Container Port Truck Dispatching Optimization using Real2Sim based Deep Reinforcement Learning

Jiahuan Jin[a], Tianxiang Cui[a,*], Ruibin Bai[a] and Rong Qu[b]

[a]School of Computer Science, University of Nottingham Ningbo China & Nottingham Ningbo China Beacons of Excellence Research and Innovation Institute, 199 Taikang E Rd, Ningbo, 315100, China
[b]School of Computer Science, University of Nottingham, UK, 7301 Wollaton Rd, Nottingham, NG8 1BB, UK

ARTICLE INFO

Keywords:
transportation
deep reinforcement learning
vehicle routing
digital port
uncertainties

ABSTRACT

In marine container terminals, truck dispatching optimization is often considered as the primary focus 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. However, many existing studies rely on strong assumptions that often overlook the uncertainties and dynamics innate to real-life applications. In this work, we propose a dynamic truck dispatching system for container ports equipped with the latest IoT technologies. The system is comprised of Real2Sim simulation and a truck dispatch agent, trained through a spatial-attention based deep reinforcement learning module, supported by an expert network. The proposed Real2Sim framework has the ability to model the non-linear complexities and non-deterministic events while our attention-aware deep reinforcement learning module is capable of making full use of both historical and real-time port data to learn a high-quality truck dispatching policy under uncertainties. Extensive experiments show our proposed method has good generalization and achieves the state-of-the-art results on the problems derived from real-life data of a large international port.

1. Introduction

Ocean shipping is the main transport mode for global trade. In fact, it is estimated that around 80% of global trade by volume and over 70% of global trade by value are carried by sea and are handled by ports worldwide. As such, the oceans provide the main transport arteries for global trade. According to projections by the International Transport Forum (ITF) (Forum, 2021), maritime freight transport will grow at a compound annual growth rate of 3.6% through 2050. This will lead to a near tripling of maritime trade volumes by 2050. This comes with opportunities and challenges. It is not surprising to witness an increasing drive for more operational efficiency in existing maritime container terminals. Over past decades, a series of container terminal related problems are investigated, such as Quay Crane (QC) or Yard Crane (YC) assignment and scheduling, truck dispatching, berth allocation, and container storage assignment.

Optimizations of these problems could help improve the utilization ratio of different equipment and resources, shorten vessels' berthing time, thus gain more business competitiveness in the container terminal. Typically, these optimizations are Combinatorial Optimization Problems (COPs) and most of them are NP-hard. Existing studies have largely focused on model-driven approaches, which first formulate the problem via mathematical models, and then use various optimization algorithms to solve the mathematical model. Generally speaking, there exist two main families of approaches for solving COPs. Exact algorithms, such as Branch-and-Bound, Branch-and-Pricing frameworks, are based on the clever exploitation of structures of the objective and constraints functions to prune the solution space while ensuring optimality. These algorithms can obtain the optimal solution eventually, but they may be prohibitive for solving large problem instances because of the exponential time complexity. Alternatively, approximation algorithms, such as heuristics and hyper-heuristics, can produce good-quality solutions within a reasonable computational time, but without proven optimality guarantees.

Even though the approximation algorithms may work well on some classical COPs, once the problem statement changes slightly, they need to be revised. To apply the approximation algorithms in newly encountered problems is an

*Corresponding author
jiahuan.jin@nottingham.edu.cn (J. Jin); tianxiang.cui@nottingham.edu.cn (T. Cui); ruibin.bai@nottingham.edu.cn (R. Bai); rong.qu@nottingham.ac.uk (R. Qu)
ORCID(s):

open challenge stemming from the No Free Lunch (NFL) theorem (Wolpert and Macready, 1997). In fact, one main issue of the model-driven approach is that it normally focuses on the deterministic variants of the problem, in which some strong assumptions are often pre-setup in the model. Since these assumptions are generally different from the practical scenarios, as a result, the algorithms developed using the model-driven approach may be hard to deploy in the real-world environment because of the high level of uncertainties. Taking container terminal truck dispatching optimization as an example, the uncertainties may come from several aspects, such as the truck traveling speed, the service time of QC or YC operations and the degree of yard congestion. Traditional model-driven approaches are always vulnerable to these uncertain factors. The solutions generated in the off-line manner often encounter various issues in the non-deterministic environment such as inferior service quality, increased costs, and infeasible solutions, all of which would lead to substantial losses. Although some modelling techniques like stochastic programming can address this issue partially, they often lead to extremely large models that tend to be intractable for most practical problems.

Recently, Reinforcement Learning (RL) algorithms have been proved effective in sequential decision making problems (Cui et al., 2024). The integration of deep learning and reinforcement learning (DRL), is drawing much attention due to its remarkable achievements in video games playing (e.g. Deep-Q-learning for Atari game (Mnih et al., 2015), AlphaStar for StarCraft II (Vinyals et al., 2019)) and board games (Alpha Go (Silver et al., 2016), Alpha Zero (Silver et al., 2017)). Reinforcement learning can be viewed as an approximation of Dynamic Programming (DP) (Bellman, 1957) which is a general divide-and-conquer technique for complex problems (such as COPs) by decomposing them into several parts (sub-problems), which have a recursive relationship. After each part has been solved, DP provides a systematic procedure for determining the combination of results of the sub-problems in order to obtain an overall solution. Therefore, DP is typically used for the problems when there are overlapping or recursive function calls and the future may depend on both the current state and past states. In RL, the problem to resolve is normally described as a Markov Decision Process (MDP). MDPs are used when sequential decision making is required. In contrast with DP, MDP model problems where the future depends only on the current state, and not on any past states. Technically, a problem that is deemed solvable by DP requires two main properties: overlapping sub-problems and optimal substructure. As a matter of fact, MDP can satisfy both of the two properties. To this end, we believe RL can provide an appropriate paradigm for finding solutions for COPs (in the equivalent DP formulations). Moreover, since the objectives in most COPs are deterministic, it can lead to relatively simple reward mechanisms and increase sample efficiency for RL, avoiding reward shaping that is normally adopted in complex real-world problems. As a matter of fact, DRL has already shown promising abilities to obtain high-quality solutions to some COPs (Cui et al., 2023; Haydari and Yilmaz, 2020; Kong et al., 2019; Mazyavkina et al., 2021).

Most of current RL researches are based on the simulation-only settings. One main drawback is that they focus on the ideal environment and thus lack of the practical applicability. On one hand, real-world optimization scenarios are too complicated to be fully reproduced in a simulation environment due to the lack of on-the-spot details and information. One the other hand, to train the RL agent in the real-world environment is usually painstaking because of extreme time cost and even infeasible for most of practical applications. To bridge such challenging reality gap, one possible solution is to use Real2Sim transferring (Rusu et al., 2017; Tobin et al., 2017) by building high-fidelity simulators directly from real-world data, so that the RL agent can be better adapted to the dynamics and uncertainties in reality.

Motivated by the demand of more effective algorithms for solving challenging COPs in the maritime container terminal, this work proposes an DRL approach for truck dispatching optimization, along with a powerful AnyLogic simulation to mimic the real-life complexities and dynamics. The proposed approach is tailored to the truck dispatching optimization problem taken from a real-world container terminal, which is also a special version of classic pickup and delivery problem (PDP). Our contributions are three-fold. Firstly, a flexible truck dispatching system is developed for real-life container ports with full-scale Internet of Things (IoT) capabilities. Our simulation system is able to sufficiently capture real-life complexities and non-deterministic factors while the tailor-made deep reinforcement learning method maximizes the utilization of both historical and real-time data generated from port IoT systems. These novelties result in significant performance gains compared to the previous works. Secondly, a tailor-made neural network structure is developed which enables the RL agent to efficiently handle state vectors of different lengths caused by diverse training instances and better use context-aware spatial information. In addition, benefit from the novel network design, the expert knowledge can be incorporated to further accelerate the training convergence. Compared with existing approaches, our proposed approach show much better generalization abilities across problem instances with different configurations. Finally, thanks to our simulation visualisation, some interesting operational insights are discovered and

analyzed. These insights are potentially beneficial for efficient container terminal management. Because of this, our DRL based simulation optimization also contribute to explainable AI to a certain degree.

The remainder of this paper is organised as follows: Related work is presented in Section 2. The problem description and formulation can be found in Section 3 and the proposed solution approach is presented in Section 4. The experimental results are reported in Section 5. Section 6 concludes the paper.

2. Literature Review

In container terminals, trucks are the most essential schedulable resources and closely connected to QCs and YCs. A sophisticated truck dispatching rule not only improves the port throughput and operation efficiency but also leverages the utilization of other equipment. The examined inner trucks dispatching in practice heavily rely on human experience (heuristic approach). As one of the most simplistic approaches, dedicated dispatching policy makes trucks organized as groups and each group serves one particular QC. Such a method is easy to implement and deploy in reality and thus used by many real-world container terminals nowadays. However, it has been proved that fixed QC-Truck dispatching policy can cause high empty mileage and low efficiency (Nguyen and Kim, 2012; Tao and Qiu, 2015). Other intuitive dispatching heuristics are implemented based on some dynamic prioritized factors such as distance, estimated time or queue length. Such dispatching rules are designed by prioritizing the task assignment and therefore are computational efficient. However, these methods could only obtain a local optima policy due to their greedy and myopic nature. Chen et al. (2016) proposed a manually handcrafted dispatching rule based on the supply-and-demand mechanism of QC. Such a heuristic considered both the spatial and the temporal factors and it was proved to outperform the existing dispatching methods that were practically deployed in Ningbo-Meshan Container Terminal. This heuristic is used as one of the benchmarks in this work.

The simple handcrafted dispatching rules usually fail to obtain competitiveness performance since they only rely on partial (mostly limited) observed information at each decision step. Another methodology that aims at acquiring high-quality solutions consider such a problem as a long-time planning rather than real-time decision making and solve it in a offline manner. In another words, all dispatching decisions are pre-scheduled without concerning the real-time feedback. One representative approach is mixed integer programming (Qin et al., 2020a; Zhang et al., 2005). Such methods are model-driven and work well for small problem instances, but can be computationally infeasible for large scale cases. In contrast, meta-heuristic approaches such as genetic algorithm (Skinner et al., 2013; Xin et al., 2021a) and particle swarm optimization (He et al., 2015; Tang et al., 2014) could overcome the massive and obtain high-quality solutions in a reasonable computation time. Some approaches use look-ahead optimization that could shorten the planning horizon thus reduce the computation time (Kim and Bae, 2004). Hybrid approaches that combine various meta-heuristics are also investigated (Chen et al., 2013; Hsu et al., 2021). However, most of these research studies overlooked the possible dynamic factors and uncertainties, thus, make them hard to be deployed in real container terminals. Therefore, these methods improve the solution quality at expense of reduced feasibility.

One of the compromise means is Genetic Programming (GP) which yields heuristic rules rather than a specific planning. Therefore, it could handle online dispatching and solution quality could also be improved through an evolutionary process. Chen et al. (2020) adopted GP to solve truck dispatching problem where the solutions are encoded as decision trees for making real-time decisions. The experiments have demonstrated that GP could reveal some hidden factors during the evolving process and thus outperform manually handcrafted heuristic approach. However, it has a low sample efficiency and may fail to construct the high-quality heuristic rules unless the genetic operators are fine-tuned.

Recently, using RL to solve COPs has been revisited since neural combinatorial optimization is proposed (Bello et al., 2017). The idea is to combine RL, sequence-to-sequence learning (Sutskever et al., 2014) and pointer network (Vinyals et al., 2015) so that the solutions of COPs can be directly obtained by a well-trained neural network. Consequently, RL has witnessed flourish successes on several classic COPs and their variants such as TSP (Xin et al., 2021b; Zhang et al., 2022a; Zheng et al., 2021), vehicle routing problem (VRP) (James et al., 2019; Ma et al., 2021b; Xin et al., 2020; Zhou et al., 2023) and knapsack problem (Cappart et al., 2021; Ozsoydan and Gölcük, 2023; Tu et al., 2023). Similar methodologies have also been promoted to multi-objective optimization for these COPs (Li et al., 2020; Lin et al., 2022; Zhang et al., 2022b). The use of graph neural network in such a methodology is also investigated (Wu et al., 2021). Rather than constructing a solution directly through the sequence-to-sequence model, iterative solution perturbation can also be an alternative learning method (Chen and Tian, 2019; Lu et al., 2019).

Apart from solving canonical COPs, RL has also shown competitive ability for decision marking and optimization problems in real-world scenarios. The pickup and delivery problem (PDP), as a representative variant of VRP is broadly

investigated to be solved by DRL since it shows strong connections to numerous real-world optimizations including our examined problem (a full-truckload version of PDP). The relevant methodologies are innovated by using customized design of attention mechanism (Li et al., 2021), hierarchical RL (Ma et al., 2021a), and multi-agent RL (Zong et al., 2022). By setting different constraints such as multiple time windows for customers, some real-world applications such as food delivery problem (Jahanshahi et al., 2022; Zou et al., 2022) and taxi dispatching problem (Liang et al., 2021; Liu et al., 2020; Qin et al., 2020b) are also evolved from the classic PDP. These are two most important Internet business forms nowadays and have attracted considerable research efforts in the field of operational research. Such problems are benefit from the advantages of RL based approaches due to their online and dynamic nature.

The examined truck dispatching problem is naturally suitable for RL-based methods (Bai et al., 2023) since it is an online decision-making problem that can be derived from traditional PDP. However, the aforementioned works on classic COPs are based on over-simplified dynamic details of problem environments. As a matter of fact, the optimizations in real container terminals can be far more complex. Consider the operations in a traditional container terminal as an example, several interconnected sub-problems would be jointly considered, including berth allocation problem, quay crane scheduling, yard allocations, yard crane scheduling and finally the truck dispatching. Any single operation can affect others and yet the joint optimization of all sub-problems is more challenging (Kizilay et al., 2020). Additionally, there are various uncertainties factors in most of these sub-problems. For example, port equipment-related parameters such as service time are not constant and random situation such as general disruptions may occur (Rodrigues and Agra, 2022). These factors may place a burden on obtaining high-quality solutions for RL-based approaches.

Zeng et al. (2011) made early efforts to use Q-learning to solve the integrated YC and yard trailers scheduling problem. However, such a conventional RL method cannot produce competitive results due to its over-simplified neural network structure. Recently, Zhang et al. (2021) proposed a DRL-based hyper-heuristic framework for the container terminal truck dispatching problem. The competitiveness performance of their framework was experimentally demonstrated in the real-world port environment with uncertainties. This approach is used as another benchmark of our work. However, the performance of the hyper-heuristic framework may heavily rely on the quality of low-level heuristic design. In addition, the reduced action space using low-level heuristics may further limit its performance.

Numerous container ports starts to deploy automatic guided vehicles (AGV) as the main carriers since they are driverless and suitable for highly automatic and intelligent port management (Sun et al., 2022). The Multi-Agent Deep Deterministic Policy Gradient (MADDPG) was used in Hu et al. (2023) to solve the AGV path planning problem with the aim of avoiding the conflict in an automated container terminal. Zheng et al. (2022) adopted deep-Q-network (DQN) method to address the multi-AGV dynamic scheduling problem. In these applications, truck dispatching also happens at cases of inter-terminal container transportation where trucks need to transfer containers among several port terminals. Such problems have been also investigated with RL (Adi et al., 2020). In addition, RL could also help to leverage the truck dispatching optimization in other industrial field such as surface mining (Afrapoli et al., 2019; de Carvalho and Dimitrakopoulos, 2021).

3. Problem Description and Formulation

There are several main components in a container terminal, namely, berth, quay crane (QC), trucks, storage yard, and yard crane (YC), as shown in Fig 1. Berth and QC are at the seaside. The storage yard and YC locate at the landside. The berth is the area for vessels to load and unload containers. A QC is used to carry the containers between ships and trucks. Each QC can only finish one unit task so queues are common. Yards are used to store the containers temporarily. They are divided into yard blocks and each yard block is further divided into multiple bays of identical length. Yard cranes(YCs) are used to transfer the containers between trucks and yards. Like QCs, YCs handle a unit task each time. Normally each yard block is equipped with a YC. Finally trucks are used to transport containers between vessels and yard blocks and must strictly follow the traffic directions and other safety related rules. A cut-out example of the container terminal layout is illustrated in Fig. 1.

Truck dispatching happens upon a vessel's arrival. A truck fleet of unit capacity takes charge of loading and unloading containers between QCs and YCs. All the loading and unloading tasks of unit size (i.e. one 40-inch container or two 20-inch containers) are arranged and allocated to different QCs in advance by a separate procedure. Each QC has a list of tasks that need to be executed (i.e., a task is assigned to a specific truck) in a pre-defined order. The allocated tasks for the trucks are represented as a task instruction as shown in Table 1. It contains an unique task id, the source location, the destination location, the task type, the source bay and the destination bay. The source or destination location is either QC or YC. Task type, which are represented by LOAD and DISCHARGE, indicates loading to the

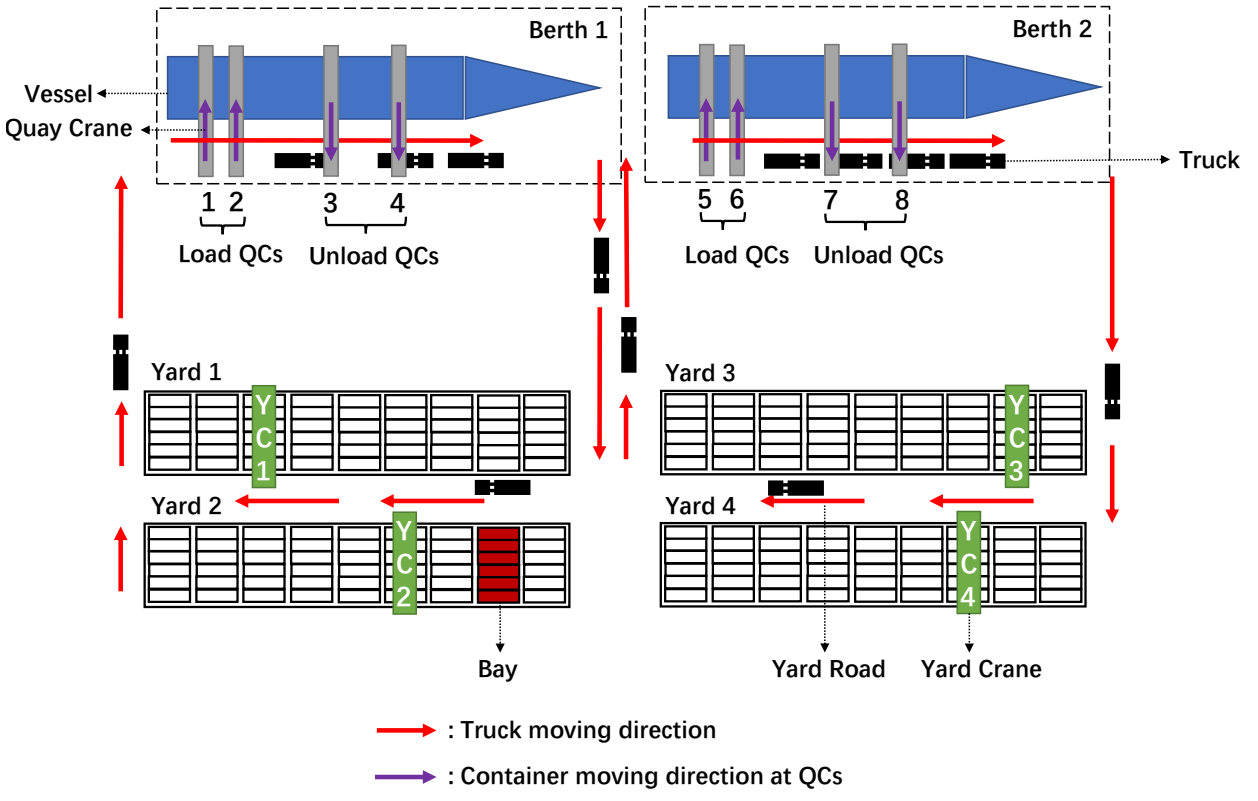

Figure 1: A cut-out example of the container terminal layout. The red arrows represent the truck moving directions allowed at different areas. In this case, each vessel are equipped 2 load QCs and 2 unload QCs.

Table 1
An example of task instructions. For task 1, the source container at bay 15 that is dedicated to QC6 are about to transfer to the yard 32 at bay 12.

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

ship or unloading from the ship, respectively. There are two types of QC, namely loading and unloading QC, where loading QCs only execute LOAD task and unloading QCs only execute DISCHARGE task. The source and destination bay are the stopping positions for the truck at yard or vessel. A task instruction refers to moving a container from one location to another location. Generally speaking, the procedure of executing a task assigned to a truck can be described as below:

1. Move to the source bay of the source location.
2. Pick up the target container after waiting for the queuing and the crane service time.
3. Move to the destination bay of the destination location.
4. Unload the target container after waiting for the queuing and the crane service time.
5. Request a new task upon the completion of the current task.

A truck could only execute one task at one time. When a task of a specific QC is dispatched, this task needs to be removed from the QC's task list. When a truck finishes the task, the scheduler needs to dynamically select a new task for it based on the states of all active QCs, YCs and trucks. The decision must be made within a relative short period

Table 2
Notations used in the problem formulation.

Notation	Description
	Sets
V	The set of dispatchable trucks. Each truck is retrieved by $v \in V$.
Y	The total set of yard blocks, each of which is deployed with a yard crane.
Q	The set of quay cranes, each of which is retrieved via $q \in Q$.
L^q	The list of tasks attached to quay crane $q \in Q$. Tasks must be completed in the given order.
W	The set of all tasks attached with all the quay cranes Q, hence $W = \cup_{q \in Q} L^q$.
B^y	The set of all tasks using a same yard crane $y \in Y$.
	Indices
i, j	Indices used for sequential access to task lists and the objective vector.
t	Index used for time steps.
	Parameters
w_i^q	The i-th task in task list L^q, $q \in Q$.
$\alpha(q)$	A binary variable to indicate the type of QC q.
O_i^q	Time duration for quay crane q to operate task w_i^q.
T_{init}	Initial time of the simulation.
	Decision and Auxiliary Variables
$x(w_i^q, v)$	A binary variable to indicate whether task w_i^q is assigned to truck v or not.
T_i^q	The starting time of quay crane q to operate its task w_i^q.
$z(k, l)$	A binary variable to indicate whether task l is serviced immediately after task k by any truck. A truck travels from/to the depot is marked as a dummy task 0.
S_v^k	The time when truck v commits its service to task k.
E_v^k	The time when truck v finishes task k at its destination node.
$D(k, v)$	The time duration for truck v moving to the target QC of the task k.
$H(k, v)$	The time duration for truck v waiting at the queue of the target QC of the task v.
$T_{arr}^{k,y}$	The arrival time of task k at its yard y.
$T_{comp}^{k,y}$	The operation completion time of task k at its yard y.

of time (usually a few seconds). Normally, only the first task in each QC's task list can be selected. For example, if there are m QCs with the non-empty task list, this means the scheduler has m choices for the current truck dispatching decision. Moreover, the QC service constraint requires QC serving the trucks in their task dispatching order, which means sometimes trucks with latter dispatched tasks need to wait in the QC queue if trucks with earlier dispatched tasks are not arrived at this QC. We define the amount of time the QC takes to finish all assigned tasks as QC makespan.

In our application, the objective is to minimize the the summation of all QC's idle time in its QC makespan. This objective can help improve the QC utilization ratio and affect the entire port throughput as a vessel would have less duration time at the berth if all QCs are kept busy. Consequently, the objective of examined problem is formulated from the perspective of QC operation flow (as depicted in Fig. 2). Assume that each QC has two states, namely, operating and idle. The QC operation flow indicates the repeated process that a QC switching between its two states. The objective function, constraints and the uncertainty factors are formulated based on such a perspective.

Below is the mathematical formulation of the objective function and relative constraints. The involved notations can be found in Table 2 which also includes several variables used in section 4.2.1. Denote Q and Y be the set of all QCs and YCs nodes, respectively. Let V be the set of unit-sized homogeneous port inner trucks. The problem can be formulated on a directed graph $G(N, A)$ where N is the union of depot node 0, and the set of points of work (PoW), consisting of all quay cranes Q and all yard cranes Y, i.e. $N = \{0\} \cup Q \cup Y$. A represents the capacitated road network of the container terminal connecting different nodes. Therefore, the truck travel speed shall depend on the volume of the traffic at different segments in A. In addition, truck movement over G follows the shortest path rule and the routing process is automatically conducted within the simulator. Upon the arrival of a vessel, a number of quay cranes Q are assigned to this vessel and each quay q is given a task list L^q to complete in the given sequence. A task is one of two

possible types: either loading (i.e. yard to ship) and unloading (ship to yard). Trucks are dispatched to complete each of these transportation tasks, along with load/unloading operations by cranes at both their source and destination nodes.

The dispatch is triggered by an idle truck. Starting from the depot node, each truck needs to complete a series of transportation tasks and returns to the depot if there is no new task assigned to it. Let $W = \bigcup_{q \in Q} L^q$ be the set of all tasks. Our problem is to assign each of these tasks to a dispatchable truck so that the objectives are optimized. Two set of variables are used to encode the solution to our problem. The first set of variable is the assignment variable $x(w_i^q, v)$ which takes value 1 if task w_i^q is assigned to truck v and 0 otherwise. The auxiliary variable T_i^q is used to denote the time that the i-th task of quay crane q is served by quay crane q. Let O_i^q be the corresponding operating time duration, $O_i^q > 0$. The second set of variables $z(k, l)$ defines the sequence in which a pair of tasks are serviced by a particular truck.

It can be seen from Fig. 2 that idle waiting at quay cranes can happen when the operation of a truck may be interrupted because of insufficient feeder trucks which should be minimized through intelligent truck dispatching. This leads to the following objective function Eq.(1) that also takes into account idle time at the start of the operations. Specifically, the objective function considers truck's traveling time, truck's queuing time, and the service time of cranes (QC and YC). The variable T_i^q in the objective function can be further expanded by Eq.(2), where $D(w_i^q, v)$ is the time duration (excluding the time that is spent at the yard) that the truck v takes to reach the target QC for task w_i^q, $D(w_i^q, v) > 0$. α is an indicator of QC type ($\alpha(q) = 1$ indicating q is for loading task and $\alpha(q) = 0$ indicating q is for unloading task). $T_{comp}^{w_i^q, y}$ and $T_{arr}^{w_i^q, y}$ are completion and arrival time at the target yard y for task w_i^q and $H(w_i^q, v)$ is the corresponding queuing time at the QC, $H(w_i^q, v) \geq 0$.

$$\min \sum_{q \in Q} \sum_{i=2}^{|L^q|} (T_i^q - T_{i-1}^q - O_{i-1}^q) + \sum_{q \in Q} (T_1^q - T_{init}) \tag{1}$$

$$T_i^q = \sum_{v \in V} x(w_i^q, v)[S_v^{w_i^q} + D(w_i^q, v) + \alpha(q)(T_{comp}^{w_i^q, y} - T_{arr}^{w_i^q, y}) + H(w_i^q, v)] \tag{2}$$

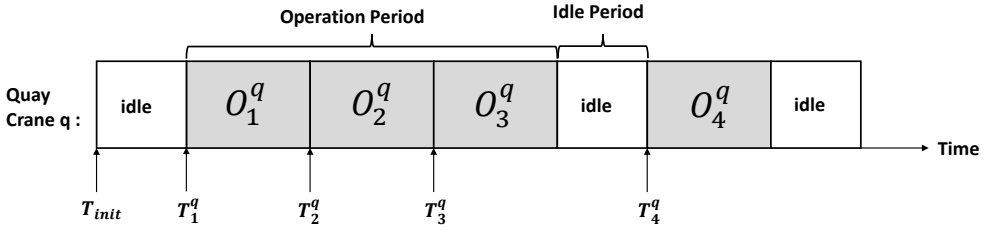

Figure 2: An example of QC operation flow. $T_1^q, T_2^q, T_3^q, T_4^q$ are the starting times that quay crane q starts to operate the containers of different tasks. $O_1^q, O_2^q, O_3^q, O_4^q$ are their corresponding operating duration which are indicated by the grey cells.

A number of constraints must be satisfied. The first set of constraints is to ensure the full completion of all tasks (constraint (4) and (5)) and each task is transported by exactly one truck (constraint (3)). Additionally, any task assignments that violates the order of tasks in each queue L^q is prohibited (constraint (6)).

$$\sum_{v \in V} x(w_i^q, v) = 1 \quad \forall w_i^q \in W \tag{3}$$

$$\sum_{k \in W \cup \{0\}} z(k, l) = 1 \quad \forall l \in W \tag{4}$$

$$\sum_{l \in W \cup \{0\}} z(k, l) = 1 \quad \forall k \in W \tag{5}$$

$$\sum_{i\in L^q \& i>j} \sum_{j\in L^q} z(w_i^q, w_j^q) \quad = \quad 0 \quad \forall q \in Q \tag{6}$$

The second set of constraints is related to the operation time at quay cranes and non-stop requirements for any continuous tasks by a truck. Constraint (7) ensures that the tasks need to be operated (load or unload) by QC in order. Denote S_v^k and E_v^k as the time steps that a truck v commits to receive and finishes the task k respectively. Thus, the time duration for truck v to execute task k is $E_v^k - S_v^k$, while the movement time of truck v, queuing, loading and unloading time of related cranes are considered. Constraint (8) guarantees truck executes the tasks continuously.

$$T_i^q \quad \geq \quad T_{i-1}^q + O_{i-1}^q \quad \forall q \in Q, \quad i \in [2, |L^q|] \tag{7}$$

$$S_v^l x(l, v) \quad = \quad E_v^k z(k, l) \quad \forall v \in V, k \in W, l \in W \tag{8}$$

Finally, denote B^y be the set of all tasks using a same yard crane $y \in Y$. Then, the operation completion time of each task $k \in B^y$ at yard crane y, denoted as $T_{comp}^{k,y}$, should respect company's first-come-first-service (FCFS) rule. This constraint is expressed by Eq. (9) where $T_{arr}^{k,y}$ and $T_{arr}^{l,y}$ indicate the arrival time of task k and l at their corresponding yard y. Such a FCFS constraint highlights the complexity of the examined problem due to its nonlinearity and indeterminacy nature which relies on the implementation of simulation.

$$T_{comp}^{k,y} = FCFS(T_{arr}^{k,y}, T_{arr}^{l,y}) \quad \forall y \in Y, \forall k \in B^y, \forall l \neq k \in B^y \tag{9}$$

Note that the auxiliary variables listed in Table 2 serve two main purposes: first, they are used to help model nonlinearities and uncertainties in the objective function and related constraints. Second, these auxiliary variables can help describe the important state variables that are closely monitored intermediate data in real-life decision making (e.g., $T_{arr}^{k,y}$ and $T_{comp}^{k,y}$). In our examined problem, they are fully observable from the simulator. Additionally, they can represent the information of real-world container terminal that cannot be explicitly computed (e.g., T_i^q, S_v^k, and E_v^k). The events related to the examined problem (e.g., $z(k, l)$) are proceeded based on these variables inside the simulation.

4. The Proposed DRL Approach

4.1. Reinforcement Learning

Reinforcement Learning (RL) is one of machine learning branches that aim to train agents (decision markers) from interactions with the environment (problem) in a trial-and-error manner. In general, the agent repeatedly takes the actions suggested by a set of policies and then update the policies based on the numerical rewards from the environment. Through this iterative process, the agent can gradually improve its policies to obtain performance across different realizations of random variables.

RL solves the sequential decision-making problems that can be formalized as the Markov Decision Process (MDP) (Sutton and Barto, 1998). Generally speaking, a MDP can be represented by a tuple $M = (\mathcal{S}, \mathcal{A}, R_a, P_a, \gamma)$, where $\mathcal{S}$ is the set of state representations. $\mathcal{A}$ is the set of actions that can be selected by the agent. R_a is the immediate reward obtained after transferring the current state to the next state due to action a. P_a is the state transition probability, which can be represented as $P_a(s', s) = Pr(s_{t+1} = s', |s_t = s, a_t = a)$, indicating the probability of action a in state s at time step t leading to state s' at time step $t + 1$. $\gamma \in [0, 1]$ is the decay factor that can be fine-tuned during algorithm development.

The goal of the agent acting in MDP is to find a policy π_θ that can map states into actions. Solving MDP means finding an optimal policy that maximize the discounted accumulated rewards.

4.2. Truck Dispatching Optimization as an MDP

A finite MDP with discrete time step is applied to formulate the truck dispatching optimization problem. The time interval between two adjacent steps is dynamic and is dependent on the time that a certain truck just finishes its current task (i.e. becomes dispatchable). The objective of the truck dispatching optimization is to minimize the total idle time that defined in Section 3 by dispatching candidate tasks to trucks at each time step. The details of our formulation are as follows:

4.2.1. Environment

A Real2Sim transferring simulation based on AnyLogic software is developed to model the environment of the container terminal. Usually, it is infeasible and unsafe to train a dispatching policy in the real-world environment directly. With the help of simulation, events in reality can be accelerated. Based on the plentiful built-in tools of Anylogic, abundant aspects of real-world container terminal management such as physical component, business logic and uncertainties can be considered to mimic the real operational complexities and dynamics (see Fig. 3). To this end, a high-fidelity and effective simulation environment that relates to the examined truck dispatching optimization problem is created. Also, according to the historical experiences in the real container terminal, some representative scenarios are summarized and used to design the problem instances and configure the simulation in training process. Some sophisticated dispatching heuristics are also considered as the forms of expert knowledge and used to initialize (part of) the neural network, which is further described in Sec 4.3. By such means, the well-trained agent through our proposed Real2Sim approach has the potential to be deployed in real-world container terminal with a little tuning effort.

The parameter settings follow the advice from our collaborators in Ningo-Zhoushan port in order to make the simulation as realistic as possible. The environment simulates the entire truck dispatching process and the uncertain factors are also taken into account. In the real-world situation, the truck speed cannot be constant even if it travels on the empty road. The truck's travelling speed is changing all the time as it is affected by turning, passing by other vehicles, and the load weight. In our simulation environment, the truck's travelling speed at the road (except the yard road) is defined by a speed range which follows a given distribution. Given the fact that in many container terminals, the container loading and unloading operations are manually executed and the proficiency may vary from person to person, crane bridge moving distance may also vary considerably in each service, the service time of each container loading and unloading operation is non-deterministic. Therefore, in our simulation environment, the service time of each operation is also formulated as a given distribution. The degree of yard congestion is another factor that can cause the uncertainty. Normally, in the container yard, a three-lane road is shared by two neighbor yards. The middle lane is used for truck passing through, the other two lanes are used for container loading and unloading operations. Under certain circumstances, this three-lane road can be quite congested because there are too many collision avoidance related truck manoeuvres. Therefore in our simulation environment, we set truck's speed inversely proportional to the total number of trucks on this road at the same time when trucks enter or leave the yard roads. The specific simulation parameters can be found in Table 3 and Fig. 4 presents an environment inner screenshot.

Some uncertain factors are modeled based on the real-world container terminal operations. These factors are embedded in the simulation environment and are briefly introduced here. Denote C_t^v as the traveling speed of the truck v at time step t. Denote $g_t^{y,y'}$ as the number of trucks queuing at the adjacent yards y and y' that share a same yard road segment at time step t. Eqs. (10) and (11) are trucks state variables at time step t that affect truck speed.

$$\beta(v,t) = \begin{cases} 1, & v \text{ is on-load state at time step } t \\ 0, & v \text{ is empty-load state at time step } t \end{cases} \tag{10}$$

$$\phi(v,t) = \begin{cases} 1, & v \text{ is on its target yard road of the} \\ & \text{current task at time step } t \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Eq. (12) represents the truck speed in different areas in container terminal, in kilometer per hour.

$$C_t^v = \begin{cases} c_1 \sim f_1, & \beta(v,t) = 1, \phi(v,t) = 0 \\ c_2 \sim f_2, & \beta(v,t) = 0, \phi(v,t) = 0 \\ 10, & g_t^{y,y'} > 10, \phi(v,t) = 1 \\ 30 - 2g_t^{y,y'}, & 1 \le g_t^{y,y'} \le 10, \phi(v,t) = 1 \end{cases} \tag{12}$$

Table 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)

where c_1 and c_2 are sampled from the probability density functions for truck speed with empty and on load state, f_1 and f_2, respectively. f_1 and f_2 are modeled from real-world container terminal operation environment.

The uncertainties of truck traveling time at different areas make S_v^k and E_v^k (defined in section 3) non-deterministic in truck operation flow. Furthermore, O_i^q is not a constant as it can be affected by the movement of QC bridge, trolley, and hook as well as the operator's proficiency. These uncertain factors are also modeled as distributions estimated from the real-world port operation scenario.

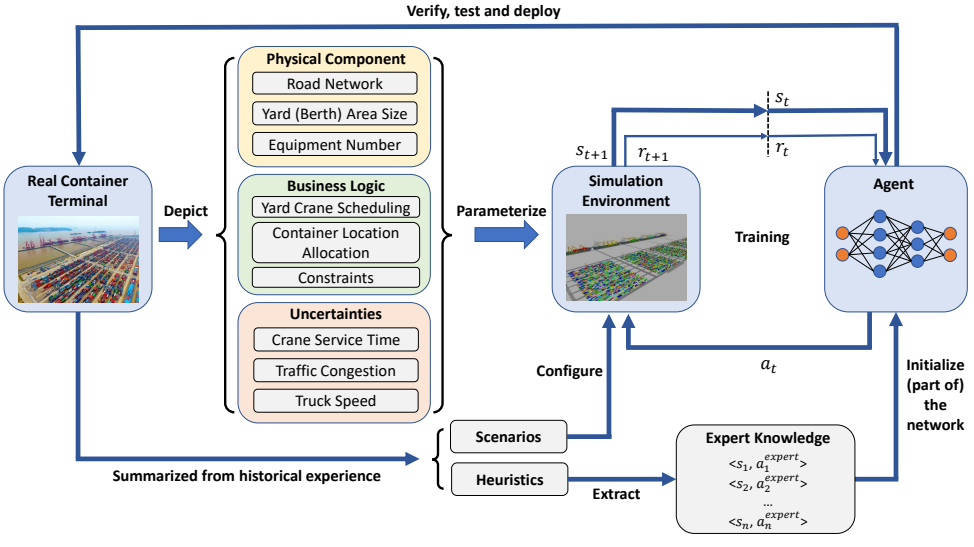

Figure 3: The Real2Sim framework of the proposed reinforcement learning environment.

4.2.2. State

Once a given truck finishes its current task, a request is raised and the centre dispatcher (RL agent) assigns a new task to the truck (dispatch). The observation at that time is based on the information of the target truck and the state of each QC. The following information is considered as the state observations of the agent at time step t for the truck v that makes the request. The observations are designed by thoroughly investigating the examined problem through the implemented Real2Sim based simulator. The detailed feature analysis can be found in section 5.3.2.

- The amount of the remaining tasks of each QC: $\mathbf{RE}_t = [RE_t^1, RE_t^2, \dots, RE_t^{|Q_t'|}]^\intercal$.
- The travelling distances for truck v to arrive at each candidate QC: $\mathbf{DQ}_t = [DQ_t^1, DQ_t^2, \dots, DQ_t^{|Q_t'|}]^\intercal$.
- The travel distance between truck v and the first operation location of the first task of each candidate QC: $\mathbf{DF}_t = [DF_t^1, DF_t^2, \dots, DF_t^{|Q_t'|}]^\intercal$. For import container tasks, the first operation location is a QC. For export container tasks, the first location is a YC at yard blocks.

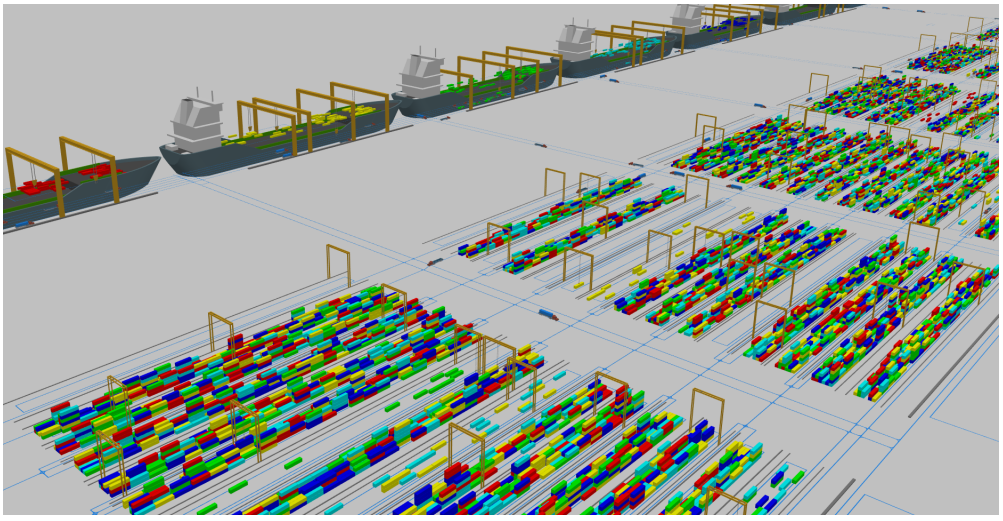

Figure 4: A screenshot of the container terminal simulation environment.

- The transport distance of the second task of the candidate QCs: $\mathbf{DS}_t = [DS_t^1, DS_t^2, \dots, DS_t^{|Q_t'|}]^\intercal$.
- The total number of trucks currently working for each QC: $\mathbf{TW}_t = [TW_t^1, TW_t^2, \dots, TW_t^{|Q_t'|}]^\intercal$.
- The total number of trucks heading to each QC: $\mathbf{TH}_t = [TH_t^1, TH_t^2, \dots, TH_t^{|Q_t'|}]^\intercal$.
- The queue length of each QC: $\mathbf{QL}_t = [QL_t^1, QL_t^2, \dots, QL_t^{|Q_t'|}]^\intercal$.
- The queue length of each QC's first task in the corresponding yard block: $\mathbf{QY}_t = [QY_t^1, QY_t^2, \dots, QY_t^{|Q_t'|}]^\intercal$.
- The type of each QC: $\mathbf{TY} = [TY^1, TY^2, \dots, TY^{|Q_t'|}]^\intercal$, $TY^q = [1, 0]$ if the type is load, $TY^q = [0, 1]$ if the type is discharge, $q \in [1, |Q_t'|]$.

Q_t' refers to the set of non-empty QCs at the time step t, which is dynamic in an episode. The shape of $\mathbf{TY}$ is $|Q_t'| \times 2$ since $\mathbf{TY}$ is encoded as a two-digit one-hot vector indicating whether a QC is loading or unloading. The shapes of other 8 features are all $|Q_t'| \times 1$. Thus, the state vector s_t at time step t for dispatching truck v is encoded as a $|Q_t'| \times 10$ matrix where each row of the matrix is denoted as $[RE_t^q, DQ_t^q, DF_t^q, DS_t^q, TW_t^q, TH_t^q, QL_t^q, QY_t^q, TY^q]$, $q \in [1, |Q_t'|]$.

4.2.3. Actions

Given the state s_t, the action space at time step t for dispatching truck v is $\boldsymbol{a_t} = \{q \mid RE_t^q > 0\,, q \in Q_t'\}$, where q indicates the selected non-empty QC. Once q is selected, its first remaining task is assigned to truck v. Note that the number of actions at each dispatching time step is not always the same because tasks from some QCs may have been completed by then.

4.2.4. Reward

In this work, the reward design follows the same principle of the work in Zhang et al. (2021) which is primarily concerned with the QC idle time to each task. The reward for assigning a specific working instruction (task) w_i^q to the target truck is defined as the idle time that the corresponding QC needs to wait before it starts to execute w_i^q. If we denote the reward of dispatching w_i^q at time step t as r_t, then $r_t = -(T_i^q - T_{i-1}^q - O_{i-1}^q)$, $i \geq 2$ and $r_t = -(T_1^q - T_{init})$, $i = 1$, which is the negative of the objective function defined in Eq.(1). Note that the reward r_t cannot be obtained immediately because the time step t for dispatching w_i^q is earlier than the time that the corresponding QC starts to operate on it, (i.e. $t < T_i^q$), as a result, each r_t is computed episodically. That is, at each episode, when all the tasks in a given problem instance are dispatched and finished (i.e. an episode is finished), the rewards are calculated retrospectively.

4.2.5. State Transition

The state transition from s_t to s_{t+1} is governed by the function: $s_{t+1} = F(s_t, a_t, u_t)$. The transition may not only depend on the action a_t but also can be affected by the uncertainties u_t existed in the environment. In this work, the transitions for $\mathbf{DQ}_t, \mathbf{DF}_t, \mathbf{DS}_t, \mathbf{TW}_t, \mathbf{TH}_t, \mathbf{QL}_t, \mathbf{QY}_t$ are subject to u_t, which is caused by the uncertainties introduced in section 4.2.1. Specifically, the operation time of each YC and QC may vary, and the speed for each truck is not constant, these will cause the non-deterministic probability transitions for the aforementioned states. The distributions of each component of u_t are learnt from the real-world operation data provided by our port collaborators and embedded in the simulation environment. On the other hand, the transitions for $\mathbf{RE}_t$ are directly affected by agent's actions. In this work, the state transition is automatically executed by the simulation environment.

4.3. Network Topology

The policy network used in this work is depicted in Fig. 5. Given the state vector described in section 4.2.2, the policy network takes the feature vectors of each QC as the inputs and outputs a list of action probabilities that represents the selection probability of each input QC.

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 feature vector for each QC and the cross-scenario net is used as the actor for RL agent so that the probability distribution of actions are computed. For the expert network, the feature vectors of each QC are firstly fed into a three-layer long-short term memory (LSTM). Next, the hidden states of each LSTM step are fed into an feed forward layer to generate a 512-dimensional feature vector for each QC, namely H^{Expert}. Here 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 which takes the same input states and outputs a tensor with a 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 consists of the information for both H^{Expert} and H^{Target} are generated. Then, an attention layer which has the similar structure with 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. Only the parameters of cross-scenarios network are updated in training process and the expert network is fixed.

Unlike the conventional LSTM which takes the sequential structure data as input, our network treats the state vector of each QC as a dynamic set with spatially connected elements (target truck, QCs, tasks, etc). The spatial information of both truck and QCs is embedded in the state design (see section 4.2.2). The input data follows the fixed order based on QC's position in terminal (same direction as the roads besides QCs). The use of LSTM is to make network model capable of handling 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 entire QC sequence are fully propagated at each step and thus capable of capturing some hidden features such as one-way road at sea-side. The gate component controls the information flow of the expert knowledge based on the raw observation. With the help of the attention layer, each extracted QC feature vector is aware of the entire sequence information and the multi-head attention mechanism allows the agent 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 length 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 became empty) by removing that QC from the input list.

4.4. 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. Recently, researchers try to combine IL with DRL to order to reduce 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 RL agent and obtain some basic prior knowledge of the examined problem. The parameters of expert network are learned through a simple on-policy iterative supervised learning algorithm (as shown in Algorithm 1) 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 the environment uncertainties and thus should be suitable as prior knowledge for RL agent. The loss is defined by difference between expert network output and

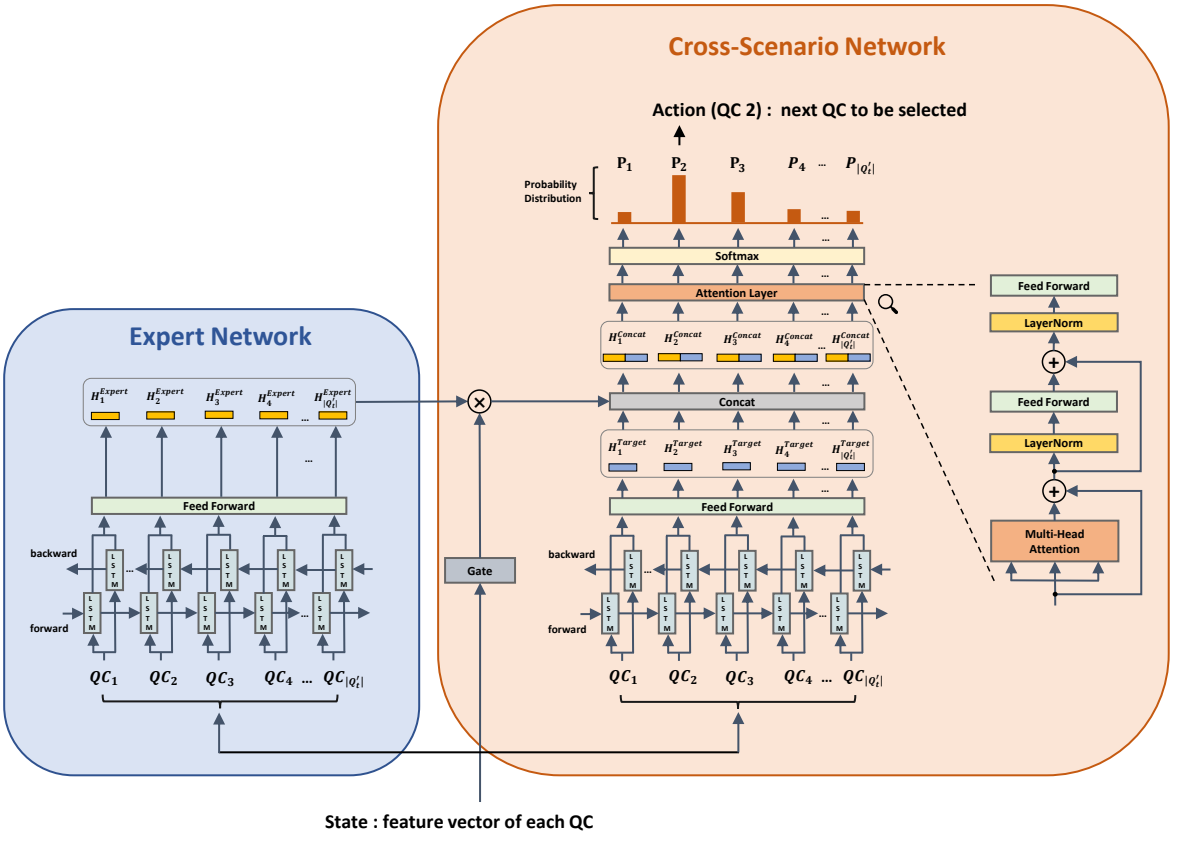

Figure 5: The expert network and cross-scenario network structure.

the label (decision of the manual heuristic). For example, for a given state observation at some time step t, manual heuristic would provide an unique answer for which task to be selected for this state (input) and the expert network using one fully-connected layer with softmax would output a probability distribution of the actions. Then the cross entropy between the network output (probability distribution) and label (one-hot vector form) can be computed. The loss can be used to update the expert network parameters by back-propagation accordingly. The imitation mechanism could make the agent converge to the heuristic level in a relatively short time period and consequently make RL training process more stable. The detailed experimental results are presented in section 5.3.3.

Algorithm 1: 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

4.5. Policy Based RL

Model-free RL aims to obtain an effective behavior policy through the trial and error interactions with the environment (Nachum et al., 2017). Generally speaking, model-free RL can be divided into two categories: value-based methods and policy-based methods. Compared with value-based methods, policy-based methods directly optimize the policy function that explicitly maps states to actions while remaining stable in the training process. For complex tasks, policy-based methods tend to perform better as they can handle the exploration/exploitation trade off by learning a stochastic policy. In this work, we adopt the standard policy-based method, Proximal Policy Optimization (PPO) (Schulman et al., 2017) (see Algorithm 2), to tackle the examined problem.

A widely used variation of policy based methods is to subtract a baseline value from the return to reduce the variance of gradient estimation while keeping the bias unchanged. A good baseline estimation can help stabilise the training process and accelerate the convergence. In this work, we adopt the shared baseline which is calculated as $b = \frac{\bar{R}}{N}$ where N is the number of episodes and $\bar{R}$ is the total return. As mentioned in section 4.2.4, the immediate reward r_t cannot be obtained instantly at its time step t. Rather, it needs to be computed at the end of the episode. Recall the reward definition, $r_t = -(T_i^q - T_{i-1}^q - O_{i-1}^q)$, dispatching a truck to an idle QC would result in a small T_{i-1}^q, which may depreciate r_t. Consequently, the agent may tend to choose the QC with long queue length rather than the QC is about to be idle, misleading the optimization direction and causing RL algorithms convergence failures. To resolve this issue, all state-actions pairs in one single episode are assigned with a same advantage function, $R^n - b$, where the R^n is the total reward of an episode. As a result, the advantage function can clearly reflect the difference between average episode reward and total episode reward and the gradient updating direction is the same as the optimization objective of the examined problem.

Algorithm 2: 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

Table 4
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

Table 5
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 data-driven genetic programming approach.
Manual Heuristic	A sophisticated heuristic used in real port.
DRL-HH	Policies trained by DRL-based hyper-heuristic approach.

5. Experiment Design and Result Analysis

5.1. Problem Instances

A set of problems with different instance configurations are generated for RL training. The configurations are designed according to different case scenarios in the real-world port operation environment. Each case scenario is based on various truck fleet sizes, number of tasks, number of QCs, container storage mode and yard distribution. In the simulation environment, each ship is equipped with two loading QCs and two unloading QCs. We choose 60, 80 and 100 as the number of trucks in our problem settings since too many or too little trucks may lead to obvious optimal dispatch policies and hence leaves no room for optimization. There are two stacking modes: `mixed` stacking indicates the loading/unloading containers can share the same yard while `separated` stacking indicates the yards for the loading/unloading containers are different. There are two types of yard distribution as well: `centralized` indicates the loading/unloading container are stored in one or two yards while `distributed` indicates the loading/unloading container are stored in more than two yards. The configurations of the training instances can be found in Table 4. Note that one configuration represents a set of (infinite) instances with similar initialization conditions because of the uncertainties in the environment. A static problem instance is created by fixing the configuration and random seed.

5.2. Benchmark

A manually designed dispatching heuristic (Chen et al., 2016) is deployed and used as the benchmark for the examined problem. Dispatching rule that generated by genetic programming (Chen et al., 2020) is also considered as a benchmark since it is a common approach for online decision-making optimization. We also compare the proposed method with our recent work, deep reinforcement learning-based hyper-heuristic (DRL-HH) (Zhang et al., 2021) since it can properly handle online decision-making problems. Moreover, it also uses expert knowledge (via low-level heuristics) to stabilize the training process of RL. For the sake of fairness, both genetic programming based approach and DRL-HH are trained in the same simulation environment with the same problem instances. Apart from these three methods, some heuristic dispatching rules (Chen et al., 2022; Nguyen and Kim, 2012; Tao and Qiu, 2015) based on the priority factors used in real port operations are also included for the comparison. The details can be found in Table 5.

Table 6
The performance of proposed method in comparison with different benchmark algorithms.

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	**26854**	**20856**	**35374**	**42299**	**24314**	**26142**	**29306**

5.3. Experiment Results

5.3.1. Results of the Proposed DRL Approach

During the training process, a set of instances from the 6 different configurations listed in Table 4 are generated. The metrics used for evaluation is the total idle time periods (in seconds) of all QCs. For the test experiments, same configurations are adopted. For each test configuration, the algorithms were run 100 times with different random seeds. The comparison results can be found in Table 6.

It can be seen that the proposed DRL method outperforms all other 9 algorithms for all configurations. The simple heuristic dispatching rules (top 6 in Table 6) fail to achieve competitive performance since most of them only greedily consider one prioritized factor such as truck number or traveling distance. Manual heuristic performs better than those simple dispatching rules because it considers supply-demand information on both spatial and temporal perspectives. Genetic programming based approach could further improve the result due to its ability to repeatedly refine the policy through an evolutionary process and construct dispatching policy based on all possible real-time observations accordingly. DRL-HH is marginally better than the genetic programming based approach and obtain the second best results on average for all configurations. Although DRL-HH can balance the long-term and short-term rewards, its performance may still heavily rely on the low-level heuristic design. Sometimes, the actions for the low-level heuristic are too restrictive, and this may cause DRL-HH fail to explore some promising regions. In contrast, our proposed method is more effective since it can directly search for specific actions. On average, our proposed method obtains a significant improvement (6.9%) compared with the DRL-HH method. For the subsequent experiments, 3 representative benchmark algorithms, manual heuristic (Chen et al., 2016), genetic programming (GP) based approach (Chen et al., 2020) and DRL-HH (Zhang et al., 2021) are selected for further analysis.

5.3.2. Generalization

To evaluate the generalization performance of the proposed algorithm, a set of customized problem instances with different configurations is generated for testing. The testing instances are therefore more abundant and many of them are unseen by the agent in the training process. The testing instance design follows the principle of control variables. That is to say, when changing one of the variables (number of trucks, tasks, QCs, stacking mode and yard distribution), other variables are kept the same. All variables are the same as the base configuration (Config 2: 60 trucks, 400 tasks, 16 QCs, mixed storage, and distributed yard) except the changing one. The results are reported in Table 7.

Again, the proposed DRL method outperforms benchmark manually heuristic, GP and DRL-HH method in all testing instances. The results demonstrate the generalization ability of the proposed method to handle various of real-world scenarios.

Some observations through the experiments are deserved to be discussed. Generally, the performance of the dispatching algorithms heavily rely on the truck-QC ratio. High truck-QC ratio can make the QC operation smoother, and consequently reduce the QC idle time. This can be confirmed by the positive correlation between number of QCs and the objective, and the negative correlation between number of trucks and the objective, respectively. Apart from the objective value, the improvements over the benchmark for GP, DRL-HH and our method are also negatively correlated to truck amount. As we mentioned earlier, the optimization space of the examined problem is sensitive to the truck amount, high truck-QC ratio may make less space for algorithm's improvement.

Table 7
The performance of proposed method in comparison with manual heuristic and a DRL-HH method under different instance configurations.

Configuration	Heuristic Method (Benchmark)	Obj value / Imp (%) against benchmark		
		GP	DRL-HH	Ours
60 Trucks	24390	23269 4.6%	22641 7.2%	21153 **13.3%**
80 Trucks	16716	15980 4.4%	15946 4.6%	14958 **10.5%**
100 Trucks	13658	13423 1.7%	13246 3.0%	12730 **6.8%**
400 Tasks	24390	23269 4.6%	22641 7.2%	21153 **13.3%**
800 Tasks	46628	44583 4.4%	43425 6.9%	40598 **12.9%**
1200 Tasks	67735	65201 3.7%	64280 5.1%	64212 **5.2%**
8 QCs	7786	7541 3.1%	7412 4.8%	7188 **7.7%**
12 QCs	13972	13523 3.2%	13012 6.9%	12528 **10.3%**
16 QCs	24390	23269 4.6%	22641 7.2%	21153 **13.3%**
20 QCs	30850	28912 6.3%	28592 7.3%	25984 **15.8%**
24 QCs	40145	38123 5.0%	37816 5.8%	35154 **12.4%**
28 QCs	49700	48325 2.8%	47431 4.6%	44133 **11.2%**
Mixed Storage	24390	23269 4.6%	22641 7.2%	21153 **13.3%**
Separate Storage	22597	21332 5.6%	21376 5.4%	19704 **12.8%**
Centralized Yard	26920	26230 2.6%	25816 4.1%	25062 **6.9%**
Distributed Yard	24390	23269 4.6%	22641 7.2%	21153 **13.3%**

The improvement of the proposed method over benchmark drops noticeably from the instance of 800 tasks to instance of 1200 tasks. This can be explained in terms of QC operation logic. The objective value could only be reduced by keeping all QCs as busy as possible without idle waiting. However, long QC task list makes it vulnerable to relatively poor dispatching policy and long interruption of QC operation may happen in such situation. This is also the experiment that performance of DRL-HH is closest to the proposed DRL method and DRL-HH shows its relative robustness towards the task size.

The improvements of the proposed DRL method over benchmark in mixed and separated storage mode are basically the same. The objective value in mixed storage mode is higher than the separated storage mode. This can be caused by the relative high yard congestion level and more container relocation operations in the mixed storage mode. The performance of the proposed method in experiment of distributed yard is better than the centralized yard in terms of both improvement ratio and objective value. This is caused by higher congestion level in the centralized yard compared with the distributed yard.

To further demonstrate the generalization performance of the proposed method, we trained the dispatching policy on the base configuration (Config 2: 60 trucks, 400 tasks, 16 QCs, mixed storage, and distributed yard) only and test it with a set of unseen configurations. The experiment results show that the policy trained on one single configuration could also be generalized to solve the unseen problem configurations. We exclude DRL-HH for this experiment since its input is fixed so that the single model cannot handle instances with different QC numbers.

Table 8 presents the generalization performance of our proposed method and GP in comparison with benchmark. Generally, the policy that is trained on base configuration by our method could outperform both GP and the benchmark method in all unseen test cases. It can be also observed that our proposed method has a slighter over-fitting effect compared with GP. This is benefited from the novel state design which could make RL agent effectively observe the environment changing in the unseen instances and is also benefit from the proposed network structure which incorporates the scenario-independent expert knowledge that could deal with different input QC sizes. The state feature design reveals several key factors for truck dispatching policy. Features like the total amount of trucks working for each QC, the total amount of trucks heading to each QC and the queue length of each QC may reflect the situation of future truck supply. The travelling distance of the truck can be used to estimate the travelling time for the truck to arrive at the target QC. QC type is considered as prior knowledge for the agent since the supply-demand patterns for different types of QC may vary a lot. Target yard queue length could provide the information of yard congestion level some time in the future. The QC-yard distance of the second task can expose the information of QC's future tasks since the information (distance to the target yard) of QC's future task list could greatly affect the probability of current QC

Table 8
The generalization performance of proposed method in comparison with manual heuristic.

Configuration	Heuristic Method (Benchmark)	Obj value / Imp (%) against benchmark	
		GP	Ours
8 QCs	7786	7568 2.8%	7483 **3.9%**
12 QCs	13972	13228 5.3%	12039 **13.8%**
16 QCs (base)	24390	21032 13.8%	20256 **16.9%**
20 QCs	30850	28903 6.3%	27660 **10.3%**
24 QCs	40145	38625 3.8%	37152 **7.5%**
60 trucks (base)	24390	21032 13.8%	20256 **16.9%**
70 trucks	20762	19035 8.3%	18597 **10.4%**
80 trucks	16716	16035 4.1%	15258 **8.7%**
90 trucks	14632	14345 2.0%	13352 **8.7%**
100 trucks	13658	13758 -0.7%	13130 **3.9%**

Table 9
The performance of proposed method in comparison with 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%**

selection. The network structure for handling different input lengths could also help RL agent get rid of the influence of QC with empty task list.

Apart from the unseen problem instances, the proposed DRL method also shows robustness to unknown uncertainties. As introduced earlier, the truck speed at different areas and crane operation time are the uncertain factors for the examined problem. In the training environment, crane operation time are non-deterministic which follows the same setting of Zhang et al. (2021) while the truck speed is assumed to be constant. In the testing environment, the truck speed is treated as unknown uncertainty for the RL agent. The trained models for DRL-HH, GP and our method are evaluated in the testing environment with the unknown uncertainty. The comparison results against manual heuristic can be found can be found in Table 9.

It can be seen that DRL-HH are not competitive and fails to outperform heuristic solutions in some cases. It is not surprising as the unseen truck speed uncertainty at yard side may further aggravate yard congestion effect and the low-level heuristics used for DRL-HH may limit its exploration ability. This is because unseen state generated in new environment and the truck speed uncertainty at yard side further aggravate yard congestion effect. 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 our proposed method has the potential to be deployed in the real-world port operation environment.

5.3.3. 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 convergence speed. Table 10 shows the performance of our method with and without the expert network. For agent without expert network, a single cross-scenario network without gate component is adopted (See Fig. 5 for details). As we can see, compared with the single cross-scenario network, the agent with expert network has the better performance in most cases except Config 4. Using a single network to obtain an uniform policy for different scenarios (parameterized environments) can be problematic. A certain policy that work well for some particular scenarios may perform badly in others (e.g. Config 3) due to possible overfitting. Our approach alleviates such defect

Table 10
The performance of proposed method in comparison with or without 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** %

by incorporating the prior knowledge into RL agent. The results are more balanced among different environment configurations since the expert network is trained by scenario-insensitive data through imitation learning.

Fig. 6 shows the convergent performance of agent without the expert network. 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 the better score steadily. In contrast, as can be found in Fig. 7, the agent with the expert network converge rapidly at early stages and achieve heuristic level after only 100 iterations. Benefit from the prior knowledge incorporated into the network, the agent uses only 650 iterations to obtain better score than that of the agent without expert network.

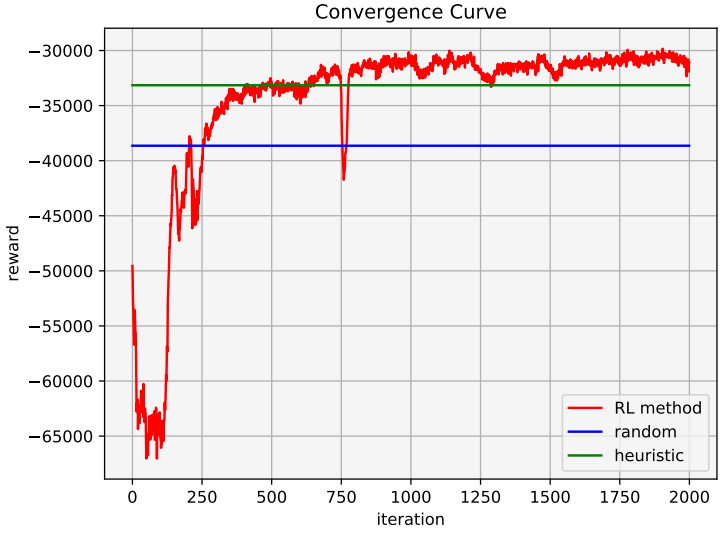

Figure 6: The convergence performance of proposed method in comparison with manual heuristic.

5.3.4. Comparative Results of the Proposed DRL Approach with Offline Solution

To further examine the limits of the solutions obtained by the proposed method, some selected instances are solved in offline manner. Each instance consists of one specific problem configuration and one fixed random seed, which therefore guarantees the unique results for a fixed sequence of actions. Consequently, each configuration can be viewed as a static problem instance. For different time steps, a multi-start local search-based heuristic is deployed to modify the dispatching sequence and iteratively refine the results. The total time limit given for instance is set to 72 hours. The final result obtained by the local-search could be considered as the offline solution for the specific testing instance. The results can be found in Table 11.

Generally, the average gap between the solutions obtained by our algorithm and the offline solutions is around 6%. The experimental results further demonstrate the effectiveness of the proposed approach.

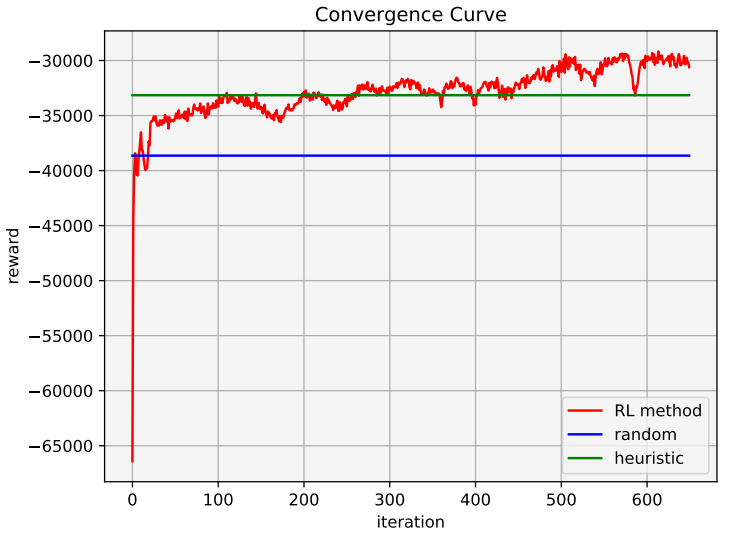

Figure 7: The convergence performance of proposed method in comparison with manual heuristic using Imitation Learning.

Table 11
Comparative results of the proposed method with estimated upper bound.

	Estimated Upper Bound	Our Method	Gap(%)
Instance 1	25832	27350	5.88%
Instance 2	19932	21042	5.57%
Instance 3	34241	35640	4.09%
Instance 4	39895	42385	6.24%
Instance 5	22839	24612	7.76%
Instance 6	24221	26610	9.86%

5.4. Managerial Insights

Apart from the great adaptability and generalization performance of the proposed RL method in multi-scenario cases, these experiments also provide some useful insights for the container terminal management. It is not difficult to see that truck-QC ratio is a crucial factor because it not only has an impact on QC utilization but also can limit the optimization space for RL algorithm. In our experiments, for a given number of QCs, less trucks could make high improvement for RL dispatching policy against the benchmark, but the objective value may also increase as an side effect. Ideally, the port managers should set an acceptable objective level and also leave some space for RL dispatching policy to play a role in it. The optimal truck-QC ratio should be dynamic and there is a mapping between it and different scenarios. For example, in this work, we empirically find this ratio within range [4.5, 5.2] can achieve relative fair objective value and RL improvement in the base configuration setting. Furthermore, the performance of dispatching policy is also sensitive to container spacial-related factors. In our experimental settings, stack mode makes little effect on the algorithm performance since there is no outer-truck interference. Distributed yard can offer bigger space for RL performance improvement compared to centralized yard. Apart from the aforementioned stack mode and yard distribution mechanism, the relative locations of involved yards and task sharing schemes among neighboring QCs are also important. Some of such factors are related to container storage space allocation which is another popular COP in container terminal. In this work, some of these factors are embedded into the configurations for RL dispatching policy evaluation but a sophisticated configuration design along with the proposed RL dispatching policy should further facilitate the container terminal operation efficiency.

Figs. 8 and 9 present the generalization performance of our proposed method in comparison with benchmark based on two metrics, namely total idle time (left Y-axis) which is the objective value of the examined problem and average makespan per QC (right Y-axis). In experiments with various QC amount (Fig. 8), it is obvious that less QCs could make

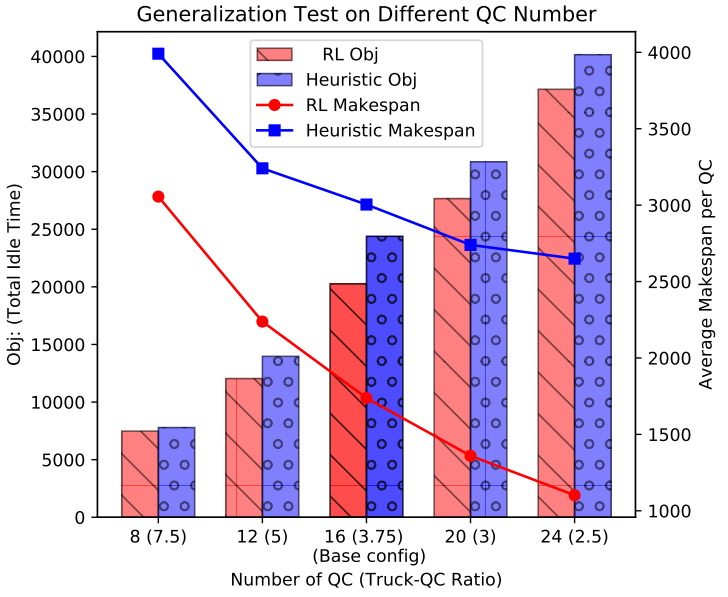

Figure 8: The performance of proposed method trained on single configuration in comparison with benchmark under different QC amount.

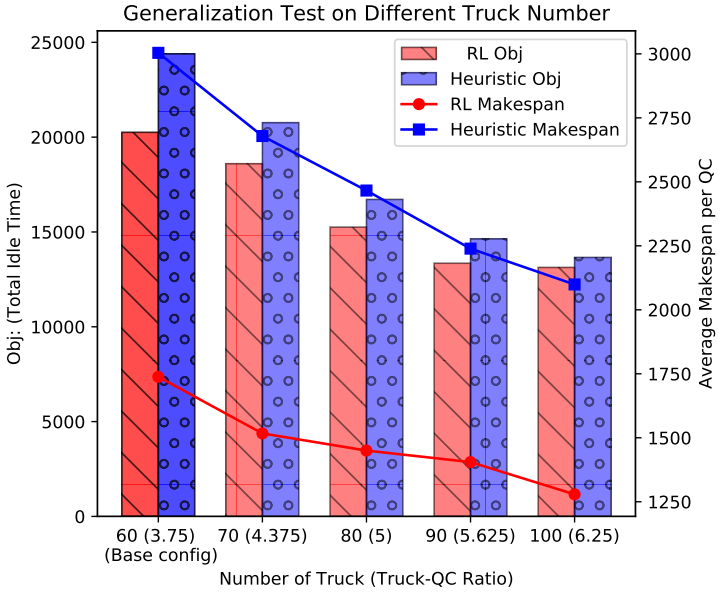

Figure 9: The performance of proposed method trained on single configuration in comparison with benchmark under different truck amount.

objective value (idle time) decreased on both methods, which is caused by high truck-QC ratio. However, the average QC makespan is increased because of less tasks per QC. Therefore, deploying QC is faced with the trade-off between QC idle time and QC makespan. In experiments with different truck amount (Fig. 9), both metrics are decreased at the expense of operational cost (truck deployment, labour cost, etc.).

The experiment further reveals trade-offs among different equipment deployments. It is always strategic to balance cost and efficiency in practical container terminal. Our proposed method, together with the simulation system, is

capable of exploring optimal equipment deployment strategy under different scenarios. It can provide a reliable reference for terminal operators to schedule internal resources to increase the overall operation efficiency.

6. Conclusion and Future Work

In this work, we propose a deep reinforcement learning based approach to solve a real-life truck dispatching optimization problem in the maritime container terminal. The examined problem is formalized as a MDP which takes into account different optimization scenarios and uncertainties stemming from the real-world port operations. The proposed Real2Sim DRL approach is data-driven, which barely rely on any exogenous forecasts. The proposed approach respects the nature of the examined problem by posing it multi-dimensional states derived from Real2Sim based simulator and it can devise more effective policies by exploiting the entire action domain. Quantitative results have highlighted several advantages from the proposed method. Firstly, thanks to our Real2Sim environment settings and novel dual-network structure, it has the ability to handle more complex and non-deterministic factors and can achieve the state-of-the-art results compared with existing methods. Secondly, thanks to the LSTMs and the multi-head attention, it has good generalization ability in both unseen instances and unknown uncertainties. Furthermore, our method together with Real2Sim based simulation can provide a reliable reference for terminal managers to better schedule internal resources under different operation scenarios.

The proposed DRL-based method could be further improved in several ways. For example, the proposed state design does not explicitly consider the constraints of the problem. 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 a constraint can cause considerable amount of QC idle time even with a well-trained dispatching policy. Few studies focus on enhancing RL agent's constraint awareness/handling abilities for COPs in the literature and it would be a promising future direction. We could also further improve our network structure by using more than one expert net so that RL agent can consider different types of expert knowledge by leveraging attention mechanism. Additionally, it would be interesting to investigate more sophisticated policies for yard crane scheduling in place of the current "first-come first-served" policy to further improve the productivity. Finally, the examined problem can be extended to a multi-objective optimization problem that takes the truck travelling distances and equipment movement distances into considerations.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 72071116). This work is also supported by Ningbo Natural Science Foundation (Project ID 2023J194).

References

Adi, T. N., Iskandar, Y. A., and Bae, H. (2020). Interterminal truck routing optimization using deep reinforcement learning. *Sensors*, 20(20):5794.

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.

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.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

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.

Cappart, Q., Moisan, T., Rousseau, L.-M., Prémont-Schwarz, I., and Cire, A. A. (2021). Combining reinforcement learning and constraint programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3677–3687.

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, 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*.

Chen, X., Bai, R., Qu, R., Dong, H., and Chen, J. (2020). 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. and Tian, Y. (2019). Learning to perform local rewriting for combinatorial optimization. In *Adv. Neural Inf. Process. Syst.*, volume 32, pages 6278–6289.

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.

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.

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

Haydari, A. and Yilmaz, Y. (2020). Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*.

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.

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.

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.

James, J., Yu, W., and Gu, J. (2019). Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3806–3817.

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.

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.

Kong, W., Liaw, C., Mehta, A., and Sivakumar, D. (2019). A new dog learns old tricks: RL finds classic optimization algorithms. In *International Conference on Learning Representations*.

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

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.

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

Liu, Z., Li, J., and Wu, K. (2020). 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)*.

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

Ma, Y., Li, J., Cao, Z., Song, W., Zhang, L., Chen, Z., and Tang, J. (2021b). Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems*, 34:11096–11107.

Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

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.

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.

Ozsoydan, F. B. and Gölcük, İ. (2023). A reinforcement learning based computational intelligence approach for binary optimization problems: The case of the set-union knapsack problem. *Engineering Applications of Artificial Intelligence*, 118:105688.

Qin, T., Du, Y., Chen, J. H., and Sha, M. (2020a). Combining mixed integer programming and constraint programming to solve the integrated scheduling problem of container handling operations of a single vessel. *European Journal of Operational Research*, 285(3):884–901.

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

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

Rusu, A. A., Večerík, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. (2017). Sim-to-real robot learning from pixels with progressive nets. In *Conference on robot learning*, pages 262–270. PMLR.

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

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.

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.

Sun, P. Z., You, J., Qiu, S., Wu, E. Q., Xiong, P., Song, A., Zhang, H., and Lu, T. (2022). Agv-based vehicle transportation in automated container terminals: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):341–356.

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

Tang, L., Zhao, J., and Liu, J. (2014). Modeling and solution of the joint quay crane and truck scheduling problem. *European Journal of Operational Research*, 236(3):978–990.

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.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE.

Tu, C., Bai, R., Aickelin, U., Zhang, Y., and Du, H. (2023). A deep reinforcement learning hyper-heuristic with feature fusion for online packing problems. *Expert Systems with Applications*, page 120568.

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.

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. *Advances in neural information processing systems*, 28.

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

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.

Xin, J., Meng, C., D'Ariano, A., Wang, D., and Negenborn, R. R. (2021a). Mixed-integer nonlinear programming for energy-efficient container handling: formulation and customized genetic algorithm. *IEEE Transactions on Intelligent Transportation Systems*.

Xin, L., Song, W., Cao, Z., and Zhang, J. (2020). Step-wise deep learning models for solving routing problems. *IEEE Transactions on Industrial Informatics*, 17(7):4861–4871.

Xin, L., Song, W., Cao, Z., and Zhang, J. (2021b). Neurolkh: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem. *Advances in Neural Information Processing Systems*, 34:7472–7483.

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, L.-W., Ye, R., Huang, S.-Y., and Hsu, W.-J. (2005). Mixed integer programming models for dispatching vehicles at a container terminal. *Journal of Applied Mathematics and Computing*, 17:145–170.

Zhang, R., Zhang, C., Cao, Z., Song, W., Tan, P. S., Zhang, J., Wen, B., and Dauwels, J. (2022a). Learning to solve multiple-tsp with time window and rejections via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):1325–1336.

Zhang, Y., Bai, R., Qu, R., Tu, C., and Jin, J. (2021). A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. *European Journal of Operational Research*.

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*.

Zheng, J., He, K., Zhou, J., Jin, Y., and Li, C.-M. (2021). Combining reinforcement learning with lin-kernighan-helsgaun algorithm for the traveling salesman problem. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 12445–12452.

Zheng, X., Liang, C., Wang, Y., Shi, J., and Lim, G. (2022). Multi-agv dynamic scheduling in an automated container terminal: A deep reinforcement learning approach. *Mathematics*, 10(23):4575.

Zhou, C., Ma, J., Douge, L., Chew, E. P., and Lee, L. H. (2023). Reinforcement learning-based approach for dynamic vehicle routing problem with stochastic demand. *Computers & Industrial Engineering*, page 109443.

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.

Zou, G., Tang, J., Yilmaz, L., and Kong, X. (2022). Online food ordering delivery strategies based on deep reinforcement learning. *Applied Intelligence*, pages 1–13.