Annals of Operations Research
https://doi.org/10.1007/s10479-024-05927-4

ORIGINAL RESEARCH

Optimizing inland container shipping through reinforcement learning

Vid Tomljenovic[1] · **Yasemin Merzifonluoglu**[1] · **Giacomo Spigler**[2]

Received: 30 December 2022 / Accepted: 28 February 2024

Abstract
In this study, we investigate the container delivery problem and explore ways to optimize the complex and nuanced system of inland container shipping. Our aim is to fulfill customer demand while maximizing customer service and minimizing logistics costs. To address the challenges posed by an unpredictable and rapidly-evolving environment, we examine the potential of leveraging reinforcement learning (RL) to automate the decision-making process and craft agile, efficient delivery schedules. Through a rigorous and comprehensive numerical study, we evaluate the efficacy of this approach by comparing the performance of several high-performance heuristic policies with that of agents trained using reinforcement learning, under various problem settings. Our results demonstrate that a reinforcement learning approach is robust and particularly useful for decision makers who must match logistics demand with capacity dynamically and have multiple objectives.

Keywords Reinforcement learning · Dynamic fleet assignment · Inland logistics

1 Introduction

Recent advances in information technology and logistics systems have resulted in a significant expansion of firms' distribution networks. With the marketplace becoming increasingly competitive due to globalization, rapid technological advancement, short product life cycles, and high customer expectations, companies must now provide high-quality, innovative products at affordable prices in a timely manner. As supply networks become more complex, the amount of cargo produced and needing to be transported increases, as does the number of delivery routes and transportation mode options. With such complexity in logistics operations, state-of-the-art decision models are necessary to help firms use data in smart ways to precisely respond to supply or demand patterns.

✉ Yasemin Merzifonluoglu
Y.Merzifonluoglu@tilburguniversity.edu

[1] Department of Econometrics and Operations Research, Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands

[2] Department of Cognitive Science and Artificial Intelligence, Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands

Published online: 26 March 2024

In this study, we investigate a complex logistics problem that was communicated to us by our industrial partner.[1] This company specializes in delivering sophisticated software solutions that enable their clients to gain comprehensive insight and control over every aspect of their logistical chain, including inland shipping. Inland shipping refers to the practice of transporting goods via inland waterways (such as rivers and canals) using specialized vessels designed for this purpose. This mode of transportation plays a crucial role in the transportation of goods within Europe, as more than 37,000 kms of waterways connect hundreds of cities and industrial regions. In comparison with other modes of transportation, inland waterway transport offers a competitive and environmentally-friendly alternative. The energy consumption per kilometer/ton of transported goods is approximately 17% of that of road transport and 50% of rail transport (Inland waterways, 2022). In addition to its competitive and environmentally-friendly nature, inland waterway transport is also a safe alternative, particularly for the transportation of dangerous goods. The integration of inland shipping into intermodal logistics networks is highly encouraged as it helps to alleviate congestion on overburdened road networks. As a result, the scale of inland shipping and container volumes in the ports have seen a significant increase in recent times. To manage this growth, all links in the chain are continually seeking ways to improve, such as through integrated planning or the digitization of processes. Sophisticated decision tools are urgently required in order to fully optimize the inland container shipping chain and digitize this process.

In this study, we focus on optimizing the inland container delivery problem. Barge operators have a fleet of vessels for carrying cargo, which in this case consists of a bulk of containers with various goods. The arrival time of each bulk is known with a short advance notice. The challenge is to transport these containers to their destination ports before the due date. An inland shipping freighter is responsible for connecting demand and supply and must decide which ship to dispatch, which containers to pick up, and in what order they should be delivered to their destination. The objective is to fulfill customer demand for cargo in a timely manner while minimizing costs. To achieve this, a schedule is created that includes a route and the collection of containers for each ship. This schedule is updated daily (or more frequently if needed), taking into account new orders that enter the system. Currently, this process is carried out manually by teams of industry experts, which is time-consuming and difficult to source due to the required experience and market understanding. Automating this process through a digital platform that creates high-quality, robust schedules would bring significant benefits.

The Inland Container Delivery Problem belongs to the family of Dynamic Fleet Assignment Problems (DFAP), where load assignment decisions must be determined in conjunction with vehicle routing (Min & Kang, 2021). Due to the uncertainties involved in the problem, such as unpredictable arrival of new orders, fluctuating weather conditions, port congestion and more, a platform must be able to recommend decisions quickly. Nervousness can be a significant issue when deterministic planning is adopted for the container assignment problem. In this problem shipment requests are known only closer to their arrival date, and due dates are known when shipments arrive. When deterministic models are employed to solve this problem, they typically generate static schedules and routing decisions based on expected shipment requests and due dates. However, these models struggle to adapt to the inherent uncertainties, causing frequent changes in routing, scheduling, and resource allocation decisions as the actual arrivals and due dates deviate from initial expectations. This can lead to operational inefficiencies, and extended delays, and an overall lack of resilience in the system.

[1] Name of the company is disclosed due to confidentiality reasons.

To address this, a decision-making tool is developed to enable sequential decision-making under uncertain environments.

Reinforcement Learning (RL) has been successfully applied to numerous sequential decision-making scenarios with large amounts of data (POR, 2022), making it an ideal framework for the Inland Container Delivery Problem. RL is a type of machine learning in which a computer agent takes actions and learns an optimal behavior by correlating feedback from the environment with its past actions. Although dynamic fleet assignment problems have been widely studied, there are only few studies that address them using a RL approach. This study provides a RL framework that is robust and adaptable to various scenarios with multiple objectives, such as minimizing cost and maximizing customer service. We have also developed several hand-crafted heuristic methodologies as benchmarks to evaluate the performance of the RL agents. A real-life case study utilizing historical data provided by a company is presented. This approach is currently being integrated within the company's decision support system to offer valuable guidance for future planning efforts.

This paper is structured as follows: Sect. 2 reviews existing literature, Sect. 3 introduces and explains the concept of RL and its theoretical background, Sect. 4 describes the case study and implementation of the model with RL, together with the heuristic policies, Sect. 5 compares and evaluates the RL agent-obtained policies using various performance indicators, and Sect. 6 provides the conclusions.

2 Literature review

In this section, we provide an overview of the existing literature on reinforcement learning, dynamic fleet assignment problems and application of reinforcement learning on dynamic fleet assignment problem is provided.

2.1 Deep reinforcement learning

Dynamic fleet assignment problems can be classified under the sequential decision making problem for which Markov Decision Process (MDP) is effective in capturing the stochastic and dynamic nature of this setting (Puterman, 2009). It is established that, modeling and solving a large-scale MDP model suffers from the *curse of dimensionality* due to the size of action and state space. To overcome this issue, Approximate Dynamic Programming (ADP) approximates the optimal solution of MDP's by iteratively updating the approximation of value functions. However, ADP methods are model-based, requiring an exact mathematical model of the MDP. Reinforcement learning (RL) methods are similar to ADP but do not require explicit knowledge of the MDP model (i.e., they are referred as *model-free*) (Sutton & Barto, 2018).

Deep reinforcement learning (DRL) integrates reinforcement learning with deep learning. DRL utilizes deep neural networks to estimate the value or policy functions used in traditional reinforcement learning, allowing it to learn to solve complex problems for which a linear function is insufficient. Neural networks further enable a reinforcement learning agent to generalize its experience to novel, unseen states, which is crucial for large state and action spaces. Thus, DRL provides a methodology for approximating and solving large Markov Decision Processes for which traditional dynamic programming methods are impractical.

Several DRL algorithms have been presented in the literature, each with different features and performance. Notable examples are Deep Q-learning (DQN) (Mnih et al., 2015), RAIN-

BOW (Hessel et al., 2018), and Asynchronous Advantage Action-Critic (A3C) (Mnih et al., 2016). In this work, we choose to use Proximal Policy Optimization (PPO) (Schulman et al., 2017) due to its state-of-the-art performance on a variety of domains and its low number of hyperparameters.

2.2 Dynamic fleet assignment problems

In dynamic fleet assignment problems, a fleet of resources must be managed and assigned to jobs or tasks in a manner that ensures that all tasks are completed as efficiently as possible. Table 1 summaries the literature on dynamic fleet assignment problem.

Yang et al. (1999) tackle a problem of assigning trucks in real-time to pick up and deliver loads using mathematical programming methods. Unfortunately, this solution method is only applicable to small problems, as it requires solving a separate mathematical program at each time-step. Despite this limitation, several real-time strategies are presented and compared to the optimal solution. The main aspect of those strategies is the trade-off between empty distance travelled (excess transportation cost) and waiting time to serve a demand. The trade-off between the cost and quality of service (responsiveness) is a recurring theme in our study as well. Hyland and Mahmassani (2018) study real-time assignment of autonomous vehicles to traveller requests. The fleet operator assigns vehicles to open traveler requests in real-time as traveler requests enter the system dynamically and stochastically. They implement and test multiple strategies and conclude that intelligent and optimization-based strategies outperform simple heuristics, especially in the higher demand scenarios. Cheung and Hang (2003) solve the problem of transportation of air-cargo freight forwarders with various operational issues such as heterogeneous vehicles, multiple trips per vehicle, and penalty for early arrival at customer sites. This study considers the hard time-window case (where the penalty costs are infinity) and suggests a window-sliding procedure that makes use of the network structure. A more general case of assigning drivers to tasks is tackled by Cheung et al. (2005). They formulate a dynamic stochastic decision model where tasks have (soft) time windows and uncertain duration. Using an adaptive labeling solution procedure which can handle various practical constraints and work rules, they manage to find good solutions to the general case.

Moving closer to the exact problem this paper tackles, Daham et al. (2016) focus on an inland transportation problem where trucks load and deliver standardized containers. The problem is solved using mixed integer linear programming model, which, as mentioned before, is computationally expensive for large instances and is hard to implement in real-life where decisions have to be made often and quick. Mahmassani et al. (2000) approach a similar problem of assigning truck to cargo delivery requests. They provide strategies that take into account real-time information and change assignments if the environment changes.

2.3 Reinforcement learning for dynamic fleet assignment

Deep reinforcement learning has recently shown promise in learning approximate solution strategies for complex combinatorial optimization problems. For example, DRL has been successfully used to achieve super-human performance in optimizing chip placement (Mirhoseini et al., 2021), dynamic network routing (Sun et al., 2019), the covering salesman (Li et al., 2021) and the vehicle routing problems (Nazari et al., 2018; Raza et al., 2022). Interested reader can find more on the topic in recent surveys, for example (Mazyavkina et al., 2021; Wang & Tang, 2021).

Further, previous research has investigated the application of reinforcement learning to schedule the route of delivery vehicles. The overall framework is similar, as cargo ships can be considered as vehicles and customer demand can be replaced by containers that are located in terminals and need to be delivered to their respective destination ports. The primary difference is that cargo ships can be assigned to and load multiple containers, while in most research, it is assumed that a vehicle can only handle a single demand request at a time. Oda and Joe-Wong Oda and Joe-Wong (2018) explore a model-free approach to optimizing dispatch policies of taxi drivers. They use a Deep Q-network (DQN) to learn an optimal dispatch policy through experience for each individual driver, and they conclude that it performs better in terms of customer service than alternative policies. Additionally, they suggest that there is limited value in coordinating multiple vehicles, since each vehicle learns and behaves independent of others. Min and Kang (2021) study a dynamic fleet assignment problem related to matching demand and supply. They formulate the problem as a Markov decision process (MDP) and compute a policy using RL and function approximation. Their results show that a RL policy performs better than traditional policies, such as first-come-first-serve (FCFS). Lin et al. (2018) work on a large-scale fleet management problem using reinforcement learning. They work in a multi-agent framework, where each member of a fleet acts and learns as an independent agent, and they demonstrate that explicit coordination among a large number of agents can be achieved adaptive to different contexts.

To the best of our knowledge, the only prior study that has employed reinforcement learning (RL) in a similar inland shipping problem is Schevenhoven's work Schevenhoven (2021). In his thesis, (Schevenhoven, 2021) used RL to demonstrate the suitability of a sequential decision-making approach for solving such problems. Their study, while relevant, focused on a slightly different problem scenario. In their case, each cargo was categorized as small, medium, or large, and each ship could only pick up one cargo at a time. Agents had a broader range of available actions, including moving to another port and selecting cargo to assign to the ship. This approach allowed us to focus on addressing the complexity of the problem while maintaining practical relevance. When implementing multi-agent RL, Schevenhoven employed the Proximal Policy Optimization (PPO) algorithm. However, the results of their approach did not exhibit significant improvements compared to a random policy. While the adoption of a multi-agent approach may have contributed to this outcome, the author identified three potential issues: state representation, the reward system, and hyper-parameter settings. To address these concerns, we carefully designed our state representation, considering the trade-off between providing necessary information and managing the problem's size. For example, we incorporated information about *ships sailing to port* into our state representation. In contrast to the previous study's focus on a limited set of reward functions primarily related to delivery performance, our study explored multiple reward functions and a combination of objectives. We aimed to balance the minimization of costs with the maximization of customer service. Additionally, we trained an agent in a single-ship environment, initially using a simple reward function and later fine-tuning it with more complex reward functions.

Table 1 Summary of literature for dynamic fleet assignment problem

Study	Problem	Approach	Details and key findings
Yang et al. (1999)	Fleet assignment	Mathematical programming	Real-time truck assignment for pick-up and delivery, trade-off between empty distance and waiting time. Suitable for small problems
Hyland and Mahmassani (2018)	Assignment for shared-use autonomous vehicles service	Agent-based simulation framework	Intelligent and optimization-based strategies outperform heuristics, especially in high-demand scenarios
Cheung and Hang (2003)	Driver-task assignments in container transportation	Stochastic optimization framework	Assigning drivers to tasks with (hard) time windows and uncertain durations. Solved the minimum cost flow problems iteratively
Cheung et al. (2005)	Driver-task assignments in container transportation	Stochastic optimization framework	Assigning drivers to tasks with (soft) time windows and uncertain durations, using adaptive labeling solutions to handle constraints
Daham et al. (2016)	Container transportation	Mixed integer linear programming	Inland transportation problem involving trucks and standardized containers, computational challenges for real-time decision-making
Mahmassani et al. (2000)	Cargo delivery	Hybrid local optimization and simulation	Strategies for truck assignment to cargo delivery requests that adapt to real-time changes in the environment
Oda and Joe-Wong (2018)	Dispatch policies	DQN	Model-free approach using DQN to optimize dispatch policies, emphasizing individual driver performance
Min and Kang (2021)	Fleet assignment	MDP, RL	RL-based policy outperformed traditional policies, especially first-come-first-serve (FCFS)
Lin et al. (2018)	Fleet management	Multi-Agent RL	Demonstrated successful coordination among a large number of agents for fleet management
Schevenhoven (2021)	Inland shipping	Multi-Agent RL	Demonstrated the suitability of RL for sequential decision-making. Poor performance of RL implementation
This paper	Inland shipping	Multi-Agent RL and RL	Multiple reward functions, RL performed favorably compared to the heuristics adjusting to unpredictable conditions

3 Deep reinforcement learning framework for container assignment problem

In this section, we define the problem of inland container delivery as a Markov Decision Process (MDP) using a reinforcement learning framework. We provide the notation we use to formalize the problem.

3.1 Problem definition

We consider the shipping planning problem for a planning horizon of T timesteps and a set of P ports, each reachable from all others. The total travel duration for a ship moving from port i to port j is denoted as $d_{i,j}$, and includes time for loading and unloading cargo. Port capacity is not considered.

We have a fleet represented by a set Z non-identical ships, where the capacity of ship s is denoted as K_s (in terms of a standard container size). It is important to note that, in practice, ships have both a volume and weight capacity. The volume capacity is usually the more restrictive constraint. Therefore, in this study, the capacity of the ships will be assumed to be determined solely by their volume capacity.

The decision maker is presented with a set of deterministic demand requests for the planning horizon. Each request consists of a bulk of containers that must be delivered from one port to another within a given deadline. The set B denotes the set of confirmed shipping requests, each of which has the following attributes: origin port (b_{op}), destination port (b_{dp}), number of standard size containers (b_{nc}), weights of each container (b_{wc}), arrival time to the origin port (b_{ao}) and delivery due date time to the destination port (b_{dd}).

The decision maker's goal is to optimize the assignment of requests to ships and create a plan for each ship for the planning horizon, so as to minimize the total cost of container delivery while ensuring that all containers are delivered before their specified due dates. The cost is calculated based on the total travel times of the ships involved in the delivery process.

The task of assigning confirmed shipping requests to ships is a complex combinatorial optimization problem. It requires careful consideration and planning, as new, unexpected orders are continuously added at random. Thus, the decision maker must adjust the schedule dynamically in response to these new requests. This problem can be classified as a sequential decision-making problem and modelled as a Markov Decision Process (MDP). In the following section, we will define the MDP corresponding to this task.

3.2 Container assignment problem as a Markov decision process

We formulate the container assignment problem as a Markov Decision Problem defined by the tuple $(\mathcal{S}, \mathcal{A}, p, \mathcal{T}, r, \rho_0, \gamma)$ (Sutton & Barto, 2018) in a reinforcement learning setting. We consider a finite temporal horizon T and sample the initial state of the environment (e.g., initial set of orders to deliver and location of each ship) from the initial state distribution $s_0 \sim \rho_0(s)$. At each timestep $t = 1, \ldots, \mathcal{T}$, the decision making agent selects an action $a_t \in \mathcal{A}$ according to its policy function $\pi : \mathcal{S} \to \mathcal{A}$ which maps states $s_t \in \mathcal{S}$ to actions. The MDP then changes state according to the transition function $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ (i.e., $s_{t+1} \sim p(s_t, a_t)$). Lastly, it returns a reward signal $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ after each transition.

We consider an environment with $|P|$ ports and $|Z|$ ships as defined in the previous section. Each ship is independently controlled using the same shared policy, but with ship-dependent egocentric state representations.

The goal of reinforcement learning is to find an optimal policy π that maximizes the discounted sum of expected rewards

$$\pi^{\star}(s) = \arg\max_{\pi(s)} \mathbb{E}_{\pi}\left[\sum_{t=1}^{\mathcal{T}} \gamma^t r_t | a_t \sim \pi(s_t), s_0 \sim \rho_0(s)\right]$$

In the following sections we describe in detail how the states and actions are represented, followed by a description of the reward function that characterizes the container assignment problem.

3.2.1 Action space

In order to keep the problem manageable, we restrict the agent's actions by limiting them to movement of ships. The action space therefore consists of $|P|$ possible actions, which indicate the possible destination ports a ship can relocate to.

Once a ship arrives at its destination port, after a number of timesteps that depend on the time required for the ship to reach its destination port, it automatically unloads all the containers that need to be delivered. Then, the ship has a decision to make. The ship can choose to relocate to any of the other ports or stay in the current one for an additional time unit. The bulks are then loaded using the policy described below.

After choosing a destination port, the ship is first loaded with all the available bulks that have the chosen port as their destination, prioritizing loading by delivery deadline. Our model also recommends that ships at their current port load all available cargo whenever possible, as their capacity often exceeds the cargo volume at the port. In case the space is insufficient to load all cargo in the port, we employ a random selection process to determine which containers to leave behind,[2]

It is important to note that, following industry practice, a ship cannot be rerouted until it has reached its destination. Therefore, the agent's actions are only applicable to ships that are idle in a port at the current time step.

3.2.2 State representation

At each decision step t, the agent makes decisions for each ship based on a state representation that comprises both the state of the agent and the state of the environment, and that includes all information that the agent may need to make an optimal decision.

Each ship receives its own egocentric state representation vector, which takes into account the individual ship (referred to as the *current ship*). Since we do not consider routing decisions for a ship unless it is currently idle in a port, the state representation includes the port the ship is currently in (referred to as the *current port*).

The state vector for each ship includes the following information for each port $p \in P$:

- *Containers on the current ship with destination port* p (c_p): Percentage of container capacity on the current ship occupied by containers that have to be delivered to port p.

[2] We experimented with alternative loading policies, such as *only load containers for the next port* but these resulted in inefficient ship utilization and missed delivery deadlines. While there may be cases where not loading all containers or making more informed decisions could be beneficial, our *load everything if possible* policy aligns with real-world practices. Furthermore, changing the loading policy would require a different relocation strategy for the agent, which is beyond this paper's scope.

- *Containers at the ship's current port with destination port p* (f_p): Percentage of container capacity ready to be picked up at the current port which needs to be delivered to port p compared to the current ship's capacity. If there are more containers than the current ship's capacity, the value is capped to 1.
- *Containers in other ports* (w_p): Percentage of the capacity that containers ready to be picked up at port p would occupy on the *current ship*.
- *Soon available containers in other ports* (ws_p): Percentage of the capacity that containers soon ready to be picked up (in the time step) at port p would occupy on the current ship.
- *On-board urgency indicator with destination port p* (ud_p): Urgency indicator of the containers that are currently on the *current ship* that have to be delivered to port p. It is calculated as: $ud_p = \max\left(0, 1 - \frac{\text{due date}(p)}{d_{threshold}}\right)$ where due date(p) is the due date of the most urgent container on board that has to be delivered to port p. If the due date farther away than $d_{threshold}$ we consider the delivery to not be urgent. This measure varies from 0 to 1; 0 if not urgent, 1 if the due date already passed.
- *Current port urgency indicator with destination port p* (k_p): Urgency indicator of the containers in current port that have to be delivered to port p. If there are multiple bulks with different due dates needed to be delivered from current port to port p, we take the earliest one. It is calculated the same as in the case of container on board.
- *Urgency indicator in other ports* (u_p): Urgency indicator of the most urgent bulk waiting to be picked up at port p (calculated as described before).
- *Urgency indicator of soon available containers in other ports* (us_p): Urgency indicator of the most urgent bulk available soon to be picked up at port p, 0 if not urgent, 1 if due date already passed.
- *Ships sailing to port* (s_p): Indicator of future ship capacity to be available in port p soon. That is the available capacity of the ships currently sailing there, discounted for their remaining travel time to reach port p. This measure is calculated as $s_p = \sum_{s \in Z_p} \frac{\frac{c_s}{\rho}}{1+t_s^2}$ where Z_p is the set of all ships sailing to port p currently, while c_s and t_s are the remaining capacities and travel times (in hours). This serves as an information about how much available capacity will there be soon in port p. Parameter ρ is selected to keep this measure between 0 and 1. If the value is close to 1, there will be ships in port p soon which have significant capacity available and therefore will be able to load containers from that port. If a ship is considering relocating to a particular port with the intention of loading containers, it would be better to choose a different port where ships are not arriving in order to load containers.
- *Travel duration* ($t_{op,p}$) Indicator of how much time is needed for the current ship to reach port p from current port (0 if here already, 1 if longest amount possible), calculated as: $t_{op,p} = \frac{d_{op,p}}{\max_{i,j \in P} d_{ij}}$ where $d_{op,p}$ is the amount of time needed to travel from current port op to port p, while $\max_{i,j \in P} d_{ij}$ is the maximum time needed (in hours) to travel between two ports.

3.2.3 Reward function

The objective of the Inland Container Delivery Problem is to choose a ship relocation policy that ensures all containers are delivered on time (or with negligible lateness) while minimizing transportation costs. We seek to prioritize timely delivery of containers, allowing some delays only if there is a significant cost saving. Selecting an appropriate reward function for reinforcement learning is a challenging task, often requiring experimentation and iteration to

determine the most effective one. In our exploration of reward functions, we experimented with several approaches which are further explained in the Supplementary Material 9.1.1. In this work, we found that the following reward function is well-suited for the problem.

$$r_t = \delta * \mathrm{CD}_t - \beta * \mathrm{TC}_t$$

Where CD_t is the number of containers delivered at timestep t that are delivered before their deadline, TC_t is the total transportation cost for ships that arrived at a destination port at timestep t.

At the end of the episode, a further reward is given

$$r_T = \nu * \mathrm{UC}$$

Where UC is the number of still undelivered, late containers at the end of the episode.

The reward function is thus shaped by three hyperparameters: δ, which encourages the timely delivery of containers, β, which penalizes transportation costs, and ν, which penalizes missed or late deliveries.

4 Case study and reinforcement learning agent

In the rest of the paper we focus on a real case study provided by our industrial partner.[3]

4.1 Case study

We consider an inland shipping network with $|P| = 27$ ports and a fleet of $|Z| = 9$ ships. The time horizon for one episode is set to 5 days (or a working week) with $T = 120$ time increments of 1 h. The company has provided historical data of container demand patterns for a 5 week period which we used to model the distribution of the container demand for each episode. Containers come in two standardized sizes: 1-TEU and 2-TEU (where TEU stands for Twenty-feet Equivalent Unit). The majority of the ships in the fleet (seven out of nine) have a capacity of around 260 TEU, with one medium ship with a capacity of 156 TEU and one small ship with a capacity of 24 TEU. Information is available for each shipment request, including quantity (number and weight of containers), origin and destination ports, earliest pick up time and due date.

For each port pair (i, j) where $i, j \in P$ and $i \neq j$ we extracted the following information from the data set: $N_{i,j}$ (number of bulks, or shipment requests, to be delivered), $n_{i,j}$ (average number of containers in a request), $\sigma^n_{i,j}$ (standard deviation of number of containers in a request), $w_{i,j}$ (average container weight), and $\sigma^w_{i,j}$ (standard deviation of weight of containers). We generate episodes by simulating shipment requests using a Poisson distribution with a mean λ^r, which indicates the average number of new bulks that appear every hour. We consider three different values for the arrival rate and analyze the effect of this rate in our numerical experiments.

We approximate the probability that the next bulk will appear in port i with destination port j as: $P(i, j) = \frac{N_{i,j}}{\sum_{k,l \in \mathcal{P}} N_{k,l}}$. The pattern of containers appearing in ports has a great influence on the learning of the agent, as well as the performance of some heuristics. Out of the 702 possible (i, j) combinations of valid routes, only 115 have been represented in the data.

[3] Name of the company is disclosed due to confidentiality reasons.

Each shipping request (bulk) b with origin i and destination j is generated using attributes extracted from the data. The number of containers per shipment follows a normal distribution with mean $n_{i,j}$ and standard deviation $\sigma^n_{i,j}$, and each container has a weight that follows a normal distribution with mean $w_{i,j}$ and standard deviation $\sigma^w_{i,j}$. Moreover, the size of all containers in a shipment is generated with equal probability of being small or large. Using this information, b_{wc} and b_{nc} can be calculated (the reader may refer to Sect. 3.1 for the problem notation). For a particular shipment request, the arrival time, b_{ao}, is generated from a uniform distribution between 0 and 4 days in advance, while the due date, b_{dd}, is generated from a uniform distribution between 1 and 2 days after their arrival. This is based on the real-life pattern where containers typically need to be delivered within a few days of their arrival in the port. We continue this process until we reach the desired number of containers for the (confirmed) demand in the system, NC. Once the initial demand is generated, we start simulating the episode, continually generating new shipment requests throughout its duration.

We have identified nine evaluation scenarios, derived from the current case study, for the purpose of experimental evaluation. Each scenario is defined by the number of containers at the beginning of each evaluation episode (confirmed demand $NC = \{1000, 3000, 5000\}$) and the rate of arrival of new containers ($\lambda^r = \{0.5, 1, 2\}$). Scenarios with 1000 containers are referred to as *low demand* scenarios, with 3000 as *medium demand* scenarios and 5000 as *high demand* scenarios. For each scenario, 20 independent episodes are simulated and average performance indicators are recorded.

4.2 Baseline heuristic approach

Before presenting the reinforcement learning solution for the problem, we first outline a set of baseline approaches based on hand-crafted heuristics. These heuristics are derived from the MDP defined in Sect. 3. Thus, the heuristics provide the rules that guide the ship's choice of the next action. To ensure fairness in the comparison, we maintain the assumptions used for the implementation of the reinforcement learning agent, including the action space. For instance, rerouting during the trip is not allowed, and the same prioritization rules are used in assigning loads to the ship when making the routing decision.

The heuristics are defined as follows (flowcharts are available in the supplementary materials):

- *Random Policy*: The random policy selects each possible action with an equal probability. This heuristic is used to identify a potential low level of performance.
- *Urgency Heuristic*: This rule prioritizes the shipments with the earliest due dates. If a ship has containers on board, it will pick the one with the earliest due date and relocate to its destination port. To be precise, using state representation, it will relocate to $p' = \arg\max_p ud_p$. If that is not the case, it will check if there are containers available to pick up in other ports. If there are, the ship will relocate to the port of the most urgent one. Using state representation, it will relocate to $p' = \arg\max_p u_p$.
- *Capacity heuristic*: This rule prioritizes larger bulks when considering which ones to deliver and/or to load first based on their quantities. If the ship has containers on board, it will go to the port in which it can unload most of its capacity. Using state representation, it will relocate to $p' = \arg\max_p c_p$. If not, it will determine if there is any port which has enough containers available to be picked up which would fill its capacity the most. Using state representation, it will relocate to $p' = \arg\max_p w_p$.

- *Closeness heuristic*: This rule gives precedence to the containers for which the least amount of travel time is needed. If the ship currently has containers on board, it will relocate to the nearest port. Using state representation, it will evaluate $t_{op,p}$ for all $p \in P$ for which $c_p > 0$ and relocate to $p' = \arg\min_p t_{op,p} \text{s.t.} c_p > 0$. If that is not the case, it will consider all ports which have containers ready to be picked up and relocate to the closest one. Using the state representation, it will check $t_{op,p}$ for all $p \in P$ for which $w_p > 0$ and relocate to $p' = \arg\min_p t_{op,p}$ s.t. $w_p > 0$.
- *Hybrid Heuristic*: Finally, we combine all heuristics with a single meta-rule designed as follows. If there are urgent containers on board (due date less than *maxdd=10h*), we use the *urgency heuristic*. Otherwise, we start preparing for approaching deadlines by freeing up capacity on the ships which are heavily utilized, so that they can pick up more containers later, or go to a port where many containers are available to pick up. Therefore, if there are no urgent containers, the *capacity heuristic* is used. Finally, if there are no urgent containers and the ships are not overloaded (capacity greater than *mincap=60%*), the closeness heuristic is used.

4.3 Reinforcement learning agents

Reinforcement learning agents are trained using Proximal Policy Optimization (PPO) (Schulman et al., 2017), with a single shared policy that selects actions for each ship separately. The policy and value functions are modelled as fully connected neural networks with three shared layers of 128, 64, and 32 units, and separate output heads. The hyperparameters used are reported in the supplementary materials (please see Table 3).

For the state representation, we use $d_{\text{threshold}} = 48$ (corresponding to deliveries being non-urgent if their due date is farther than 2 days ahead), $\rho = 600$, and $\max_{i,j \in P} d_{ij} = 3.92$ (the maximum time required to travel between any two ports, in hours).

We compare four different approaches for training the reinforcement learning agents:

- *RL Agent 0: Multi-ship Training*. We consider an environment with all 9 ships where a centralized agent decides the action for whichever ship is currently in a port at a given timestep, and thus has an action to make. This agent is trained for 8 million time steps using a simple reward function which assigns a positive reward of 1 for each container delivered before its deadline. We do not include penalties for transportation costs. This corresponds to reward function parameters of $\delta = 1$, $\beta = 0$, $\nu = 0$ (see Sect. 3.2.3 for the reward function).
- *RL Agent 1: Single-ship Training (Responsive)*. A reinforcement learning agent is trained in an environment with a *single* ship, whose capacity is 256 standard containers. For this setting, the demand is lowered accordingly, and only 500 containers are used as initial demand. Note that evaluation is still performed in an environment with all 9 ships, each controlled with the same trained policy. One benefit of this approach is that the problem is simpler and requires less time to train. The goal in this case is to find a policy with a high quality of service (lower percentage of late shipments and lower average lateness). We adopt the reward function with the following parameters $\delta = 1$, $\beta = 0$, $\nu = 10$.
- *RL Agent 2a: Single-ship Training (Efficient and responsive)*. Reinforcement learning agent is trained similarly to RL agent 1. However, we seek to train the agent that is more cost efficient than RL agent 1. For this purpose, we also set the reward function with the following parameters $\delta = 1$, $\beta = \frac{1}{10}$, $\nu = 10$.
- *RL Agent 2b: Single-ship Training (Efficient and responsive with initialization)*. A reinforcement learning agent is trained similarly to RL agent 1, with two differences. First,

the agent is fine-tuned by training in a similar environment, but with a different reward function. The reward function parameters are $\delta = 10$, $\beta = \frac{1}{10}$, $\nu = 10$. Second, the neural networks for the policy and value functions are pre-initialized with the weights of the trained RL agent 1.

5 Results

We compare the performance of the reinforcement learning agents against the heuristics by identifying two metrics that capture different qualities of the agent's behaviors:late containers, and cost.

Late containers is the percentage of containers that are delivered late compared to total number of containers in the system. A late container is a container that is either delivered after its due date, remained on board while its due date expired, or has never been picked up from the port before its due date passed. The metric *cost* is the total cost of logistics incurred over several components such as loading cost, unloading cost, waiting cost, and other expenses. Since the biggest component is the transportation cost, we approximate the total cost as the amount of time ships have spent travelling within the episode. We also included the *total containers* information in our results which indicates the average number of containers that appeared across the 20 evaluation episodes. This includes containers that haven't arrived yet in their arrival ports by the end of the episode, but whose imminent arrival is expected.

Table 2 shows a summary of the results for the suggested heuristics and the RL agents. Full results are presented in the supplementary materials, Section 9.2. In the supplementary materials, we also provide a comprehensive overview of the performance of different agents (grouped together based on performance indicator) across various scenarios at different values of λ^r.

As expected, the *random policy* exhibits poor performance, characterized by a large incidence of late containers and high costs. However, it sets a baseline performance level for comparison. The hybrid heuristic does well with a low -but non zero- proportion of late/undelivered containers, and with a low cost. The urgency heuristic has a tradeoff between late containers and cost, close to that of the random policy, which suggests a sub-optimal strategy. We have opted to exclude the results of the Capacity and Closeness Heuristics from our analysis due to their consistent underperformance when compared to the Hybrid Policy. Across various scenarios and values of λ^r, both Capacity and Closeness Heuristics exhibited a higher percentage of late containers and increased total costs.

RL agent 0, which was trained in a multi-agent environment with all ships present, has learnt a sub-optimal policy. This is evident from metrics that are better than a random policy, yet considerably worse than all other policies. RL agent 1, controlling all nine ships with a policy trained in an environment with only one ship, successfully delivers all loads within the due time, but at a high transportation cost comparable to the random policy. This is not surprising since the reward function did not penalize transportation costs ($\beta = 0$). RL agent 2a is trained in a similar way to RL agent 1, but with a penalty for transportation cost. This results in a drastic decrease in cost, albeit at the expense of a small increase in the proportion of late deliveries. In the case of RL agent 2b, we initialize the neural networks for the policy and value functions using the weights of the trained RL agent 1. Consequently, RL agent 2b excels in both cost efficiency and responsiveness when compared to RL agent 2a. Henceforth, for simplicity, we refer to RL agent 2b as RL agent 2.

Table 2 Summary of the performance of heuristics and the RL agents used in the study across 9 scenarios (3 cases with different initial demand with 3 different rates of arrival of subsequent delivery requests)

No. of containers (NC)	1000			3000			5000		
Rate of arrival (λ^r)	0.5	1	2	0.5	1	2	0.5	1	2
Random policy									
Total containers	1676.05	2349.15	3558.60	3641.30	4174.65	5462.70	5559.45	6086.90	7413.65
Late containers (%)	30.35	27.54	25.74	45.11	39.55	37.11	53.98	50.69	49.17
Cost	61818.64	61459.92	60876.58	59972.63	59748.77	59045.79	58173.69	58127.48	57674.15
Urgency heuristic									
Total containers	1772.50	2338.30	3748.65	3665.10	4480.90	5848.80	5750.10	6436.35	7716.95
Late containers (%)	0	0	0	0	0	0.20	0	0.48	0.02
Cost	51999.36	56805.22	56776.41	55718.72	56938.38	56063.33	54235.00	55217.46	53897.04
Capacity heuristic									
Total containers	1714.80	2352.55	3752.00	3681.35	4429.75	5840.30	5703.15	6477.35	7934.15
Late containers (%)	0.01	0.01	0.18	0.51	0.71	1.08	1.87	2.13	3.54
Cost	51094.07	54340.63	56693.46	56110.46	57643.35	56727.57	54893.24	55966.92	54562.33
Closeness heuristic									
Total containers	1804.50	2591.20	4032.35	3853.70	4729.55	6419.10	5968.25	6983.30	8470.05
Late containers (%)	0.35	0.07	0.29	0.78	1.48	1.17	2.88	2.74	2.31
Cost	50747.99	54263.94	58252.61	56004.65	58280.33	59461.54	55340.98	57290.51	57789.18

Table 2 continued

No. of containers (NC)	1000			3000			5000		
Rate of arrival (λ^r)	0.5	1	2	0.5	1	2	0.5	1	2
Hybrid heuristic									
Total containers	1753.50	2438.25	3852.75	3683.35	4409.10	5741.30	5700.25	6388.65	7741.00
Late containers (%)	0.00	0.11	0.69	0.04	0.44	0.23	1.01	0.83	1.84
Cost	25930.31	27851.84	30149.48	34210.61	35400.94	36312.66	38846.78	39526.39	40250.03
RL agent 0									
Total containers	1651.10	2367.90	3715.30	3676.15	4341.80	5893.10	5699.45	6417.95	7817.90
Late (%)	15.71	12.71	9.89	14.78	13.65	11.16	17.77	15.48	15.64
Cost	54397.09	55920.37	55275.60	53681.73	52710.65	54206.54	51296.08	50582.28	50593.56
RL agent 1									
Total containers	1694.20	2423.20	3812.80	3752.80	4451.40	6206.60	5834.40	6578.40	7998.80
Late containers (%)	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.02
Cost	63161.34	62888.31	62098.69	60995.25	60565.82	59686.86	58964.31	58510.18	57485.51
RL agent 2a									
Total containers	1691.20	2410.35	3707.30	3706.40	4486.45	5778.55	5734.70	6479.20	7879.30
Late containers (%)	1.62	0.54	0.11	0.39	0.08	0.05	0.93	0.57	0.41
Cost	31291.02	37078.89	43179.48	45133.94	46917.60	50626.18	49217.97	49880.64	50099.40
RL agent 2b									
Total containers	1692.05	2390.60	3956.95	3667.10	4471.70	5833.60	5780.20	6405.90	7857.25
Late containers (%)	0.84	0.63	0.25	0.17	0.01	0.07	0.12	0.10	0.15
Cost	14323.58	17512.89	23064.90	28233.53	30435.92	34727.54	37953.09	39169.16	42792.09

Performance is measured by tracking the proportion of late or undelivered containers (*late containers*) and the total delivery cost (*cost*), averaged over 20 simulations

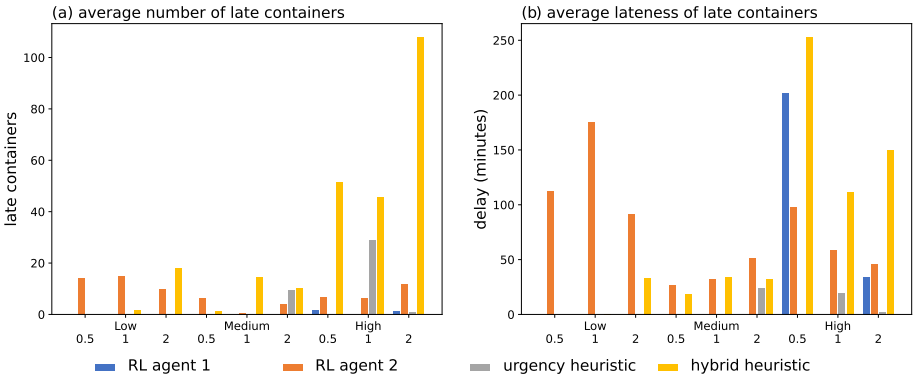

Fig. 1 Comparison of the responsiveness performance of the RL agents versus the heuristics. Responsiveness is measured by the number of late containers delivered (**a**), and by their average delay (**b**). Results for each model are averaged over 20 randomized episodes

In summary, RL agent 2 was found to be the best performing agent, demonstrating lower costs and minimal late deliveries. On the other hand. RL agent 1 still achieves competitive responsiveness results compared to RL agent 2. Therefore, subsequent subsections provide a comprehensive analysis, focusing on RL agents 1 and 2.

We provide our analysis in four categories: responsiveness (Sect. 5.1), cost efficiency (Sect. 5.2), capacity utilization (Sect. 5.3) and relocation (Sect. 5.4). The policies are lastly evaluated in a scenario where the underlying statistics differ from the training ones, by generating new delivery requests with different demand distributions over a fixed set of routes (Sect. 5.5).

5.1 Responsiveness

Figure 1a displays the average number of late containers for all nine scenarios and the policies under consideration. RL agent 1 only has late containers in the high demand scenarios, and the average number is below five containers per episode. The urgency heuristic rarely delivers late containers, but when it does, the average number is higher than RL agent 1. RL agent 2 exhibits the least satisfactory performance in the low demand scenarios, but better in medium and high demand scenarios. However, the average number of late containers is relatively low, never exceeding 20. The hybrid heuristic shows favorable performance in the low and medium demand scenarios, however its performance deteriorates in high demand scenarios, potentially exposing decision makers to numerous late deliveries.

We further look at the average lateness of deliveries (presented in Fig. 1b) by measuring the average amount of time which has passed after the due date until each late container is delivered. The urgency heuristic has the lowest mean lateness, while the hybrid heuristic also has low average lateness for low and medium demand scenarios. On the other hand, RL agents have higher mean lateness values. Although RL agent 1 rarely delivers containers late, when it does, they can be late by a couple of hours. Applying a penalty at the end of an episode for late containers has helped to prevent high lateness. To further reduce lateness, we can engineer a reward system that provides negative rewards proportional to the lateness of late containers (as opposed to the current approach of penalizing all late containers equally).

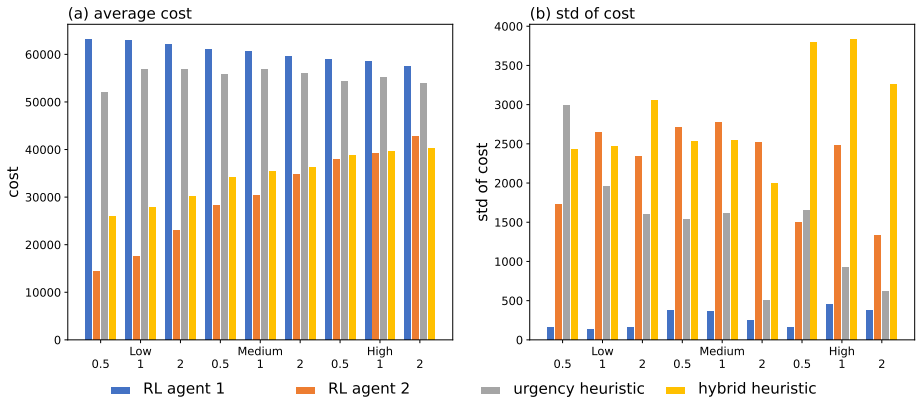

Fig. 2 Comparison of the cost efficiency of the RL agents versus the heuristics. Cost efficiency is measured by the total delivery costs incurred by the agents within evaluation episodes. Results for each model are averaged over 20 randomized episodes

5.2 Cost efficiency

The overall transportation cost is a key metric we seek to optimize. As displayed in Fig. 2a, the RL agent 1 has the highest average cost in all scenarios, which decreases as the number of containers in the system increases. This is due to the lack of negative rewards for transportation cost in the RL agent 1's training phase. The increasing number of containers also increases the loading time and reduces the available time for travel, thus leading to a higher transportation cost.

The urgency heuristic, which is the second most expensive policy after RL agent 1, has the least cost in the lowest demand scenario, while similar costs in other scenarios. In contrast, RL agent 2 is the most cost-effective policy, especially in low demand scenarios. However, the cost of RL agent 2 increases with the demand, as the number of travels required to deliver containers also increases. The hybrid heuristic is cheaper than RL agent 1 and the urgency heuristic, but more expensive than RL agent 2, except in the highest demand scenario. Nevertheless, the hybrid heuristic has a higher number of late deliveries than RL agent 2 in the high demand scenarios.

The standard deviation of costs for the different RL agents and the hybrid heuristic are presented in Fig. 2b. RL agent 1 has the lowest variance in cost across simulated episodes, largely due to its tendency to spend all available time travelling. The urgency heuristic has a relatively high standard deviation in the lowest demand scenario, but this variance decreases as the demand increases. RL agent 2 and the hybrid heuristic show a similar degree of cost variation in the low and medium demand scenarios, but in high demand scenarios the hybrid heuristic's deviation of costs increases (and performance decreases) while RL agent 2's decreases.

5.2.1 Efficient frontier

In Fig. 3, we compare the cost and responsiveness of the policies across three scenarios with different levels of demand ($\lambda^r = 1$). In the low demand scenario, the trade-off between cost and quality of service is most apparent: RL agent 2 is the cheapest policy but results in the most late containers, while the hybrid heuristic is slightly more expensive but delivers fewer

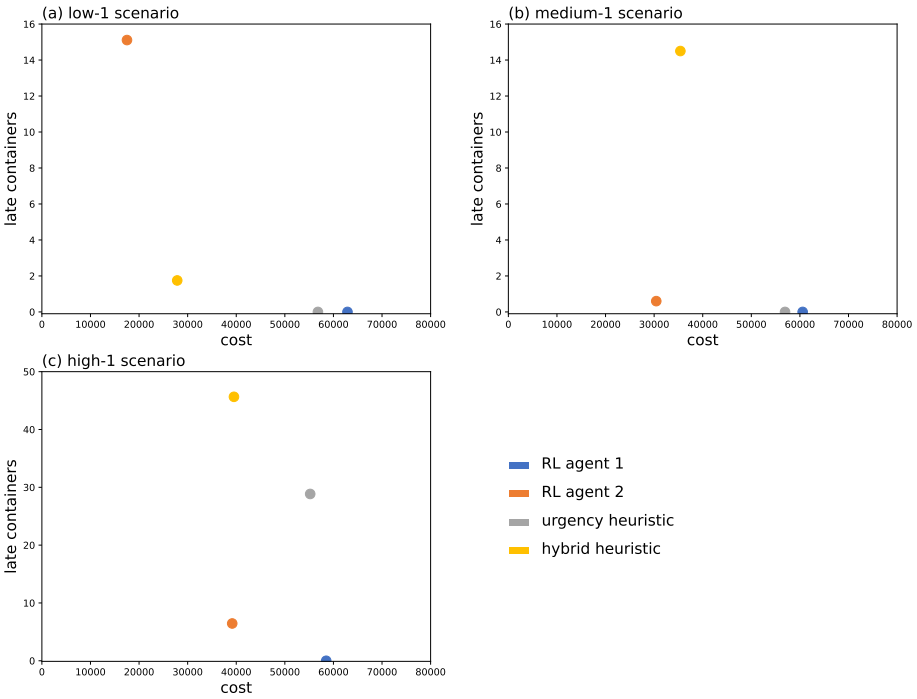

Fig. 3 Comparison of the efficient frontier of the RL agents versus the heuristics

containers late. Both the urgency heuristic and RL agent 1 have perfect quality of service, but at a higher cost.

In the medium demand scenario, the RL agents and the urgency heuristic have very good quality of service, with RL agent 2 being by far the cheapest. The hybrid heuristic has more late containers, but at a cost lower than the other two policies.

In the high demand scenario, the quality of service of the heuristics deteriorates while that of the RL agents remains satisfactory. While the hybrid heuristic has a comparable cost to RL agent 2, its inferior quality of service makes RL agent 2 the most effective policy overall.

5.3 Capacity utilization

Barge operators prioritize high capacity utilization as it indicates efficient use of ships. This is calculated as the percentage of ship's capacity that is occupied by containers on board, averaged over the length of an episode for all ships, as seen in Fig. 4. In practice, it is important to maintain a balance between high utilization and other factors such as responsiveness and cost efficiency.

The random policy has a utilization of 83% in the highest demand scenario, which is significantly higher than any of the other four policies presented (please see Supplementary Material Table 4). This is due to the fact that this policy rarely delivers containers, leading to a large number of late containers being retained on board.

In the RL framework, there are no explicit rewards for high capacity utilization. Despite this, RL agents exhibit some of the highest levels of capacity utilization among the policies

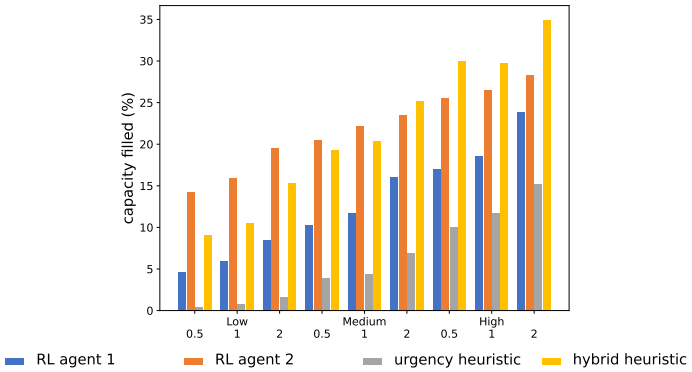

Fig. 4 Comparison of the average ship capacity filled (%)

that we tested. The urgency heuristic, on the other hand, demonstrates the lowest utilization of ships due to its greedy approach of loading and delivering containers with the most pressing due dates.

In our study, average shipment size is 8.29 containers, which means around 12.4 TEU which is approximately only 6% of the total capacity of most ships in this problem which have capacity of 208 TEU. This data and our results are generally are inline with the practice where 30% to 40% utilization is observed. To investigate the implications of fleet reduction, in the next section, we incorporate additional numerical tests with reduced fleet size. By gradually reducing the fleet size, our objective is to examine how it impacts the utilization of capacity, overall costs, and the late deliveries.

5.3.1 Effect of reduced capacity

We studied the performance of RL agent 1 and RL agent 2, urgency and hybrid heuristic in the context of reduced fleet sizes. Initially, our fleet consisted of 9 ships, jointly offering a total TEU capacity of 1984. Subsequently, we investigated scenarios with fleet sizes of 6 ships (with a TEU capacity of 1308), 7 ships (TEU capacity 1464), and 8 ships (equating to a TEU capacity of 1724). The outcomes are shown in Fig. 5, and reported in full in the supplementary material (please see Section 9.2).

One of the key observations from the computational results is that, as the fleet size diminishes, there is a noticeable increase in the number of containers arriving late, which aligns with our initial expectations. Simultaneously, we observe a proportionate surge in the utilization of the available capacity, indicating a more efficient deployment of the ships. Furthermore, corresponding reduction in operational costs becomes evident as the fleet size decreases, suggesting potential cost-saving advantages associated with a smaller fleet. This reduction in transportation costs is primarily attributable to decreased travel time for all ships. Additionally, the company benefits from cost savings related to lower fixed expenses, as this may lead to renting out fewer ships or avoiding additional rentals.

5.4 Relocations

Figure 6a shows that RL agent 1 has the highest number of relocations across all demand scenarios. However, as the demand increases, the number of relocations decreases. This is

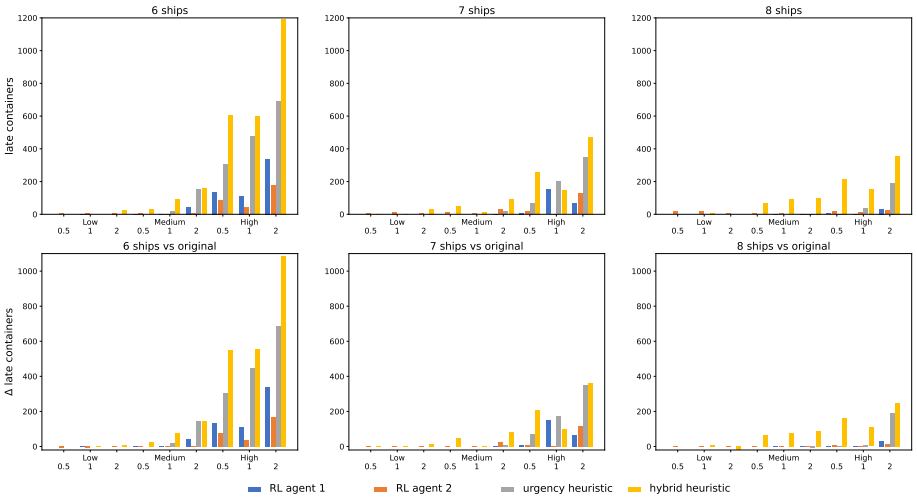

Fig. 5 Comparison of number of late containers in alternative scenarios with a reduced fleet size of 6, 7, and 8 ships (top row), together with the difference in number of late containers with respect to the corresponding policies in the default 9 ships case

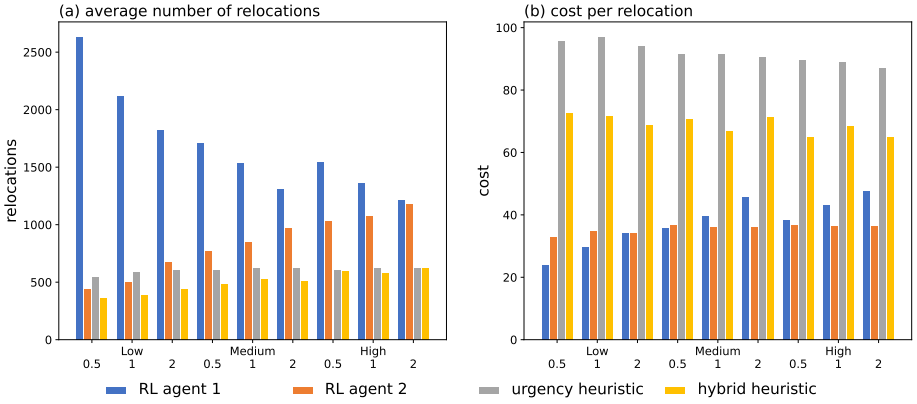

Fig. 6 Comparison of relocation metrics of the RL agents versus the heuristics. We count the number of ship relocations within evaluation episodes and the average cost per ship relocation. Results for each model are averaged over 20 randomized episodes

due to the fact that ships spend more time loading containers, leaving less time for relocations and travelling. The urgency heuristic has a relatively constant number of relocations while the hybrid heuristic has around 400 relocations in low demand episodes, and 600 in high demand episodes. The positive correlation between demand level and number of relocations becomes most apparent in RL agent 2. This is in line with the current man-made schedules, where a low number of orders doesn't require ships to change port as many times as in the case of high demand.

It is also interesting to look at the average cost per relocation (Fig. 6b). Since cost is a proxy for travel time in our framework, this metric directly reflects average route distances. The urgency heuristic has an average cost per relocation of around 90, which means that the routes that ships decide to take are on average 90 min long. The hybrid heuristic has a

somewhat lower value, at around 70, which is still significantly higher than the RL agents. We deduce that RL agents have identified routes that are efficient in terms of both vehicle capacity utilization and travel time. This allows the agents to visit more ports in less time, which allows them to have more opportunities to deliver and load containers. In comparison to the manual heuristics, the RL agents showed a tendency to select shorter routes. When traveling to a distant port, this strategy is reflected in a non-direct routing strategy that enables ships to make multiple stops along the way, providing additional opportunities for loading and unloading containers.

5.5 Alternative demand patterns

To evaluate the robustness of RL agents and the heuristic policies when the environment statistics change, we designed alternative environments in which the demand patterns differ from the original setting. Thus, the RL agents trained on simulated environments reflecting the real-world demand pattern are tested in an environment with different statistics. Comparing their performance in the new environment to the original environment provides an indication of the generalization capabilities of the learnt policies versus over fitting to the demand pattern of the training environment.

From the data provided by our partner, we observed 115 source-destination port combinations, (p_i, p_j), from which demand arose. We initially assess the scenario with demand uniformly distributed across the routes. Subsequently, we analyze settings where demand surges are concentrated at specific ports.

5.5.1 Uniform demand

For the uniform-demand experiments, we replaced the probabilities used to generate new loads with a uniform distribution across these routes, and a zero probability for all other routes. Moreover, each bulk had a random number of containers, drawn from an uniform distribution between 1 and 15. This means that all routes in the original dataset would be *equally busy*, with the same expected amount of containers to be delivered.

Figure 7 shows the average number of late containers for the four best policies across all demand levels. RL agent 1 delivers all containers on time in low and medium demand scenarios. However, in the high demand setting it has an average of around 30 late containers per episode. RL agent 2 consistently provides a high quality of service in all scenarios; the number of late containers never exceeds 20. The hybrid heuristic has a low number of late containers in the low demand scenarios, but when the demand increases, the performance deteriorates; the average number of late containers in high demand scenarios is around 140. The urgency heuristic, like RL agent 1, has perfect quality of service in low and medium demand scenarios. When the number of containers becomes sufficiently high, more containers are delivered late.

Figure 7b shows that the average cost of each policy across different scenarios follows similar patterns to the original demand setting. However, the number of late containers is higher in the new environment for most policies, as indicated by the absolute difference in the mean number of late containers between the uniform demand setting and the original demand (Fig. 7c). Notably, the performance of RL agent 2 remains unchanged, indicating its robustness and the ability to generalize to novel environments.

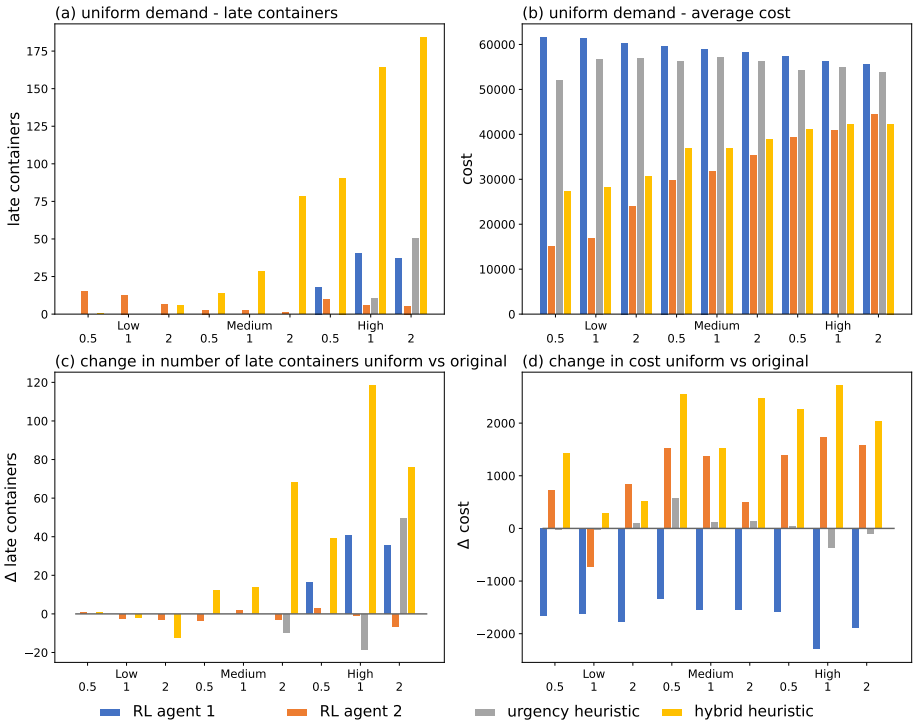

Fig. 7 The robustness of the agents is evaluated by changing the statistics of the original environment to a simplified uniform demand scenario. All agents are evaluated in the new environment, tracking the number of late containers (**a**) and average cost (**b**). Both metrics are compared to the performance of each agent in the original environment (**c**, **d**)

Compared to the original environment, we observe that the cost of the hybrid heuristic and RL agent 2 increases, though marginally. The cost of the urgency heuristic remains unchanged, while that of RL agent 1 decreases by around 2.5% (Fig. 7d).

5.5.2 Demand surges

In the previous section we tested our solution approaches when demand is uniformly distributed. It is also essential to examine the impact of demand surges for containers within a port, as these occurrences are frequently observed in real-world scenarios. Our objective is to gain insights into the dynamics of heightened container demand activity in a port and how our solution approaches respond to that.

We next focus on modeling such demand surges at a designated busy port exploring two distinct approaches. For the first approach (demand spike 1), we include the demand surge considering the initial stage of the episode. For this purpose, at the beginning of an episode, we allocate a certain proportion of containers (30%) to originate from a busy port. For the second approach (demand spike 2), we introduce demand surges during the course of an episode. In this case, we introduce containers unevenly, with 50% of them arriving at the busy port at different intervals. The results from the two approaches are shown in Fig. 8.

According to 8, in the first scenario; the performance of both RL agents deteriorates compared to the original demand setting. This effect is more evident for RL agent 1, which

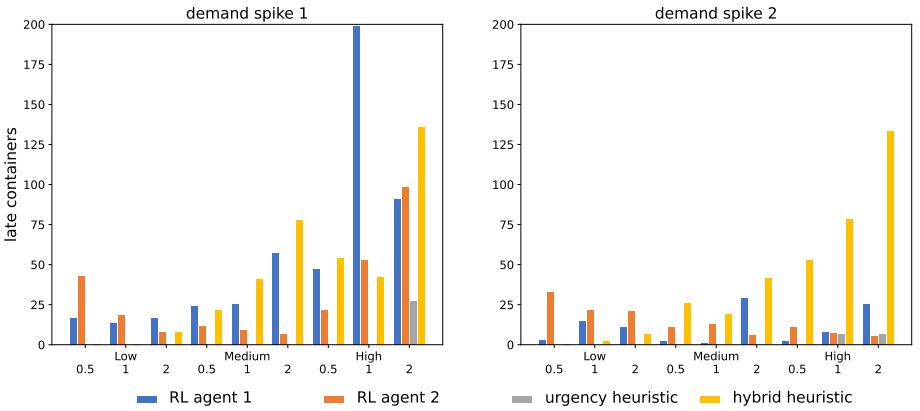

Fig. 8 Comparison of number of late containers in two scenarios involving demand surges at a designated busy port. In the first scenario (left), 30% of the containers are generated at a single busy port at the beginning of an evaluation episode. In the second scenario (right), 50% of all containers originate from a single busy port, but their arrival is spread out throughout the evaluation episode

typically ensures the timely delivery of all containers. The urgency heuristic retains its good performance and proves to be the most robust in this scenario, at least in terms of late containers. The hybrid heuristic exhibits a similar behavior as in the original scenarios, with its performance deteriorating significantly as the number of containers in the system increases.

For the second scenario, both RL agents display improved adaptability and robustness, with a marginal increase in late containers compared to the original demand scenario. The urgency heuristic maintains its high performance, similar to the first scenario. The hybrid heuristic performs comparably or slightly better than in the original demand scenarios when the container count is low. However, as the container count rises, its performance deteriorates more significantly than in the original scenario.

In the demand spike scenarios, both RL agents' and hybrid heuristics' performances deteriorate, resulting in an increased number of late containers, while the urgency heuristic consistently performs well in terms of minimizing lateness. This is intuitive as this heuristic is designed for this purpose. Generally, the performance of RL agents and heuristics is superior in demand spike 2 compared to demand spike 1. The reason for this could be that the uneven distribution of containers that occurs in demand spike 1 is apparent at the beginning of the episode, and the proper reaction should be to send as many ships as needed for that port so that it can pick up all those containers. Failure to do so leads to a surge in late containers. Agents may lack adaptability as they have probably never encountered a similar situation in training. In contrast, demand spike 2 features a dynamic, gradual bias of containers toward a single busy port during the episode. Agents may gain experience from training in handling analogous scenarios. Another explanation is that demand spike 1 is inherently more difficult to solve than demand spike 2.

5.6 Insights and trade-offs from the industry perspective

Although we evaluated different objectives in this study, the company's primary objective is to ensure the punctual delivery of containers, as deviations from this commitment may result

in client loss to competing transportation providers. However, it is crucial to acknowledge the complex and unpredictable nature of the inland shipping industry. Factors such as terminal delays upon a ship's arrival or the occasional inaccessibility of specific waterways due to low water levels introduce uncertainty to this problem. Consequently, even with the most carefully crafted schedules, there may be cases of delayed container deliveries. Therefore, a marginal proportion of late containers is regarded as acceptable and comprehensible.

While maintaining the delivery performance, fleet managers are focused on optimizing the efficiency of their ships to meet demand within prescribed time-frames while reducing resource expenditures, particularly in terms of costs. Achieving this balance could involve strategies that reduce the overall number of ships deployed, limit the frequency of stops and relocation to mitigate terminal expenses, and decrease the overall travel distance to minimize fuel expenditure.

In this study, we aim to reconcile the multiple goals of cost-efficiency and timely service delivery, taking into account the intricate dynamics of the industry and the necessity to adapt to its complexities and uncertainties. Based on the priority of these goals, reward models can be adjusted accordingly.

6 Conclusion

The container delivery problem belongs to the category of dynamic fleet assignment problems. Barge operators, who manage a fleet of ships, must determine on a route for each ship and the containers it should pick up along the way. Currently, this scheduling process is done by experienced schedulers, with the goal of delivering the containers before their expiration date at the lowest cost. We use reinforcement learning to create a system that can use historical data to make more efficient scheduling decisions. This could reduce the dependence on human schedules and enhance the performance of the supply chain.

To train our reinforcement agents, we have utilized the policy gradient method known as proximal policy optimization (PPO). To compare the performances of RL agents trained in different ways, several heuristic policies have been constructed using the same framework and information as the RL agents.

It is challenging to devise a flexible and robust heuristic which performs well on a multitude of diverse instances. Heuristics are typically tailored to perform optimally in a specific type of environment and, as the dynamics change, their performance likewise suffers.

We found that agents trained exclusively in a single-ship environment were the most successful even when applied to the full fleet environment. This is likely due to the reduced computational power and training time required. With the availability of more computing power, it is likely that even higher performance can be achieved when training directly in the full fleet environment, provided that sufficient training time is given.

Exploring alternative hyperparameter settings, RL algorithms, or reward structures could potentially lead to enhanced results. For instance, the reward function could be modified to emphasize objectives such as minimizing average lateness, location frequency, or maximizing capacity utilization explicitly. To address the complexity of the problem, one could develop an alternative environment in which a relocating policy is provided and the ships are trained to learn a loading policy for the containers. This can be followed by training the relocating policy in a similar environment until the policies converge. Further research may also introduce greater uncertainty to the environment to better reflect real-life dynamics, such as inland water levels, gas prices, and travel durations due to port congestions.

Reinforcement learning has proven to be an effective and reliable approach for the container delivery problem, allowing the agent to automatically adjust to unpredictable conditions. In practical settings, our method can be utilized within a rolling horizon framework, such as formulating the schedule for the upcoming week while considering a fixed period (e.g., one day). This gives us assurance that the methodology can be applied to various complex dynamic logistics scheduling scenarios in the future, involving dynamic demand and supply matching along with routing decisions.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10479-024-05927-4.

Acknowledgements We would like to thank Quirijn Schevenhoven who played a key role in the initial stages of this project and we are grateful for their contributions. We would also like to thank our *industrial partner* for communicating us the problem and providing access to data.

Funding The authors did not receive support from any organization for the submitted work.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose. All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Cheung, R. K., & Hang, Darren D. (2003). A time-window sliding procedure for driver-task assignment with random service times. *IIE Transactions, 35*(5), 433–444.

Cheung, R. K., Hang, D. D., & Ning, S. (2005). A labeling method for dynamic driver-task assignment with uncertain task durations. *Operations Research Letters, 33*(4), 411–420.

Daham, H., Yang, X., & Michaela, W. (2016). An efficient mixed integer programming model for pairing containers in inland transportation based on the assignment of orders. *Journal of the Operational Research Society, 68*, 12.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*.

Hyland, M., & Mahmassani, H. (2018). Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests. *Transportation Research Part C Emerging Technologies, 92*, 278–297.

Inland waterways. Retrieved 30 Nov 2022 from https://transport.ec.europa.eu/transport-modes/inland-waterways_en.

Li, K., Zhang, T., Wang, R., Wang, Y., Han, Y., & Wang, L. (2021). Deep reinforcement learning for combinatorial optimization: Covering salesman problems. *IEEE Transactions on Cybernetics, 52*(12), 13142–13155.

Lin, K, Zhao, R, Xu, Z., & Zhou, J. (2018). Efficient large-scale fleet management via multi-agent deep reinforcement learning. KDD '18, (pp. 1774–1783) Association for Computing Machinery.

Mahmassani, H., Kim, Y., & Jaillet, P. (2000). Local optimization approaches to solve dynamic commercial fleet management problems. *Transportation Research Record Journal of the Transportation Research Board, 1733*, 71–79.

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

Min, D., & Kang, Y. (2021). A learning-based approach for dynamic freight brokerages with transfer and territory-based assignment. *Computers & Industrial Engineering, 153*, 107042.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A., et al. (2021). A graph placement methodology for fast chip design. *Nature, 594*(7862), 207–212.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, (pp. 1928–1937). PMLR.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.

Nazari, M., Oroojlooy, A., Snyder, L., & Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. *Advances in Neural Information Processing Systems*, *31*.

Oda, T., & Joe-Wong, C. (2018). MOVI: A model-free approach to dynamic fleet management. CoRR, arXiv:1804.04758.

POR. optimising-inland container shipping. Retrieved 30 Nov 2022 from https://www.portofrotterdam.com/en/logistics/connections/intermodal-transportation/inland-shipping/optimising-inland.

Puterman, M. L. (2009). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley Series in Probability and Statistics, Wiley.

Raza, S. M., Sajid, M., & Singh, J. (2022). Vehicle routing problem using reinforcement learning: Recent advancements. In D. Gupta, K. Sambyo, M. Prasad, & S. Agarwal (Eds.), *Advanced Machine Intelligence and Signal Processing* (pp. 269–280). Springer Nature.

Schevenhoven, Q. (2021). *A reinforcement learning and heuristic approach to the shipbroking problem*. Tilburg University.

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

Sun, P., Yuxiang, H., Lan, J., Tian, L., & Chen, M. (2019). Tide: Time-relevant deep reinforcement learning for routing optimization. *Future Generation Computer Systems, 99*, 401–409.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Wang, Q., & Tang, C. (2021). Deep reinforcement learning for transportation network combinatorial optimization: A survey. *Knowledge-Based Systems, 233*, 107526.

Yang, J., Jaillet, P., & Mahmassani, H. (1999). On-line algorithms for truck fleet assignment and scheduling under real-time information. *Transportation Research Record Journal of the Transportation Research Board, 1667*, 107–113.