used in this work is depicted in Figure 6.3. Similar to the network described in section 4.2, given the state vector described in section 4.1.1, the policy network takes the QC features as the inputs and then outputs a list of probability distributions of the available actions.

Generally, the proposed network structure consists of an expert network and a cross-scenario network. The expert net is used for providing additional prior knowledge by extracting a feature vector for each QC, and

the cross-scenario net is used as the actor for the RL agent so that the probability distribution of actions is computed. For the expert network, the feature vectors of each QC are first fed into a three-layer bi-directional long short-term memory (LSTM). Next, the hidden states of each LSTM step are fed into a feed forward layer to generate a 512-dimensional feature vector for each QC, namely H^{Expert}. The H^{Expert} represents the knowledge of an expert dispatching policy, and it will be further fed into the cross-scenario network. The gate in the cross-scenario network is a two-layer fully connected block that takes the same input states and outputs a tensor with the same shape as H^{Expert}. The other parts of the cross-scenario network adopt the same structure as the expert network before the concatenation layer. After the concatenation is done, a new feature vector consisting of the information for both H^{Expert} and H^{Target} is generated. Then, an attention layer which has a similar structure to the self-attention block in Vaswani et al. (2017) is adopted and it maps the feature vector of each QC to a scalar. Finally, a softmax layer is used to generate the action probability distribution.

Similar to the network described in 4.2, such proposed network also treats the state vector of each QC as a dynamic set with spatial-temporal features. The spatial information of both truck and QCs is embedded in the state design (see section 4.1.1). The input data follows the fixed order based on QC's position in the terminal (same direction as the roads besides QCs). The use of LSTM is to make the network model capable of handling the dynamic size of the candidate QC. Once a QC finishes all of its tasks, it will be eliminated from the QC set. The bidirectional structure of LSTM ensures the information of the entire QC sequence is fully propagated at each step and thus capable of capturing some hidden features, such as one-way roads at the seaside. The gate component controls the information

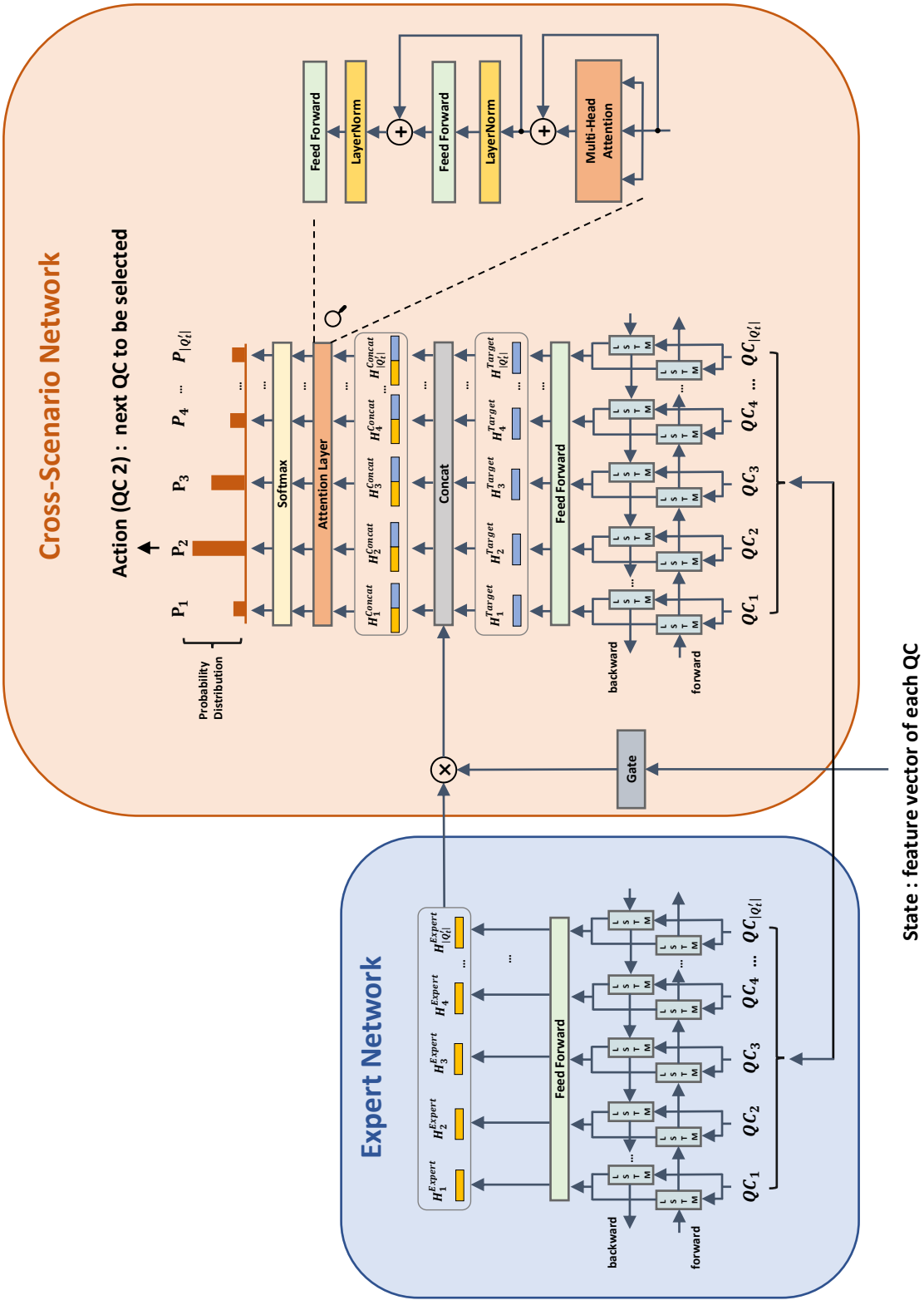


Figure 6.3: The expert network and cross-scenario network structure.

flow of the expert knowledge based on raw observation. With the help of the attention layer, each extracted QC feature vector is aware of the entire sequence of information, and the multi-head attention mechanism allows the agent to selectively focus on different parts of H^{Concat}. Thus, the agent can decide "how much" it may refer to the expert knowledge. Moreover, the capability of handling different lengths of input makes our model more competitive in general scenarios compared with the traditional structure like the fully connected network, since it would be impractical to train all possible problem instances with different input sizes separately. In addition, our proposed neural network structure could easily deal with invalid agent action (once a QC has dispatched all its tasks and becomes empty) by removing that QC from the input list.

The training process of the proposed expert network-assisted dispatching model is divided into two stages. In the first stage, the expert network is trained through imitation learning with the expert policy (Chen et al., 2016). Notably, to train the expert network, an extra fully-connected layer and a softmax layer are required to make it output a probability distribution, and these two layers are dropped after imitation learning to make it output embeddings. In the second stage, the network is trained through reinforcement learning. In this stage, only the parameters of the cross-scenario network are updated in training, and the expert network is fixed. Apart from the mechanism of expert network augmentation, other training details such as MDP modeling, algorithm, and hyper-parameters are all kept the same as described in chapter 4.

6.4 Policy Fusion

Following the similar idea that leverages the pre-trained blocks in the dispatching model, a prior expert knowledge augmentation mechanism is proposed for the MOO version of the examined truck dispatching problem. Such an approach treats well-trained policies for single objective optimization as expert knowledge. In this work, it is assumed that an MOO policy could benefit from the features extracted from the single objective policy and effectively adapt to the MOO case by the fusion of these features. Specifically, two single-objective optimization models for minimizing QC idle time and truck empty traveling distance separately are trained in advance. Then the QC features are extracted by these two network models in the same way as the expert network in section 6.3. The extracted QC features form the new fusional features as the inputs of the network for RL agent training, whose structure is similar to 5.3. Such a policy fusion process is indicated by Figure 6.4, where the main learner and preference embedding blocks are trained as the RL agent, and two single-objective policies serve as the expert networks. In general, the agent of the policy fusion model learns an MOO policy with the combined single objective QC features and the given preference weights. Indeed, the single objective policy in this framework could also be considered as a special feature engineering approach that transfers the raw observation to the kind of QC features that could be better exploited to train an MOO policy.

Similar to the methodology in section 6.3, the training process is divided to two stages. In the first stage, two networks that aim at minimizing two objectives respectively are trained. These two networks are trained through reinforcement learning, which is the same as the process in chapter 4. In the second stage, the network is also trained through reinforcement learning,

whose process and settings are the same as the methodology introduced in chapter 5. In this stage, only the parameters of the main learner and preference are updated during the training, while the parameters of the two single-objective policy blocks are fixed. The embeddings of two single-objective features are combined through concatenation. The information of preference and single-objective fusional features is combined through Hadamard product operation which is similar to PAMOO. Other settings and algorithms of training in this stage are the same as described in chapter 5. The proposed policy fusion approach obtained a significant improvement on the acceleration of the MOO policy training, and the results are reported in section 6.5.4.

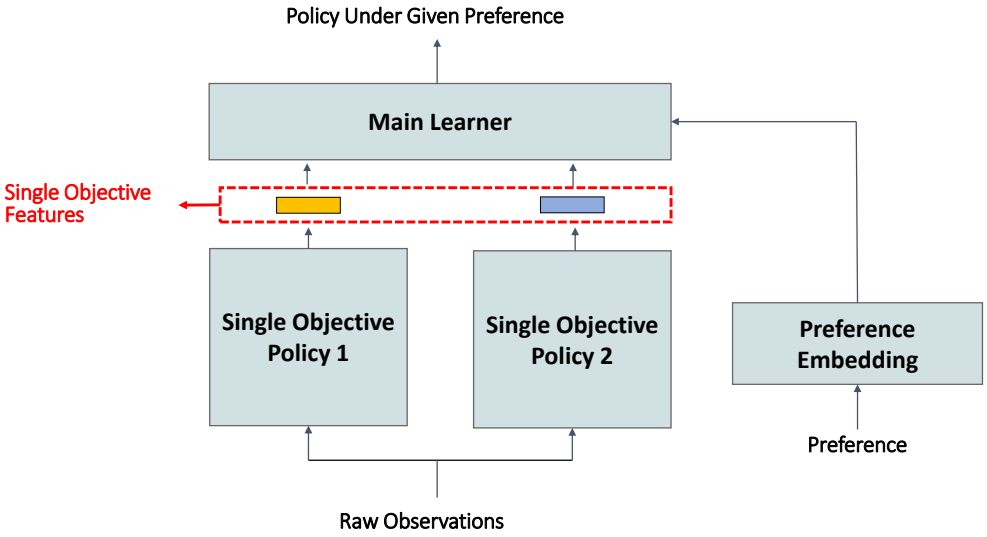


Figure 6.4: Illustration of policy fusion framework.

6.5 Experiments and Result Analysis

6.5.1 Result of Imitation Learning

Imitation learning is the approach that is used for extracting the prior expert knowledge representation in this work, which has been introduced in section 6.2. The policy in Chen et al. (2016) is selected as the expert behaviors for the imitation learning. The imitation learning in this work could be considered as a standard supervised learning of a classification problem. The tables 6.1 and 6.2 present the result of the imitation learning on both the training and testing sets respectively. The row of heuristic policy and imitated policy reports the result in terms of the objective values, gap shows the performance distance towards the heuristic policy, and Acc indicates the result of accuracy in the perspective of a classification problem.

The result is also based on the six different problem configurations that are designed in section 4.4. According to the results of imitation learning on both the training and testing sets, the agent is able to reproduce the expert behaviors on various problem configurations. The performance of the imitated policy is extremely close to the target heuristic approach, and there's no obvious overfitting occurrence in each case. Therefore, the network trained through such an imitation learning manner could be considered as the expert behavior, and the concrete expert knowledge is represented by the output vectors of the middle layer of the imitation network. For each state input, there is a knowledge representation (vectors) that indicates the information of the expert policy's decision logic when faced with this state input. One particular reason for choosing this heuristic as the expert

Table 6.1: Result of imitation learning on training sets.

Method	Config 1	Config 2	Config 3	Config 4	Config 5	Config 6	Average
Heuristic Policy	29890	23433	38889	48078	28200	34649	33857
Imitated Policy	30886	23098	40302	47153	28007	34698	34024
Gap (%)	3.33%	-1.43%	3.63%	-1.92%	-0.68%	0.14%	0.49%
Acc (%)	99.6%	99.68%	98.48%	98.31%	99.92%	99.82%	99.71%

Table 6.2: Result of imitation learning on testing sets.

Method	Config 1	Config 2	Config 3	Config 4	Config 5	Config 6	Average
Heuristic Policy	29470	24586	38915	48199	26693	31051	33152
Imitated Policy	29890	24594	40341	48300	26307	31228	33443
Gap (%)	1.43%	0.03%	3.66%	0.21%	-1.45%	0.57%	0.88%
Acc (%)	98.35%	99.2%	96.26%	98.11%	99.88%	99.82%	98.25%

policy is its robustness towards changing scenarios and ability to maintain comparable performance on each problem configuration.

6.5.2 Comparative Results of the Proposed DRL Approach with and without Imitation Learning

An ablation study is conducted to demonstrate the effectiveness of our proposed network structure in terms of both performance and speed of convergence. The evaluation metric and problem instance design follow the same settings in chapter 4. Table 6.3 shows the performance of our method with and without the expert network. For the agent without the expert network, a single cross-scenario network without the gate component is adopted (See Figure 4.2 for details). As we can see, compared with the single cross-scenario network, the agent with the expert network has better performance in most cases except Config 4. Using a single network to obtain a uniform policy for different scenarios (parameterized environments) can be problematic. A certain policy that works well for some particular scenarios may perform badly in others (e.g., Config 3) due to possible overfitting. Our approach alleviates such defects by incorporating prior knowledge into the RL agent. The results are more balanced among different environment configurations since the expert network is trained by scenario-insensitive data through imitation learning.

Figure 6.5 shows the convergent performance of the agent without the expert network, where the lines of random and heuristic are two dispatching policies that provide reference lines for the RL agent's convergence curve. Each step point represents the average score of all configurations (10 testing episodes for each). At the initial stages, the dispatching policy is even worse

than the random dispatching policy. After around 500 iterations, the DRL policy starts to outperform the benchmark heuristic method and achieves a better score steadily. In contrast, as can be found in Figure 6.6, the agent with the expert network converges rapidly at early stages and achieves the heuristic level after only 100 iterations. Benefit from the prior knowledge incorporated into the network, the agent uses only 650 iterations to obtain a better score than that of the agent without the expert network.

Table 6.3: The performance of the proposed method in comparison with or without the expert net.

Configuration	Heuristic Method (Benchmark)	Obj value / Imp (%) against benchmark	
		Without	With
Config 1	29470	27714 6.0%	26854 **8.9** %
Config 2	24586	21140 14.0%	20856 **15.2** %
Config 3	38915	36930 5.1%	35374 **9.1** %
Config 4	48199	42125 **12.6** %	42299 12.2%
Config 5	26693	24940 6.6%	24314 **8.9** %
Config 6	31051	26428 14.9%	26142 **15.8** %

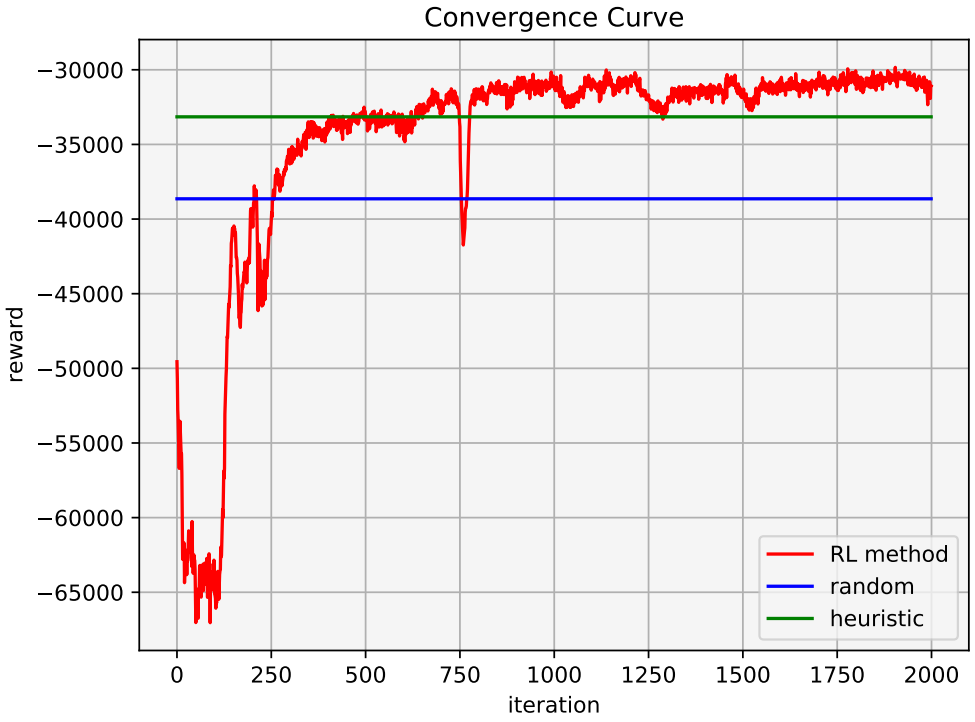


Figure 6.5: The convergence performance of the proposed method in comparison with the manual heuristic.

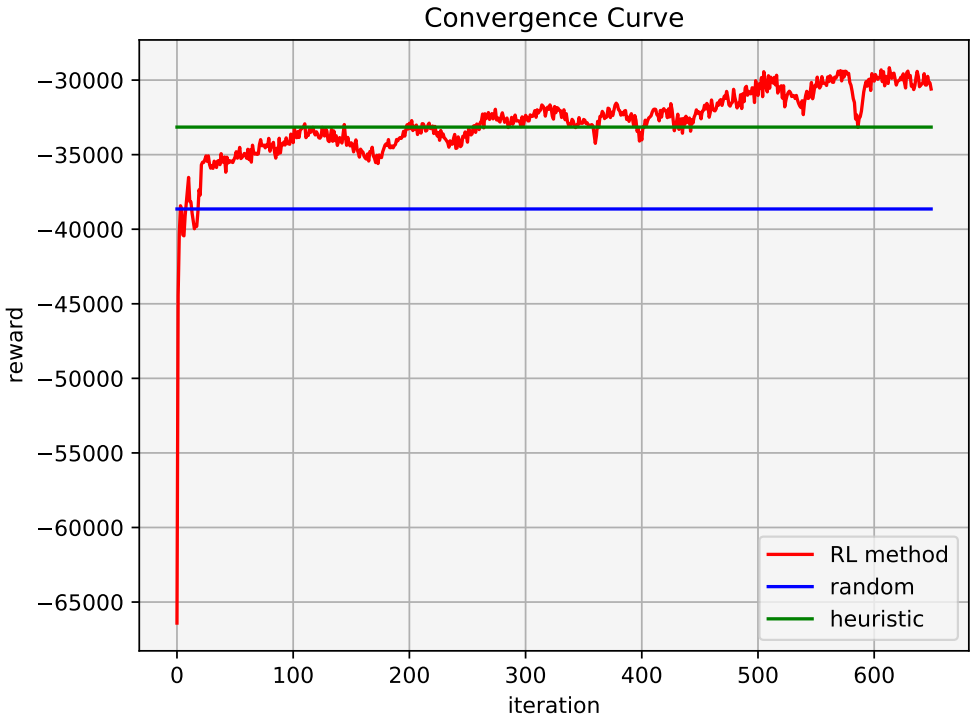


Figure 6.6: The convergence performance of the proposed method in comparison with the manual heuristic using Imitation Learning.

6.5.3 Comparative Results on various Uncertainties

Apart from the unseen problem instances, the proposed DRL method also shows robustness to unknown uncertainties. As introduced earlier, truck speed at different areas and crane operation time are the uncertain factors for the examined problem. In the training environment, crane operation times are non-deterministic, which follows the same setting of Zhang et al. (2022a) while the truck speed is assumed to be constant. In the testing environment, the truck speed is treated as an unknown uncertainty for the RL agent. The trained models for DRL-HH, GP, and our method are evaluated in the testing environment with unknown uncertainty. The comparison results against the manual heuristic can be found in Table 6.4.

It can be seen that DRL-HH is not competitive and fails to outperform

Table 6.4: The performance of the proposed method in comparison with the manual heuristic and a DRL-HH method under unknown uncertainties.

Configuration	Heuristic Method (Benchmark)	Obj value / Imp (%) against benchmark		
		GP	DRL-HH	Ours
Config 1	30469	30135 1.1%	30286 0.6%	29372 **3.6**%
Config 2	25869	24698 4.5%	25222 2.5%	23670 **8.5**%
Config 3	40268	39587 1.7%	41114 -2.1%	39140 **2.8**%
Config 4	50121	48352 3.5%	49169 1.9%	46261 **7.7**%
Config 5	28169	27332 3.0%	28338 -0.6%	26817 **4.8**%
Config 6	32118	31585 1.7 %	31411 2.2%	30769 **4.2**%

heuristic solutions in some cases. It is not surprising, as the unseen truck speed uncertainty at the yard side may further aggravate the yard congestion effect, and the low-level heuristics used for DRL-HH may limit its exploration ability. Nonetheless, the proposed DRL method still shows its great robustness to unknown uncertainties and outperforms the other methods in all testing instances. Since uncertainties cannot be enumerated and included in the training environment, the experimental results demonstrate that our proposed method has the potential to be deployed in the real-world port operation environment.

6.5.4 Result of Policy Fusion

The experiment settings in this section are basically the same as the description in chapter 5. Table 6.5 presents the convergence results of the original MOO truck dispatching policy and the policy fusion dispatching approach for cases with various truck numbers. The policy fusion is trained through the features extracted from the single objective policies, which follow the mechanism in section 6.4. Each single objective policy (network) is trained in the same way as described in chapter 4. The Original MOO policy in Table 6.5 indicates the methods that are trained as the way in chapter 5. Several iteration steps are selected as the intermediate status

to observe the convergence performance of the proposed method and the benchmark. Specifically, the hyper volume (HV) results in iterations 5, 10, 20, 50, 100, and 200 of both methods are reported. Obviously, the policy fusion approach converges much more rapidly than the original MOO policy. It starts from a performance level that the original MOO policies take about 100 iterations to achieve. Basically, it skipped the early stage that the MOO policy used to "warm up" the initial training process, which benefits from the prior expert knowledge incorporation of single-objective policies. Moreover, the original MOO policies are not well-trained within 200 iterations, while the policy fusion models have already outperformed the original MOO policies and achieved the optimal performance.

Table 6.5: Convergent results between original MOO approach and policy fusion method.

Methods	Iterations					
	5	10	20	50	100	200
80 Trucks						
Original MOO Policy	0.084	0.353	0.553	0.692	0.832	0.897
Policy Fusion	0.753	0.832	0.881	0.902	0.927	0.933
100 Trucks						
Original MOO Policy	0.046	0.129	0.302	0.491	0.672	0.753
Policy Fusion	0.694	0.759	0.832	0.897	0.923	0.952
120 Trucks						
Original MOO Policy	0.067	0.297	0.586	0.741	0.791	0.877
Policy Fusion	0.701	0.821	0.916	0.948	0.957	0.961

The Figure 6.7 shows the approximate Pareto front obtained by the proposed policy fusion and original MOO approach, both with 11 evenly distributed preferences. In this experiment, both policies are fully trained with adequate iterations of updates. Visually, the approximate Pareto front of both methods is comparable and relatively even and regular. The result demonstrated the feasibility of a multi-objective truck dispatching policy obtained by the features extracted from single-objective policies.

The Table 6.6 and 6.7 report the specific numerical result of non-dominant policies and approximated Pareto front in Fig 6.7 respectively. Overall, both the original MOO approach (PAMOO) and policy fusion approach generate 11 non-dominant policies out of their 11 preference inputs. The hyper volumes (HV) of the policy fusion and original MOO approaches are 0.74 and 0.786 respectively. The policy fusion maintains a 5.85% gap towards the original MOO method. The advantage of the policy fusion lies in its convergence efficiency at the expense of a quality drop in the obtained policy set. The HV of policy fusion is not as good as PAMOO may because it only uses the extracted feature embeddings, while lacking original observation information. Notably, the magnitude relation of HV is only meaningful in one single comparative experiment because the min-max scalars in each comparison are different.

A particular phenomenon is distinct in this experiment. Although the overall HV performance of policy fusion is inferior to the original MOO method, the result of two extreme cases (preference with (1.0, 0.0) and (0.0, 1.0)) shows more competitiveness. If the QC idle time and the empty travel time are considered individually, the policy fusion method both obtained better results. The features that the policy fusion model takes as input are obtained through extreme cases by single objective policies, which makes the fusion model tend to gain more ready-made information when the input preference weights are closer to the extreme cases. In contrast, the fusion policies with middle preference weights (like (0.5, 0.5)) are relatively poor compared with the others because more effort is required for the agent to learn how to merge the information from two kinds of extreme cases to achieve particular trade-off purposes. These experimental results also provide more insights that not only the policies for single objectives are important. More information with some typical combinations of pref-

erence weights is promising to further promote the policy fusion model's performance.

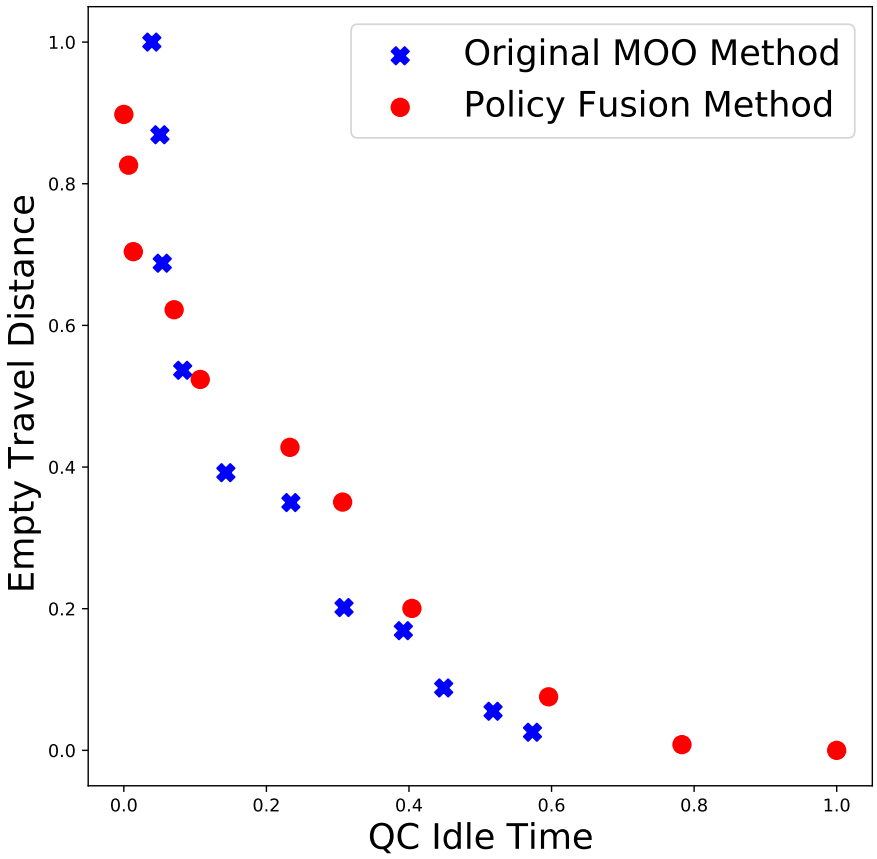


Figure 6.7: Pareto frontiers generated by original MOO method and proposed policy fusion method on instance of 120 trucks.

6.6 Summary

This section provides two mechanisms that introduce the prior expert knowledge to augment the dispatching policies for both single and multi-objective optimization of the examined problem. The purpose of doing so is to avoid the agent being trained from scratch, thus improving the convergence speed of the agents. Moreover, as another methodology that

Table 6.6: Numerical Result of Policy Fusion Method and Original MOO Method.

Preferences	Policy Fusion Method		Original MOO Method	
	QC Idle Time (min/QC)	Empty Travel Distance (m/task)	QC Idle Time (min/QC)	Empty Travel Distance (m/task)
(0.0, 1.0)	27.47	492.5	21.06	500.34
(0.1, 0.9)	24.21	494.98	20.23	509.42
(0.2, 0.8)	21.4	515.68	19.19	519.52
(0.3, 0.7)	18.52	553.95	18.34	544.31
(0.4, 0.6)	17.06	599.98	17.09	554.33
(0.5, 0.5)	15.95	623.68	15.97	599.77
(0.6, 0.4)	14.06	653.07	14.6	612.71
(0.7, 0.3)	13.51	683.25	13.69	657.03
(0.8, 0.2)	12.65	708.39	13.26	703.38
(0.9, 0.1)	12.55	745.82	13.21	759.0
(1.0, 0.0)	12.45	767.81	13.04	799.11

Table 6.7: Comparison of approximated Pareto fronts obtained by original MOO method (PAMOO) and policy fusion method.

Methods	HV	|Sols|	Sparsity (10^{-4})	Gap(%)
Policy Fusion Method	0.74	11	248.69	5.85%
Original MOO Method	0.786	11	163.56	0%

utilizes the domain expert knowledge, hyper-heuristic still maintains some drawbacks such as the performance upper bound limited by the performance of low-level heuristics and the reachability problem. Compared with the DRL-HH method in (Zhang et al., 2022a), our proposed mechanisms provide another way to avoid these drawbacks. Specifically, an expert network is trained through an imitation learning manner guided by a fine-tuned manual heuristic policy (Chen et al., 2016) as the expert behavior. Consequently, the output of such an expert network is considered as the representation of the prior expert knowledge and serves as referenced information for the agent during the process of RL training. By this means, the RL agent not only exploits information of expert behavior but also maintains an adequate exploration space for the agent to play a role and avoid reducing the decision space. We first experimentally prove that the manual heuristic policy (Chen et al., 2016) is a suitable approach to serve as the expert behavior, and our imitation network is able to reproduce the dispatching logic of the expert policy. Then, an expert network-assisted dispatching model is proposed for the RL training. The experimental results demonstrate its performance on faster and more stable convergence and a degree of robustness towards the unknown uncertainties.

Following a similar practice, such a mechanism is also extended to the multi-objective version of the examined dispatching problem. In the MOO domain, the original dispatching model is formulated as a uniform dispatching policy that takes the state and particular preference weight as input and gives the dispatching decision based on arbitrary objective pref-

erence. However, such a method requires the agent to learn all policies based on different preferences simultaneously. Even for one single scenario, the MOO problem takes several times' training cost compared with the single objective version. Therefore, the single objective dispatching policies are considered as the appropriate expert polices to generate expert knowledge. Based on such concerns, we proposed the methodology "policy fusion" which takes the features extracted from each single objective dispatching model as input and rapidly generalizes to an MOO dispatching policy. The experimental result shows that the proposed policy fusion approach gains an order of magnitude improvement in the convergence speed. Generally, the prior expert knowledge augmentation mechanisms show advantages in terms of convergence speed, robustness, and performance gain.

Chapter 7

Conclusion and Future Work

This section concludes the main works in this thesis as a whole and deeply deciphers the contributions to the fields and some potential areas of improvement. The thesis starts from a real-life truck dispatching optimization problem in a marine container terminal, which is always considered as a primary focus in port daily operational management since it not only connects several different operation optimizations but also greatly affects the utilization of port equipment. The truck dispatching optimization then be considered as a sequential decision-making process and formulated as a Markov decision process to make it solvable by traditional reinforcement learning approaches. Several improvements, such as a more sophisticated feature design, a sparse reward mechanism, and the special network structure that masks the invalid actions, further enhance the dispatching policy towards the benchmarks. Furthermore, such a problem is extended to the multi-objective optimization version. A tailor-made network is designed that properly merges the information of both raw observation and preference weight, which makes it a uniform dispatching policy that could give decisions under arbitrary user preferences. Finally, two mechanisms, namely,

imitation learning-assisted expert network and policy fusion method that introduce the prior expert knowledge to augment the dispatching model for both single and multi-objective optimization of the examined problem are proposed. Significant performance gains have been obtained in terms of convergence speed, stability of training, and robustness towards uncertainties. Some key points among these works that deserve further discussion and are being highlighted are introduced in the remaining part of the section.

7.1 Conclusion

In this section, the main contributions of the proposed methodologies are deeply analyzed and concluded. The practicability of the proposed Real2Sim framework is first introduced. Then, a new paradigm for the MOO field is formed, which benefits from the attributes of the proposed PAMOO methodology. Finally, the significance of the methodologies that introduce the prior expert knowledge augmentation and the advantages it brings to the operations research fields are discussed.

7.1.1 Practicability of the Real2Sim Framework

In this thesis, the developed Real2Sim framework is the platform for the subsequent experiments and discoveries, which serve as the cornerstone of our contributions. Generally, this Real2Sim framework has the following advantages in practice.

- Real2Sim provides a safe and controllable training environment. To train and verify policies in the simulation environment could reduce the risk in the real-world environment, such as damage of equipment and economic loss. Also, it avoids the cost of collecting data in the real-world environment, which is sometimes even infeasible. In a simulation environment, rich problem scenarios and experimental conditions are easier to design and implement, which helps to construct versatile dispatching policies.

- The proposed Real2Sim framework reduces the gap between the real-world and experimental environment. Real2Sim aims to properly reproduce the details and logic of the examined container truck dispatching process and formulate them in the digital simulation. To achieve such a purpose, we deeply investigate the physical components, operation logic, and various types of uncertainties in a real container terminal, which are used as the ingredients for implementing the simulation. Moreover, historical data of the container terminal is fully used for generating the related problem scenarios.

- A policy obtained from the proposed Real2Sim platform tends to maintain its performance in reality as much as possible. Also, a fine-tuned manual heuristic dispatching method, whose performance is verified in reality, is selected as the expert policy to extract the prior knowledge to augment the RL dispatching model. By such means, the trained policy is guided by the operations logic, historical data, scenarios, and human experiment from the real container terminal, which guarantees the practical performance of the policy in reality to the greatest extent.

To sum up, the proposed Real2Sim framework shows more practicality

towards the traditional methods and is purely based on the mathematical model. It also has the potential to be the infrastructure for the digital and intelligent development of some related real-world industries.

7.1.2 New Paradigm for Online Multi-objective Optimization and Application

In the work of MOO, a learning approach that could generate a single uniform dispatching policy that gives decisions under arbitrary user preferences is proposed. The proposed methodology provides a creative way for real-world multi-objective online optimization. It satisfies the short response time of online decision making, has the ability to overcome the uncertainties in the environment, and also provides adequate and diverse trade-off policies for users. Online decision-making management could be promoted in terms of both priori and posteriori views of multi-objective optimization. For priori view, a user is aware of the expected trade-off and the possible outcomes for a particular real-world circumstance. However, users are sometimes only aware of the specific trade-off purpose but cannot confirm the exact numerical preference weight. Therefore, the result may somehow bias the user's expectation. By adopting our methodology, users are able to repeatedly fine-tune their decisions (preference weights) in real-time until the ideal performance is reached. Such a manner is called the interactive scheme for obtaining the most desired solutions in the MOO field. Traditional MOO methods are required to solve another single-objective optimization problem based on the new weights from scratch once the user adjusts the preference. In contrast, our proposed method is able to generate the policy of the new preference weight without re-running the algorithm by interactively obtaining the target solution for the user.

In posteriori cases, users have no expected preference but are required to provide a set of Pareto optimal solutions for selection. Thanks to the generalization of our proposed methodology, the model could yield various numbers of policies by refining the preference weights. Moreover, by using the preference calibration method, a high-quality Pareto front could be provided. To the best of our knowledge, there is no similar preference calibration method for MOO that could adjust the solution set as a whole to obtain a more even and smooth Pareto front. To sum up, for both priori and posteriori views, our proposed method provides new paradigms for the online MOO and the ways to obtain the solutions for users.

7.1.3 Significance of Prior Expert Knowledge Augmentation

Prior expert knowledge augmentation is the topic that goes through the entire PhD research career. For the single objective truck dispatching problem, an expert network-assisted dispatching policy generated by leveraging imitation learning is designed. Such a mechanism could also be further extended to the combination of knowledge of several experts that dedicated to different typical scenarios, which could further enhance the agent's adaptability for more problem cases. For the MOO version of the examined problem, a methodology called policy fusion, which follows a similar idea is proposed. The prior expert knowledge augmentation for MOO problems could be promoted to the field of many-objective optimization. The time cost for training would be exponentially boosted along with the objective number increased, while our proposed policy fusion could maintain the time complexity at an acceptable level since it obtained an order of magnitude on the convergence speed. Therefore, it is promising that the proposed

policy fusion approach could obtain more performance gains in the cases of many objectives.

The overall purpose of the pre-trained model is to make the agent grasp some general knowledge (indicating the expert heuristic dispatching method and single objective optimization ability in this thesis) before learning the policy of the examined problems. Apart from the utilities that avoid some drawbacks of hyper-heuristic approaches and the acceleration effect for the agent training, the greatest significance lies in leveraging the methodologies of pre-trained models, which are already maturing techniques in deep learning fields. From this perspective, this work bridges a connection between online optimization and the traditional deep learning fields. Some powerful methodologies in natural language processing (NLP) or computer vision (CV) domains could also be adopted as prior expert knowledge augmentation approaches for online optimization problems.

7.2 Limitations and Future Work

The methodology for single-objective dispatching could be further improved in several ways. Firstly, the state design in the MDP modeling of this work is over-simplified, where most state features are statistical information that cannot fully describe the entire system. Take the feature vector TH_t (total amount of trucks heading to each QC) as one example, it only includes the truck number for each QC but ignores the specific status of each truck, such as accurate positions and specific tasks being executed. Actually, the state in the MDP, with simple feature engineering, only takes up a small proportion of the conditions of the entire container terminal. One promising way to tackle such issues is to model the state by a graph neural network

(GNN), where all information about trucks, tasks, and QCs, such as the accurate locations and moving directions of trucks and cranes is included. Such an approach could maximize the end-to-end feature extraction potential of deep neural networks (Wu et al., 2020). We are quite confident that more performance gain could be obtained by adopting GNN.

Secondly, the proposed state design rarely takes the considerations about the constraints of the problem into account. In this work, QCs are constrained to execute tasks in order, which sometimes can cause QC's waiting even if there are already trucks in the queue. According to our investigation, such constraints would cause considerable amount of QC idle time even with a well-trained dispatching policy deployed. Few studies focus on improving RL agent's constraint awareness for COPs in the literature, and it should be a promising future direction.

Apart from some potential improvements for the target truck dispatching problem, there are several variants or modifications for the examined problem that may further reduce the quay crane idle time or bring more beneficial insights into container terminal management. The yard crane is another important component that can affect the objective value of the examined problem. In this work, the yard crane schedule simply follows "first come first serve" policy. However, there is still some space to reduce the total QC idle time by finding a more sophisticated yard crane scheduling policy. Moreover, the integration of truck dispatching and yard crane scheduling optimization can be a potential future direction of this work.

The proposed MOO methodology could also be extended in several ways. Firstly, as described above, each objective has a different degree of sensitivity with respect to preference changes, which leads to an uneven and

irregular Pareto front. The factors that influence the objectives' sensitivity both in problem instance settings and in the environment should be investigated and such effects may also have benefits to the reward design in return. Secondly, more objectives could be taken into consideration. Along with the increase of the objective number, the combinations of preference weights would explode, which makes higher requirements for the convergence efficiency and generalization performance of the proposed algorithms. The corresponding method could be further improved by adopting more advanced network structures or further leveraging the policy fusion techniques. Finally, as a promising generic online multi-objective optimization scheme, it is expected to be promoted to more general combinatorial optimization problems, such as online job shop scheduling or online bin packing, to further demonstrate its extensive applicability.

Bibliography

Abels, A., Roijers, D., Lenaerts, T., Nowé, A., and Steckelmacher, D. (2019). Dynamic weights in multi-objective deep reinforcement learning. In *International conference on machine learning*, pages 11–20. PMLR.

Afrapoli, A. M., Tabesh, M., and Askari-Nasab, H. (2019). A multiple objective transportation problem approach to dynamic truck dispatching in surface mines. *European Journal of Operational Research*, 276(1):331–342.

Agra, A. and Rodrigues, F. (2022). Distributionally robust optimization for the berth allocation problem under uncertainty. *Transportation Research Part B: Methodological*, 164:1–24.

Ahmed, L., Mumford, C., and Kheiri, A. (2019). Solving urban transit route design problem using selection hyper-heuristics. *European Journal of Operational Research*, 274(2):545–559.

Angeloudis, P. and Bell, M. G. (2011). A review of container terminal simulation models. *Maritime Policy & Management*, 38(5):523–540.

Aydin, M. E. and Öztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 33(2-3):169–178.

Bae, H. Y., Choe, R., Park, T., and Ryu, K. R. (2011). Comparison of operations of AGVs and ALVs in an automated container terminal. *Journal of Intelligent Manufacturing*, 22(3).

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR.

Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

Bellman, R. (1966). Dynamic programming. *science*, 153(3731):34–37.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2017). Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations*.

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421.

Bielli, M., Boulmakoul, A., and Rida, M. (2006). Object oriented model for container terminal distributed simulation. *European Journal of Operational Research*, 175(3):1731–1751.

Bin, L., Wenfeng, L., and Yu, Z. (2008). Agent-based modeling and simulation for vehicle dispatching at container terminals. *Journal of system simulation*, 20(19):5158–5161.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Bose, J., Reiners, T., Steenken, D., and Voß, S. (2000). Vehicle dispatching at seaport container terminals using evolutionary algorithms. In *Proceedings of the 33rd annual Hawaii international conference on system sciences*, pages 10–pp. IEEE.

Bucur, P. A. and Hungerländer, P. (2017). A reinforcement learning approach for the dynamic container relocation problem.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

Cai, B., Huang, S., Liu, D., Yuan, S., Dissanayake, G., Lau, H., and Pagac, D. (2012). Multiobjective optimization for autonomous straddle carrier scheduling at automated container terminals. *IEEE Transactions on Automation Science and Engineering*, 10(3):711–725.

Cao, J., Shi, Q., and Lee, D.-H. (2008). A decision support method for truck scheduling and storage allocation problem at container. *Tsinghua Science & Technology*, 13:211–216.

Cao, J. X., Lee, D.-H., Chen, J. H., and Shi, Q. (2010). The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. *Transportation Research Part E: Logistics and Transportation Review*, 46(3):344–353.

Carlo, H. J., Vis, I. F., and Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. *European journal of operational research*, 235(2):412–430.

Castelletti, A., Pianosi, F., and Restelli, M. (2011). Multi-objective fitted q-iteration: Pareto frontier approximation in one single run. In *2011*

International Conference on Networking, Sensing and Control, pages 260–265. IEEE.

Chang, D., Jiang, Z., Yan, W., and He, J. (2011). Developing a dynamic rolling-horizon decision strategy for yard crane scheduling. *Advanced Engineering Informatics*, 25(3):485–494.

Che, A., Wang, Z., and Zhou, C. (2024). Multi-agent deep reinforcement learning for recharging-considered vehicle scheduling problem in container terminals. *IEEE Transactions on Intelligent Transportation Systems*.

Chen, C., Hu, Z.-H., and Wang, L. (2021). Scheduling of AGVs in automated container terminal based on the deep deterministic policy gradient (DDPG) using the convolutional neural network (CNN). *Journal of Marine Science and Engineering*, 9(12):1439.

Chen, J., Bai, R., Dong, H., Qu, R., and Kendall, G. (2016). A dynamic truck dispatching problem in marine container terminal. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE.

Chen, L., Langevin, A., and Lu, Z. (2013). Integrated scheduling of crane handling and truck transportation in a maritime container terminal. *European Journal of Operational Research*, 225(1):142–152.

Chen, M., Beutel, A., Covington, P., Jain, S., Belletti, F., and Chi, E. H. (2019a). Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464.

Chen, M., Gao, L., Chen, Q., and Liu, Z. (2020a). Dynamic partial re-

moval: A neural network heuristic for large neighborhood search. *arXiv preprint arXiv:2005.09330*.

Chen, X., Bai, R., Qu, R., and Dong, H. (2022). Cooperative double-layer genetic programming hyper-heuristic for online container terminal truck dispatching. *IEEE Transactions on Evolutionary Computation*, 27(5):1220–1234.

Chen, X., Bai, R., Qu, R., Dong, H., and Chen, J. (2020b). A data-driven genetic programming heuristic for real-world dynamic seaport container terminal truck dispatching. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Chen, X., Bai, R., Qu, R., Dong, J., and Jin, Y. (2024). Deep reinforcement learning assisted genetic programming ensemble hyper-heuristics for dynamic scheduling of container port trucks. *IEEE Transactions on Evolutionary Computation*.

Chen, X., Ghadirzadeh, A., Björkman, M., and Jensfelt, P. (2019b). Meta-learning for multi-objective reinforcement learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 977–983. IEEE.

Chen, X. and Tian, Y. (2019). Learning to perform local rewriting for combinatorial optimization. In *Adv. Neural Inf. Process. Syst.*, volume 32, pages 6278–6289.

Choe, R., Kim, J., and Ryu, K. R. (2016). Online preference learning for adaptive dispatching of AGVs in an automated container terminal. *Applied Soft Computing*, 38:647–660.

Choi, H. R., Park, B. K., Lee, J., and Park, C. (2011). Dispatching of

container trucks using genetic algorithm. In *The 4th International Conference on Interaction Sciences*, pages 146–151. IEEE.

Choo, E. U. and Atkins, D. R. (1983). Proper efficiency in nonconvex multicriteria programming. *Mathematics of Operations Research*, 8(3):467–470.

Cordeau, J.-F., Laporte, G., Legato, P., and Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem. *Transportation science*, 39(4):526–538.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812.

Cui, T., Ding, S., Jin, H., and Zhang, Y. (2023). Portfolio constructions in cryptocurrency market: A CVaR-based deep reinforcement learning approach. *Economic Modelling*, 119:106078.

Cui, T., Du, N., Yang, X., and Ding, S. (2024). Multi-period portfolio optimization using a deep reinforcement learning hyper-heuristic approach. *Technological Forecasting and Social Change*, 198:122944.

da Costa, P. R. d. O., Rhuggenaath, J., Zhang, Y., and Akcay, A. (2020). Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. *arXiv preprint arXiv:2004.01608*.

de Carvalho, J. P. and Dimitrakopoulos, R. (2021). Integrating production planning with truck-dispatching decisions through reinforcement learning while managing uncertainty. *Minerals*, 11(6):587.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., and Rousseau, L.-M. (2018). Learning heuristics for the TSP by policy gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, pages 170–181. Springer.

Dkhil, H., Yassine, A., and Chabchoub, H. (2017). Multi-objective optimization of the integrated problem of location assignment and straddle carrier scheduling in maritime container terminal at import. *Journal of the Operational Research Society*, pages 1–23.

Dkhil, H., Yassine, A., and Chabchoub, H. (2018). Multi-objective optimization of the integrated problem of location assignment and straddle carrier scheduling in maritime container terminal at import. *Journal of the Operational Research Society*, 69(2):247–269.

Dragović, B., Zrnić, N., Dragović, A., Tzannatos, E., and Dulebenets, M. A. (2024). A comprehensive bibliometric analysis and assessment of high-impact research on the berth allocation problem. *Ocean Engineering*, 300:117163.

Drungilas, D., Kurmis, M., Senulis, A., Lukosius, Z., Andziulis, A., Januteniene, J., Bogdevicius, M., Jankunas, V., and Voznak, M. (2023). Deep reinforcement learning based optimization of automated guided vehicle time and energy consumption in a container terminal. *Alexandria Engineering Journal*, 67:397–407.

Dulebenets, M. A. (2016). A new simulation model for a comprehensive

evaluation of yard truck deployment strategies at marine container terminals. *Open Science Journal*, 1(3).

Ehrgott, M. and Gandibleux, X. (2003). Multiobjective combinatorial optimization—theory, methodology, and applications. In *Multiple criteria optimization: State of the art annotated bibliographic surveys*, pages 369–444. Springer.

Expósito-Izquierdo, C., Melián-Batista, B., and Moreno-Vega, J. M. (2014). A domain-specific knowledge-based heuristic for the blocks relocation problem. *Advanced Engineering Informatics*, 28(4):327–343.

Fazlollahtabar, H. and Saidi-Mehrabad, M. (2015). Methodologies to optimize automated guided vehicle scheduling and routing problems: a review study. *Journal of Intelligent & Robotic Systems*, 77:525–545.

Forum, I. T. (2021). *ITF Transport Outlook 2021*.

Fotuhi, F., Huynh, N., Vidal, J. M., and Xie, Y. (2013). Modeling yard crane operators as reinforcement learning agents. *Research in transportation economics*, 42(1):3–12.

Galle, V., Barnhart, C., and Jaillet, P. (2018). Yard crane scheduling for container storage, retrieval, and relocation. *European Journal of Operational Research*, 271(1):288–316.

Gao, L., Chen, M., Chen, Q., Luo, G., Zhu, N., and Liu, Z. (2020). Learn to design the heuristics for vehicle routing problem. *arXiv preprint arXiv:2002.08539*.

Gao, Y., Chang, D., and Chen, C.-H. (2023). A digital twin-based approach for optimizing operation energy consumption at automated container terminals. *Journal of Cleaner Production*, 385:135782.

Grafelmann, M., Nellen, N., and Jahn, C. (2023). Reinforcement learning at container terminals: A literature classification. In *Interdisciplinary Conference on Production, Logistics and Traffic*, pages 147–159. Springer.

Grieves, M. W. (2005). Product lifecycle management: the new paradigm for enterprises. *International Journal of Product Development*, 2(1-2):71–84.

Grunow, M., Günther, H.-O., and Lehmann, M. (2007). Strategies for dispatching AGVs at automated seaport container terminals. In *Container terminals and cargo systems*, pages 155–178. Springer.

Guan, Y. and Cheung, R. K. (2004). The berth allocation problem: models and solution methods. *Or Spectrum*, 26(1):75–92.

Guo, L., Wang, J., and Zheng, J. (2021). Berth allocation problem with uncertain vessel handling times considering weather conditions. *Computers & Industrial Engineering*, 158:107417.

Guo, X., Huang, S. Y., Hsu, W. J., and Low, M. Y. H. (2011). Dynamic yard crane dispatching in container terminals with predicted vehicle arrival information. *Advanced Engineering Informatics*, 25(3):472–484.

Gupta, A., Roy, D., de Koster, R., and Parhi, S. (2017). Optimal stack layout in a sea container terminal with automated lifting vehicles. *International Journal of Production Research*, 55(13):3747–3765.

Gutin, G. and Punnen, A. P. (2006). *The traveling salesman problem and its variations*, volume 12. Springer Science & Business Media.

Hakan Akyüz, M. and Lee, C.-Y. (2014). A mathematical formulation and efficient heuristics for the dynamic container relocation problem. *Naval Research Logistics (NRL)*, 61(2):101–118.

Hammersley, J. (2013). *Monte carlo methods*. Springer Science & Business Media.

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., et al. (2022). A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–59.

He, J., Huang, Y., Yan, W., and Wang, S. (2015). Integrated internal truck, yard crane and quay crane scheduling in a container terminal considering energy consumption. *Expert Systems with Applications*, 42(5):2464–2487.

He, J., Zhang, W., Huang, Y., and Yan, W. (2013). A simulation optimization method for internal trucks sharing assignment among multiple container terminals. *Advanced Engineering Informatics*, 27(4):598–614.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Helsgaun, K. (2017). An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B.,

Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. (2018). Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Hochreiter, S. (1997). Long short-term memory. *Neural Computation MIT-Press*.

Homayouni, S. M. and Tang, S. H. (2013). Multi objective optimization of coordinated scheduling of cranes and vehicles at container terminals. *Mathematical Problems in Engineering*, 2013.

Hottung, A., Tanaka, S., and Tierney, K. (2020). Deep learning assisted heuristic tree search for the container pre-marshalling problem. *Computers Operations Research*, 113:104781.

Hottung, A. and Tierney, K. (2020). Neural large neighborhood search for the capacitated vehicle routing problem. In *ECAI 2020*, pages 443–450. IOS Press.

Howard, R. A. (1960). Dynamic programming and markov processes. *MIT Press google schola*, 2:39–47.

Hsu, H.-P., Tai, H.-H., Wang, C.-N., and Chou, C.-C. (2021). Scheduling of collaborative operations of yard cranes and yard trucks for export containers using hybrid approaches. *Advanced Engineering Informatics*, 48:101292.

Hu, H., Yang, X., Xiao, S., and Wang, F. (2023). Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning. *International Journal of Production Research*, 61(1):65–80.

Hu, X., Guo, J., and Zhang, Y. (2019). Optimal strategies for the yard truck

scheduling in container terminal with the consideration of container clusters. *Computers & Industrial Engineering*, 137:106083.

Imai, A., Chen, H. C., Nishimura, E., and Papadimitriou, S. (2008). The simultaneous berth and quay crane allocation problem. *Transportation Research Part E: Logistics and Transportation Review*, 44(5):900–920.

Imai, A., Nagaiwa, K., and Tat, C. W. (1997). Efficient planning of berth allocation for container terminals in Asia. *Journal of Advanced transportation*, 31(1):75–94.

Imai, A., Nishimura, E., and Papadimitriou, S. (2001). The dynamic berth allocation problem for a container port. *Transportation Research Part B: Methodological*, 35(4):401–417.

Imai, A., Nishimura, E., and Papadimitriou, S. (2003). Berth allocation with service priority. *Transportation Research Part B: Methodological*, 37(5):437–457.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ishibuchi, H., Akedo, N., and Nojima, Y. (2014). Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Transactions on Evolutionary Computation*, 19(2):264–283.

Ishibuchi, H. and Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 28(3):392–403.

Jahanshahi, H., Bozanta, A., Cevik, M., Kavuk, E. M., Tosun, A., Sonuc, S. B., Kosucu, B., and Başar, A. (2022). A deep reinforcement learning

approach for the meal delivery problem. *Knowledge-Based Systems*, 243:108489.

Jaszkiewicz, A. (2002a). Genetic local search for multi-objective combinatorial optimization. *European journal of operational research*, 137(1):50–71.

Jaszkiewicz, A. (2002b). On the performance of multiple-objective genetic local search on the 0/1 knapsack problem-a comparative experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412.

Johnson, D. (1990). Local search and the traveling salesman problem. In *Proceedings of 17th International Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science,(Springer-Verlag, Berlin, 1990)*, pages 443–460.

Jozefowiez, N., Semet, F., and Talbi, E.-G. (2008). Multi-objective vehicle routing problems. *European journal of operational research*, 189(2):293–309.

Ke, L., Zhang, Q., and Battiti, R. (2013). MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and antcolony. *IEEE transactions on cybernetics*, 43(6):1845–1859.

Kenton, J. D. M.-W. C. and Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358.

Kim, J., Choe, R., and Ryu, K. R. (2013). Multi-objective optimization of dispatching strategies for situation-adaptive AGV operation in an

automated container terminal. In *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, pages 1–6.

Kim, K. H. and Bae, J. W. (2004). A look-ahead dispatching method for automated guided vehicles in automated port container terminals. *Transportation science*, 38(2):224–234.

Kim, K. H. and Hong, G.-P. (2006). A heuristic rule for relocating blocks. *Computers & Operations Research*, 33(4):940–954.

Kim, K. H. and Lee, H. (2015). Container terminal operation: current trends and future challenges. *Handbook of ocean container transport logistics*, pages 43–73.

Kim, K. H., Lee, K. M., and Hwang, H. (2003). Sequencing delivery and receiving operations for yard cranes in port container terminals. *International Journal of Production Economics*, 84(3):283–292.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kizilay, D., Van Hentenryck, P., and Eliiyi, D. T. (2020). Constraint programming models for integrated container terminal operations. *European Journal of Operational Research*, 286(3):945–962.

Konda, V. and Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in neural information processing systems*, 12.

Koo, P.-H. (2013). Dispatching transport vehicles in maritime container terminals. *International Journal of Business Tourism and Applied Sciences*, 1:90–97.

Kool, W., Van Hoof, H., and Welling, M. (2018). Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

Koza, J. R. (1994). Genetic programming II: Automatic discovery of reusable subprograms. *Cambridge, MA, USA*, 13(8):32.

Koza, J. R. and Poli, R. (2005). Genetic programming. In *Search methodologies*, pages 127–164. Springer.

Landers, M. and Doryab, A. (2023). Deep reinforcement learning verification: a survey. *ACM Computing Surveys*, 55(14s):1–31.

Le-Anh, T., van der Meer, J. R., et al. (2004). Testing and classifying vehicle dispatching rules in three real-world settings. *Journal of Operations Management*, 22(4):369–386.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, L. H., Chew, E. P., Tan, K. C., and Wang, Y. (2010). Vehicle dispatching algorithms for container transshipment hubs. *OR spectrum*, 32(3):663–685.

Lee, Y. and Lee, Y.-J. (2010). A heuristic for retrieving containers from a yard. *Computers & Operations Research*, 37(6):1139–1147.

Legato, P. and Mazza, R. M. (2001). Berth planning and resources optimisation at a container terminal via discrete event simulation. *European Journal of Operational Research*, 133(3):537–547.

Li, J., Xin, L., Cao, Z., Lim, A., Song, W., and Zhang, J. (2021). Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2306–2315.

Li, J., Yao, L., Xu, X., Cheng, B., and Ren, J. (2020a). Deep reinforcement learning for pedestrian collision avoidance and human-machine cooperative driving. *Information Sciences*, 532:110–124.

Li, K., Deb, K., Zhang, Q., and Kwong, S. (2014). An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE transactions on evolutionary computation*, 19(5):694–716.

Li, K., Zhang, T., and Wang, R. (2020b). Deep reinforcement learning for multiobjective optimization. *IEEE transactions on cybernetics*, 51(6):3103–3114.

Li, W., Wu, Y., Petering, M. E., Goh, M., and De Souza, R. (2009). Discrete time model and algorithms for container yard crane scheduling. *European Journal of Operational Research*, 198(1):165–172.

Liang, E., Wen, K., Lam, W. H., Sumalee, A., and Zhong, R. (2021). An integrated reinforcement learning and centralized programming approach for online taxi dispatching. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4742–4756.

Lim, A. (1998). The berth planning problem. *Operations research letters*, 22(2-3):105–110.

Lin, X., Yang, Z., and Zhang, Q. (2022). Pareto set learning for neural multi-objective combinatorial optimization. *arXiv preprint arXiv:2203.15386*.

Lin-Yao, Y., Si-Yuan, C., Xiao, W., Jun, Z., and Cheng-Hong, W. (2019). Digital twins and parallel systems: state of the art, comparisons and prospect. *Acta Automatica Sinica*, 45(11):2001–2031.

Liu, M., Lee, C.-Y., Zhang, Z., and Chu, C. (2016). Bi-objective optimiza-

tion for the container terminal integrated planning. *Transportation Research Part B: Methodological*, 93:720–749.

Liu, R., Nageotte, F., Zanne, P., de Mathelin, M., and Dresp-Langley, B. (2021). Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review. *Robotics*, 10(1):22.

Liu, W., Zhu, X., Wang, L., Yan, B., and Zhang, X. (2021a). Optimization approach for yard crane scheduling problem with uncertain parameters in container terminals. *Journal of Advanced Transportation*, 2021(1):5537114.

Liu, X.-Y., Yang, H., Gao, J., and Wang, C. D. (2021b). Finrl: Deep reinforcement learning framework to automate trading in quantitative finance. In *Proceedings of the second ACM international conference on AI in finance*, pages 1–9.

Liu, Y., Liu, Q., Zhao, H., Pan, Z., and Liu, C. (2020a). Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2128–2135.

Liu, Z., Li, J., and Wu, K. (2020b). Context-aware taxi dispatching at city-scale using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):1996–2009.

Lu, H., Zhang, X., and Yang, S. (2019). A learning-based iterative method for solving vehicle routing problems. In *Int. Conf. Learn. Represent. (ICLR)*.

Lu, H.-A. and Jeng, J.-Y. (2006). Modeling and solution for yard truck dispatch planning at container terminal. In *Operations Research Proceedings 2005*, pages 117–122. Springer.

Lu, Y. and Le, M. (2014). The integrated optimization of container terminal scheduling with uncertain factors. *Computers & Industrial Engineering*, 75:209–216.

Luo, J. and Wu, Y. (2015). Modelling of dual-cycle strategy for container storage and vehicle scheduling problems at automated container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 79:49–64.

Luo, J., Wu, Y., and Mendes, A. B. (2016). Modelling of integrated vehicle scheduling and container storage problems in unloading process at an automated container terminal. *Computers & Industrial Engineering*, 94:32–44.

Lust, T. and Teghem, J. (2010). The multiobjective traveling salesman problem: a survey and a new approach. In *Advances in Multi-Objective Nature Inspired Computing*, pages 119–141. Springer.

Ma, Y., Hao, X., Hao, J., Lu, J., Liu, X., Xialiang, T., Yuan, M., Li, Z., Tang, J., and Meng, Z. (2021). A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. *Advances in Neural Information Processing Systems*, 34:23609–23620.

Maashi, M., Kendall, G., and Özcan, E. (2015). Choice function based hyper-heuristics for multi-objective optimization. *Applied Soft Computing*, 28:312–326.

Maashi, M., Özcan, E., and Kendall, G. (2014). A multi-objective hyper-heuristic based on choice function. *Expert Systems with Applications*, 41(9):4475–4493.

Manikas, A., Boyd, L., Guan, J., and Hoskins, K. (2020). A review of operations management literature: a data-driven approach. *International Journal of Production Research*, 58(5):1442–1461.

Mansouri, S. A., Lee, H., and Aluko, O. (2015). Multi-objective decision support to enhance environmental sustainability in maritime shipping: A review and future directions. *Transportation Research Part E: Logistics and Transportation Review*, 78:3–18.

Martin-Iradi, B., Pacino, D., and Ropke, S. (2022). The multiport berth allocation problem with speed optimization: Exact methods and a cooperative game analysis. *Transportation Science*, 56(4):972–999.

Meisel, F. and Bierwirth, C. (2006). Integration of berth allocation and crane assignment to improve the resource utilization at a seaport container terminal. In *Operations Research Proceedings 2005: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Bremen, September 7–9, 2005*, pages 105–110. Springer.

Meisel, F. and Bierwirth, C. (2009). Heuristics for the integration of crane productivity in the berth allocation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):196–209.

Milz, S., Arbeiter, G., Witt, C., Abdallah, B., and Yogamani, S. (2018). Visual slam for automated driving: Exploring the applications of deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 247–257.

Mnih, V. (2016). Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare,

M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Mooney, C. Z., Duval, R. D., and Duvall, R. (1993). *Bootstrapping: A nonparametric approach to statistical inference*. Number 95. sage.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Nazari, M., Oroojlooy, A., Snyder, L., and Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. *Advances in neural information processing systems*, 30.

Nguyen, M. S., Lee, K. J., and Hong, J. (2018). Dispatching of multiple autonomous intelligent vehicles considering stochastic travel times by genetic algorithm. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 454–459. IEEE.

Nguyen, V. D. and Kim, K. H. (2009). A dispatching method for automated lifting vehicles in automated port container terminals. *Computers & Industrial Engineering*, 56(3):1002–1020.

Nguyen, V. D. and Kim, K. H. (2012). Heuristic algorithms for constructing transporter pools in container terminals. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):517–526.

Nishimura, E., Imai, A., and Papadimitriou, S. (2005). Yard trailer routing at a maritime container terminal. *Transportation Research Part E: Logistics and Transportation Review*, 41(1):53–76.

Niu, B., Xie, T., Chan, F. T., Tan, L., and Wang, Z. (2014). Particle swarm optimization for the truck scheduling in container terminals. In *2014 International Conference on Information Science, Electronics and Electrical Engineering*, volume 3, pages 1392–1396. IEEE.

Osman, I. H. and Kelly, J. P. (1997). Meta-heuristics theory and applications. *Journal of the Operational Research Society*, 48(6):657–657.

Paquete, L., Chiarandini, M., and Stützle, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In *Metaheuristics for multiobjective optimisation*, pages 177–199. Springer.

Parisi, S., Pirotta, M., Smacchia, N., Bascetta, L., and Restelli, M. (2014). Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2323–2330. IEEE.

Park, Y.-M. and Kim, K. H. (2003). A scheduling method for berth and quay cranes. *OR spectrum*, 25(1):1–23.

Pillay, N. and Qu, R. (2018). *Hyper-heuristics: Theory and applications*. Springer.

Prayogo, D. N., Komarudin, A. H., and Mubarak, A. (2022). Bi-objective recoverable berth allocation and quay crane assignment planning under environmental uncertainty. *International Journal of Technology*, 13(3):677–689.

Qin, Z., Tang, X., Jiao, Y., Zhang, F., Xu, Z., Zhu, H., and Ye, J. (2020). Ride-hailing order dispatching at Didi via reinforcement learning. *INFORMS Journal on Applied Analytics*, 50(5):272–286.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China technological sciences*, 63(10):1872–1897.

Rahimian, E., Akartunalı, K., and Levine, J. (2017). A hybrid integer programming and variable neighbourhood search algorithm to solve nurse rostering problems. *European Journal of Operational Research*, 258(2):411–423.

Rashidi, H. and Tsang, E. P. (2011). A complete and an incomplete algorithm for automated guided vehicle scheduling in container terminals. *Computers & Mathematics with Applications*, 61(3):630–641.

Redmon, J. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Rodrigues, F. and Agra, A. (2022). Berth allocation and quay crane assignment/scheduling problem under uncertainty: A survey. *European Journal of Operational Research*, 303(2):501–524.

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113.

Roy, D. and de Koster, R. (2018). Stochastic modeling of unloading and loading operations at a container terminal using automated lifting vehicles. *European Journal of Operational Research*, 266(3):895–910.

Royset, J. O., Dell, R. F., and Zyngiridis, I. (2009). Optimizing container movements using one and two automated stacking cranes. *MANAGEMENT*, 5(2).

Ruiz-Montiel, M., Mandow, L., and Pérez-de-la Cruz, J.-L. (2017). A temporal difference method for multi-objective reinforcement learning. *Neurocomputing*, 263:15–25.

Sadeghian, S. H., Ariffin, M. K. A. b. M., Hong, T. S., and Ismail, N. b. (2014). Integrated scheduling of quay cranes and automated lifting vehicles in automated container terminal with unlimited buffer space. In *Advances in Systems Science: Proceedings of the International Conference on Systems Science 2013 (ICSS 2013)*, pages 599–607. Springer.

Schaul, T. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schulman, J. (2015). Trust region policy optimization. *arXiv preprint arXiv:1502.05477*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

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

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis,

D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550:354–359.

Silver, E. A. (2004). An overview of heuristic solution methods. *Journal of the operational research society*, 55(9):936–956.

Skinner, B., Yuan, S., Huang, S., Liu, D., Cai, B., Dissanayake, G., Lau, H., Bott, A., and Pagac, D. (2013). Optimisation for job scheduling at automated container terminals using genetic algorithm. *Computers & Industrial Engineering*, 64(1):511–523.

Soria-Alcaraz, J. A., Ochoa, G., Swan, J., Carpio, M., Puga, H., and Burke, E. K. (2014). Effective learning hyper-heuristics for the course timetabling problem. *European Journal of Operational Research*, 238(1):77–86.

Soriguera, F., Espinet, D., and Robuste, F. (2007). Optimization of internal transport cycle in a marine container terminal managed by straddle carriers. *Transportation research record*, 2033(1):21–30.

Sun, M., Xiao, J., and Lim, E. G. (2021). Iterative shrinking for referring expression grounding using deep reinforcement learning. In *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 14055–14064.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International*

Conference on Neural Information Processing Systems - Volume 2, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. The MIT Press.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Talpaert, V., Sobh, I., Kiran, B., Mannion, P., Yogamani, S., El-Sallab, A., and Pérez, P. (2019). Exploring applications of deep reinforcement learning for real-world autonomous driving systems. In *14th International Conference on Computer Vision Theory and Applications*, pages 564–572. SCITEPRESS-Science and Technology Publications.

Tao, F., Xiao, B., Qi, Q., Cheng, J., and Ji, P. (2022). Digital twin modeling. *Journal of Manufacturing Systems*, 64:372–389.

Tao, J. and Qiu, Y. (2015). A simulation optimization method for vehicles dispatching among multiple container terminals. *Expert systems with Applications*, 42(7):3742–3750.

Tian, Y., Si, L., Zhang, X., Cheng, R., He, C., Tan, K. C., and Jin, Y. (2021). Evolutionary large-scale multi-objective optimization: A survey. *ACM Computing Surveys (CSUR)*, 54(8):1–34.

Tomazella, C. P. and Nagano, M. S. (2020). A comprehensive review of Branch-and-Bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem. *Expert Systems with Applications*, 158:113556.

Tuan, T. A., Khoa, N. T., Quan, T. M., and Jeong, W. (2021). ColorRL: Reinforced coloring for end-to-end instance segmentation. In *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 16722–16731.

Vacca, I., Bierlaire, M., and Salani, M. (2007). Optimization at container terminals: status, trends and perspectives. In *Swiss Transport Research Conference*, number CONF.

Vallada, E., Belenguer, J. M., Villa, F., and Alvarez-Valdes, R. (2023). Models and algorithms for a yard crane scheduling problem in container ports. *European Journal of Operational Research*, 309(2):910–924.

Vamplew, P., Yearwood, J., Dazeley, R., and Berry, A. (2008). On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In *AI 2008: Advances in Artificial Intelligence: 21st Australasian Joint Conference on Artificial Intelligence Auckland, New Zealand, December 1-5, 2008. Proceedings 21*, pages 372–378. Springer.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Van Moffaert, K. and Nowé, A. (2014). Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio,

Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A., Chung, J., Choi, D., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J., Jaderberg, M., and Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Vis, I. F., de Koster, R. M. B., and Savelsbergh, M. W. (2005). Minimum vehicle fleet size under time-window constraints at a container terminal. *Transportation science*, 39(2):249–260.

Voß, S. (2000). Meta-heuristics: The state of the art. In *Workshop on Local Search for Planning and Scheduling*, pages 1–23. Springer.

Wan, Y.-w., Liu, J., and Tsai, P.-C. (2009). The assignment of storage locations to containers for a container stack. *Naval Research Logistics (NRL)*, 56(8):699–713.

Wang, C., Wang, P., Qin, T., Wang, C., Kumar, S., Guan, X., Liu, J., and Chang, K. (2021). SocialSift: Target query discovery on online social media with deep reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst. (TNNLS)*, pages 1–15.

Wang, J., Liu, K., Yuan, Z., Yang, X., and Wu, X. (2024). Simulation modeling of super-large ships traffic: Insights from Ningbo-Zhoushan

Port for coastal port management. *Simulation Modelling Practice and Theory*, page 103039.

Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., Dai, B., and Miao, Q. (2022). Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR.

Wang, Z., Zhang, Q., Zhou, A., Gong, M., and Jiao, L. (2015). Adaptive replacement strategies for MOEA/D. *IEEE transactions on cybernetics*, 46(2):474–486.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8:279–292.

Weerasinghe, B. A., Perera, H. N., and Bai, X. (2024). Optimizing container terminal operations: a systematic review of operations research applications. *Maritime Economics & Logistics*, 26(2):307–341.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Wu, H., Wang, J., and Zhang, Z. (2019). MODRL/D-AM: Multiobjective deep reinforcement learning algorithm using decomposition and attention model for multiobjective optimization. In *International Sympo-*

sium on Intelligence Computation and Applications, pages 575–589. Springer.

Wu, Y., Song, W., Cao, Z., Zhang, J., and Lim, A. (2021). Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems*, 33(9):5057–5069.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Xiaolong, H. and Jiawei, F. (2016). Analysis of AGV dispatching and configuration simulation of automated container terminals. *Journal of Chongqing Jiaotong University (Natural Science)*, 35(5):151.

Xing, Z., Liu, H., Wang, T., Chew, E. P., Lee, L. H., and Tan, K. C. (2023). Integrated automated guided vehicle dispatching and equipment scheduling with speed optimization. *Transportation Research Part E: Logistics and Transportation Review*, 169:102993.

Yang, X., Hu, H., Jin, J., and Luo, N. (2022). Joint optimization of space allocation and yard crane deployment in container terminal under uncertain demand. *Computers & Industrial Engineering*, 172:108425.

Yangl, Z., Li, C., and Zhao, Q. (2018). Dynamic time estimation based AGV dispatching algorithm in automated container terminal. In *2018 37th Chinese Control Conference (CCC)*, pages 7868–7873. IEEE.

Yun, W. Y. and Choi, Y. S. (1999). A simulation model for container-terminal operation analysis using an object-oriented approach. *International Journal of Production Economics*, 59(1-3):221–230.

Zaghdoud, R., Collart-Dutilleul, S., Ghedira, K., Mesghouni, K., and Zidi, K. (2013). A multi-objective approach for assignment containers to

AIVs in a container terminal. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2460–2466. IEEE.

Zaghdoud, R., Mesghouni, K., Dutilleul, S. C., Zidi, K., and Ghedira, K. (2012). Optimization problem of assignment containers to AIVs in a container terminal. *IFAC Proceedings Volumes*, 45(24):274–279.

Zehendner, E. and Feillet, D. (2012). Column generation for the container relocation problem. In *International Material Handling Research Colloquium (IMHRC 2012)*, pages to–be.

Zehendner, E., Rodriguez-Verjan, G., Absi, N., Dauzère-Pérès, S., and Feillet, D. (2015). Optimized allocation of straddle carriers to reduce overall delays at multimodal container terminals. *Flexible Services and Manufacturing Journal*, 27:300–330.

Zeng, Q., Yang, Z., and Hu, X. (2011). A method integrating simulation and reinforcement learning for operation scheduling in container terminals. *Transport*, 26(4):383–393.

Zhang, J., Hua, X.-S., Huang, J., Shen, X., Chen, J., Zhou, Q., Fu, Z., and Zhao, Y. (2019). City brain: practice of large-scale artificial intelligence in the real world. *IET Smart Cities*, 1(1):28–37.

Zhang, Q., Hu, W., Duan, J., and Qin, J. (2021). Cooperative scheduling of AGV and ASC in automation container terminal relay operation mode. *Mathematical Problems in Engineering*, 2021:1–18.

Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731.

Zhang, Q., Liu, W., Tsang, E., and Virginas, B. (2009). Expensive mul-

tiobjective optimization by MOEA/D with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474.

Zhang, X., Li, H., and Sheu, J.-B. (2024). Integrated scheduling optimization of AGV and double yard cranes in automated container terminals. *Transportation Research Part B: Methodological*, 179:102871.

Zhang, X., Xiong, G., Ai, Y., Liu, K., and Chen, L. (2023a). Vehicle dynamic dispatching using curriculum-driven reinforcement learning. *Mechanical Systems and Signal Processing*, 204:110698.

Zhang, Y., Bai, R., Qu, R., Tu, C., and Jin, J. (2022a). A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. *European Journal of Operational Research*, 300(2):418–427.

Zhang, Y., Bao, X., Zhang, L., Chen, L., Tang, X., Zhang, Z., and Zheng, Y. (2023b). Digital twin enhanced reinforcement learning for integrated scheduling in automated container terminals. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–6. IEEE.

Zhang, Z., Wu, Z., Zhang, H., and Wang, J. (2022b). Meta-learning-based deep reinforcement learning for multiobjective optimization problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Zhen, L. and Chang, D.-F. (2012). A bi-objective model for robust berth allocation scheduling. *Computers & Industrial Engineering*, 63(1):262–273.

Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N. J., Xie, X., and Li, Z. (2018). DRN: A deep reinforcement learning framework for

news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 167–176.

Zheng, S. (2018). *Research on Model and Algorithm of Container Retrieval Operations in Terminal Yards - Based on Relocation Paths Optimization*. PhD thesis, South China University of Technology.

Zhicheng, B., Weijian, M., Xiaoming, Y., Ning, Z., and Chao, M. (2014). Modified hungarian algorithm for real-time ALV dispatching problem in huge container terminals. *Journal of Networks*, 9(1):123.

Zhong, M., Yang, Y., Sun, S., Zhou, Y., Postolache, O., and Ge, Y.-E. (2020). Priority-based speed control strategy for automated guided vehicle path planning in automated container terminals. *Transactions of the Institute of Measurement and Control*, 42(16):3079–3090.

Zhou, C., Stephen, A., Tan, K. C., Chew, E. P., and Lee, L. H. (2024). Multiagent q-learning approach for the recharging scheduling of electric automated guided vehicles in container terminals. *Transportation Science*.

Zhu, W., Qin, H., Lim, A., and Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. *IEEE Transactions on Automation Science and Engineering*, 9(4):710–722.

Zong, Z., Zheng, M., Li, Y., and Jin, D. (2022). Mapdp: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9980–9988.