Dray-Q: Demand-dependent trailer repositioning using deep reinforcement learning

Hadi Aghazadeh a, Yunli Wang b, Sun Sun c, Xin Wang a,∗

a Department of Geomatics Engineering, University of Calgary, Canada
b Digital Technologies Research Centre, National Research Council Canada, Ottawa, Canada
c Digital Technologies Research Centre, National Research Council Canada, Waterloo, Canada

ARTICLE INFO

Keywords:
Deep reinforcement learning
Trailer repositioning
Drayage operations
Pickup and delivery
Just-in-time inventory

ABSTRACT

Trailer repositioning in drayage operations is a crucial element in the efficient transportation of goods between global commerce and local communities. Prior research has made two main assumptions to simplify the problem which are the availability of empty trailers at each yard and predetermined movements for empty trailers. In contrast, this paper proposes a demand-dependent trailer repositioning framework that considers delay penalties and limited available trailers while also addressing ad-hoc changes in orders. To address this problem, we formulate a novel drayage operation pickup and delivery framework (Dray-Q) that optimizes the just-in-time movement of empty trailers in response to the prevailing demand. The main objective is to avoid scenarios where there is either a surplus of empty trailers or a deficit, ensuring that the provisioning of trailers in each yard is both timely and adequate. In order to train the agent, our framework utilizes the advanced Reinforcement Learning algorithm, Rainbow-DQN, to learn efficient the real-time trailer repositioning policy. We introduce a multi-objective reward function that balances empty trailer supply and demand, minimizes delays, and considers customer priorities in the dispatching of empty trailers. Dray-Q is flexible and adaptable to changes in customer order settings and can be scaled to accommodate different combinations of order, trailer, and yard sizes. Experimental results demonstrate that Dray-Q outperforms well-known baseline methods in the literature and can be implemented at a production level with exceptional performance and generalizability.

1. Introduction

Drayage operations connect customers and container ports and provide an effective solution to the first/last-mile delivery in intermodal transportation as intermodal transportation consolidates the economic advantages of rail/vessel transportation with the versatility of trucking. Drayage Operations are the most important aspect of intermodal transportation (Chen et al., 2022). Despite the relatively short distances covered by trucks in drayage operations, the cost associated with these operations which involve the first and last-mile delivery processes can constitute between 25% and 40% of the overall cost of intermodal transportation (Zhang et al., 2020, 2011a). The optimization of drayage operations presents a formidable challenge, given the interplay of resources, time, and external factors. Among the various components of drayage operations, the optimization of trailer repositioning has been identified as a crucial aspect, attracting significant research efforts toward developing robust and practical solutions (Braekers et al., 2011).

∗ Corresponding author.
E-mail addresses: hadi.aghazadeh@ucalgary.ca (H. Aghazadeh), yunli.wang@nrc-cnrc.gc.ca (Y. Wang), sun.sun@nrc-cnrc.gc.ca (S. Sun), xcwang@ucalgary.ca (X. Wang).

https://doi.org/10.1016/j.trc.2024.104641
Received 20 September 2023; Received in revised form 6 March 2024; Accepted 27 April 2024
Available online 20 May 2024
0968-090X/Crown Copyright © 2024 Published by Elsevier Ltd. All rights reserved.
Fig. 1. An example of trailer repositioning in drayage operations. The scenario involves the transportation of two orders, with the trailer initially departing from yard 1 to travel to yard 3 for the first pick-up (time step 1). The next movements are illustrated with dashed lines. After loading the order, the trailer proceeds to yard 2 for unloading (time steps 2,3). Meanwhile, the second order encounters a delay due to the unavailability of an empty trailer for loading. Consequently, the trailer is repositioned in an empty state to yard 4 to collect the delayed order (time steps 4,5). Eventually, the trailer delivers both orders to yard 1, completing the operation (time step 6).

The nature of drayage operations in literature is marked by multiple traveling salesman problems with time windows and resource constraints (Ileri et al., 2006; Zhang et al., 2011b). Of the three primary resources involved in drayage operations, namely trailers, trucks, and drivers, the management of a pool of trailers presents the greatest challenge due to its complexity (Zhang et al., 2020; Braekers et al., 2013). This complexity arises from the consideration of both loaded and empty trailers along with a limited number of trailers in relation to the demand (Braekers et al., 2011). According to a report by Drewry Shipping Consultants of London, 20% of all trailer movements consist of empty movements, which do not generate any revenue (Braekers et al., 2011). However, these empty movements are essential for dispatching loaded trailers efficiently, emphasizing the importance of their presence being just in time from an optimization standpoint. Fig. 1 provides an overview of trailer repositioning in drayage operations.

This simple yet realistic illustration consists of four yards, two customer orders, and one trailer. The trailer’s itinerary is as follows: the trailer departs from yard 1 and travels to yard 3 to pick up the first shipment. The trailer then transports the cargo to yard 2 for unloading. As a result of the unavailability of an empty trailer for loading, the second shipment experiences a delay at time step 3. Following this, the trailer is repositioned to yard 4 in an empty state to collect the second delayed shipment, finally, the trailer delivers the order to yard 1.

This scenario illustrates the pickup and delivery process within drayage operations, where a single trailer undergoes various state transitions across different time steps to fulfill customer demands. Furthermore, the challenge of trailer repositioning entails three main obstacles:

1. The empty trailers are not constantly accessible in every yard. In recent literature, optimization models have emerged integrating both loaded and empty trailers. However, many of these models simplify by assuming that empty trailers are consistently accessible at every yard, a condition that may not align with practical scenarios (Ileri et al., 2006; Zhang et al., 2011a). The heightened complexity stemming from restricted trailer availability mandates monitoring state transitions of trailers, transitioning between loaded and empty states at different time intervals, rather than treating each trailer as a singular entity with a fixed state across all time steps.

2. The empty trailer movements are not distinct, pre-established orders originating from the customer’s side, but they depend on the flow of customer demand. Certain models, aiming to alleviate the previous assumption, treat empty trailers as separate demands with prearranged movements (Zhang et al., 2020). While this strategy addresses the challenge of insufficient empty trailers across yards and streamlines trailer movement, it could lead to an uneven distribution of trailers among yards. Additionally, in many instances, customers are unable to forecast their empty trailer requirements promptly, leading to operational disruptions in the pickup and delivery process.

3. There is a need for real-time handling of ad-hoc changes and unplanned orders. The occurrence of spontaneous changes in order specifications, such as time windows or cancellations, poses a significant challenge in drayage operations (Zhang et al., 2011a). It is noted that only 60% of drayage requests are known prior to the day of operation, with the remaining requests emerging spontaneously (Zhang et al., 2011a). Additionally, unforeseen events like breakdowns, adverse weather, or customer adjustments may necessitate changes to the dispatching plan. These challenges underscore the need for an
approach capable of handling ad-hoc demands in real-time. Drayage operations are also susceptible to delays due to various factors, including weather conditions and breakdowns, affecting travel time. Moreover, internal factors such as empty trailer availability, influenced by planning decisions, contribute to operational risks. While maintaining a surplus of empty trailers reduces this risk, it entails extra costs. Alternatively, efficient planning offers a more cost-effective and robust solution.

Among the mentioned challenges, there is a lack of research exploring the relaxation of two essential assumptions at the same time: the constant availability of empty trailers across all yards and the treatment of empty trailers as discrete, predetermined orders from customers. Furthermore, while some studies have tackled ad-hoc changes, they often treat them as a statistical phenomenon in stochastic models, without fully addressing the requirement for immediate, real-time response (Zhang et al., 2011a).

To tackle these three challenges simultaneously, a novel approach named Demand-Dependent Trailer Repositioning with Deep Reinforcement Learning (Dray-Q) is proposed. Dray-Q takes the advantage of a variant of the well-established Deep Reinforcement Learning method, Rainbow Deep Q-Networks (DQN). This approach accounts for both loaded and empty trailer movements, the limited overall number of trailers in the system, and relaxes the unrealistic assumption of readily available empty trailers. Empty trailers are allowed to move just in time to maximize the balance of supply and demand across yards, making them dependent on existing demand. Dray-Q accommodates delays with high penalties and efficiently handles ad-hoc changes and unplanned orders without the need for frequent optimization model runs again. This method has the potential to significantly enhance the efficiency and productivity of drayage operations within the supply chain. In essence, this paper distinguishes itself from prior research through the following contributions:

(1) A novel framework for drayage operations, called Dray-Q, has been proposed to address two primary bottlenecks of drayage operations. These include the dependence of empty trailer repositioning on demand, the limitation of available trailers, time windows for pickup and delivery, and (un)-loading time to participate in the system.

(2) In order to tackle the challenge of real-time trailer repositioning, this study fine tunes the state-of-the-art reinforcement learning algorithm, Rainbow-DQN, to learn the complexities of Dray-Q framework. This approach enables proactive responses to unanticipated changes in drayage operations' trailer repositioning planning.

(3) A multi-objective, novel reward function is designed to address both the balance between empty trailer supply and demand and the minimization of delays, while also considering customer priority in the dispatching of empty trailers.

(4) A formulation using Mixed Integer Linear Programming (MILP) to address trailer repositioning considering trailer limitations, demand-dependent movements, and delay penalties has been presented. This MILP model was applied and solved for small-scale cases, and its results were compared with those obtained using the proposed Dray-Q framework and baseline models.

The subsequent section of this paper has been structured as follows: The literature review is provided in Section 2 to offer a thorough examination of pertinent research. Section 3 furnishes a formal account of the Dray-Q approach and its constituent parts. In Section 4, an extensive analysis of the experimental outcomes are presented, which incorporates a comparison with alternative models and baselines. Ultimately, the conclusions derived from this study are delineated in Section 5.

2. Literature review

In the following section, we review the literature on the proposed approach, including models for pickup and delivery in drayage operations and reinforcement learning based applications that provide insights into solving similar problems with deep reinforcement learning.

2.1. Trailer repositioning in drayage operations

The scholarly literature has extensively studied drayage operations. This section investigates related literature on trailer repositioning, divided into single-stage and multi-stage models in terms of whether the models take account of other planning and scheduling tasks during the trailer repositioning.

Single-stage Models Single-stage models exclusively address trailer repositioning without considering the following planning and scheduling tasks. A preliminary inquiry undertaken by Zhang et al. delved into the intricacies associated with identifying the optimal trajectory for maneuvering heterogeneous trailers (Zhang et al., 2011b). To address this problem, the authors approached it as a multiple Traveling Salesman Problem (m-TSP) that incorporated time windows and resource constraints. They translated the problem into a directed graph representation and employed a reactive tabu search algorithm to solve it. Notably, this formulation encompassed various time-related factors such as waiting time, and loading and unloading time, among others. While the research focused on a limited number of trailers, it treated the movement of empty trailers as a separate entity, regardless of the movement of loaded trailers. In a different study, Zeng et al. employed reinforcement learning, specifically Q-Learning, to dispatch trailers in order to mitigate yard congestion during loading or unloading operations (Zeng et al., 2012). The proposed approach involved the utilization of a simulation model to construct the system environment, while the Q-Learning algorithm was employed to acquire optimal dispatching rules for yard trailers. Shiri et al. conducted a study wherein time-window constraints pertaining to arrival and delivery times were taken into account in order to ascertain the most favorable sequence of drayage operations (Shiri and Huynh, 2016). The objective was to determine the optimal allocation of empty trailers considering the spatial distribution of demand and supply, including customer locations, trailer terminals, empty trailer depots, and truck depots. To tackle this problem, a MILP model
was developed, which was subsequently solved using a customized variant of the tabu search algorithm. By positing the presence of available empty trailers throughout yards and treating trailer movements as individual orders, the task of trailer repositioning could be reformulated as a route planning problem. It is noteworthy that the presence of readily accessible trailers in all yards obviated the requirement for trailer repositioning. Nevertheless, in the context of route planning, various types of trailer repositioning, including live, non-live pickup, and drop, are taken into account. Song et al. introduced a novel solver that integrated column generation with branch-and-cut techniques, aiming to expedite the identification of an optimal and precise solution for trailer repositioning (Song et al., 2017). This approach termed the branch-and-price-and-cut algorithm, facilitated the attainment of an exact solution for large-scale trailer repositioning problems. In a separate study conducted by Zhang et al. to formulate trailer repositioning, despite assuming the independency of empty trailer movement, the constraint imposed by the limited availability of trailers in each yard introduced a nonlinearity in modeling and heightened the complexity of the problem (Zhang et al., 2020). To address this issue, the authors proposed a mixed binary nonlinear programming model as a solution methodology. This proposed model aimed to optimize the allocation of trailers while considering various constraints like time windows and minimum empty trailer levels at each yard at each time step. To obtain an effective solution, a Large Neighborhood Search (LNS) algorithm was employed to explore the solution space and identify optimal configurations.

**Multi-stage Models** In practical drayage operations, trailer repositioning represents the initial phase, in conjunction with route planning and driver scheduling. To address these multiple phases in a comprehensive manner, multi-stage models endeavor to partition trailer repositioning into distinct tasks to accommodate more practical constraints and integrate them into a unified framework. One of the earlier studies on trailer repositioning examined both loaded and empty trailers but made the assumption that empty trailers were always available (Ileri et al., 2006). Consequently, the model focused on finding feasible route assignments and driver scheduling rather than optimal trailer movements across yards. The model encompassed two distinct phases, namely route planning and driver scheduling. In the pursuit of route planning, a tree search algorithm was employed. As for the driver scheduling, a MILP model was devised. In another study, trailer repositioning along with route planning in terms of truck movement was considered (Zhang et al., 2011a). Zhang et al. conducted a comprehensive classification of trailer repositioning tasks, segmenting them into two primary categories: well-defined tasks, which pertained to movements of loaded trailers, and flexible tasks, which involved the repositioning of empty trailers without predetermined time windows. In order to address this challenge, the authors formulated drayage routing plans through the resolution of a two-stage stochastic optimization model at multiple decision epochs throughout the operational day. To accomplish this, they developed an innovative MILP model by leveraging a modified version of the Partial Swarm Optimization (PSO) algorithm. Braekers et al. proposed two models, one sequential and the other integrated, for dispatching both loaded and empty trailers (Braekers et al., 2013). The integrated model outperformed the sequential one in terms of minimizing the total number of vehicles involved and the distance traveled. The primary contribution of this study was a hierarchical multi-objective function of minimizing the number of vehicles used and total distance traveled that incorporates both empty and loaded trailer movement in conjunction with routing considerations. Finally, Cui et al. adapted a two-stage solution framework to optimize trailer repositioning and truck routes (Cui et al., 2022). The first stage generated tractors' schedules to maximize profit, and the second stage ensured sufficient trailers at each trailer depot during container drayage operations. For solving the proposed model, a hybrid large neighborhood search and tabu search (LNS-TS) heuristic was developed. A summary of papers based on three assumptions: Trailer availability, empty moves as an order, and handling of delay and ad-hoc orders is provided in Table 1.

### Table 1

<table>
<thead>
<tr>
<th>Paper</th>
<th>Year</th>
<th>Trailer availability</th>
<th>Empty moves are order</th>
<th>Delay</th>
<th>Ad-hoc orders</th>
<th>Objective function</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ileri et al. (2006)</td>
<td>2006</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Min Cost</td>
<td>Tree Search, MILP</td>
</tr>
<tr>
<td>Zhang et al. (2011a)</td>
<td>2011</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Min Moves</td>
<td>MILP, PSO</td>
</tr>
<tr>
<td>Zhang et al. (2011b)</td>
<td>2011</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Min Makespan</td>
<td>Tabu Search</td>
</tr>
<tr>
<td>Zeng et al. (2012)</td>
<td>2012</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Min (Un)Load Time</td>
<td>Q-Learning (RL)</td>
</tr>
<tr>
<td>Braekers et al. (2013)</td>
<td>2013</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Min Vehicles, Dist</td>
<td>Simulated Annealing</td>
</tr>
<tr>
<td>Shiri and Huynh (2016)</td>
<td>2016</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Min Distance</td>
<td>MILP, Tabu Search</td>
</tr>
<tr>
<td>Song et al. (2017)</td>
<td>2017</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Min Makespan</td>
<td>MILP</td>
</tr>
<tr>
<td>Zhang et al. (2020)</td>
<td>2020</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Min Used Vehicles</td>
<td>Non-linear IP, LNS</td>
</tr>
<tr>
<td>Cui et al. (2022)</td>
<td>2022</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Max Profit</td>
<td>LNS-Tabu Search</td>
</tr>
</tbody>
</table>

Reinforcement learning has shown significant progress in combinatorial optimization (Delarue et al., 2020). In recent times, a significant corpus of literature on combinatorial optimization problems has been dedicated to examining the implementation of
reinforcement learning techniques to address routing problems in their conventional configuration (Nazari et al., 2018; Kool et al., 2019). However, it should be noted that the efficacy of reinforcement learning models may become compromised in the presence of supplementary constraints imposed on the problem domain. Additionally, the management of uncertainties poses another challenge that can impact both the learning aptitude of the agent and the feasibility of the model for deployment in real-world scenarios. Therefore, it is essential to gain insights from prior academic research focusing on comparable domains to the trailer repositioning that deals with constraints and uncertainties.

### Constraint Handling

Constraints in RL problems can take various forms depending on their intended purposes. Specifically, constraints are often imposed in situations where not all actions are available. The treatment of such constraints has been a subject of considerable research interest, with scholarly investigations generally falling within at least one of three distinct categories: rule-based action space design, context-based learning, and penalty-based learning. The first approach, rule-based action space design, involves defining intuitive rules as actions to reduce the action space size (Qian et al., 2022; Jiao et al., 2021; Qin et al., 2021; Deng et al., 2022). The utilization of rule-based action space has been the subject of extensive research within the context of dynamic programming (Bertsekas, 2005). In trailer repositioning context, rule-based action space design could involve defining rules for assigning resources and scheduling tasks based on certain criteria, such as availability, priority, or capacity constraints. The second approach, context-based learning, aims to enhance the agent’s decision-making capabilities by incorporating contextual information beyond the current state and available actions (Lin et al., 2018; Liu et al., 2022; Yu and Hu, 2022; Li et al., 2022; Zhang et al., 2022). In trailer repositioning, context-based learning can involve incorporating contextual information like trailer availability, order deadlines, and order dependencies beyond the current state and available actions. Lastly, penalty-based learning focuses on using strongly penalized rewards or gradient decent part to discourage the agent from taking forbidden actions (Losapio et al., 2021; Chandak et al., 2020; Boutillier et al., 2018). Within the realm of trailer repositioning, penalty-based learning methods can be employed to deter unfavorable actions. For instance, these methods can effectively discourage an agent from selecting a trailer that is unavailable or already designated for another order or en route.

### Uncertainty Handling

Uncertainty refers to a situation in which there is a lack of knowledge or information about a specific event or outcome. In the context of problems akin to trailer repositioning, ad-hoc changes and delays represent the main sources of uncertainty. While ad-hoc changes can be handled by rerunning a trained reinforcement learning model, other approaches have attempted to model them more systematically (Silva and Proso, 2022; Silva et al., 2023). However, addressing delays requires the agent to be trained to account for the delay structure. Several studies have endeavored to treat delay as a form of uncertainty in pickup and delivery problems (Zhang et al., 2022).

To optimize trailer repositioning in real-world drayage problems, several factors must be considered. These include trailer limitations, dependent movement of empty trailers based on demand, and real-time handling of delays and ad-hoc changes in orders. The literature lacks a solution that addresses all of these requirements through a unified multi-objective function. An effective approach to mitigate the aforementioned issues can be accomplished through the utilization of RL framework. Nonetheless, the RL paradigm necessitates meticulous consideration of state space, action space, and reward signal, as well as the adoption of expert-based methods such as penalty-based learning to tackle action space constraints, and environmental uncertainties, thereby facilitating the achievement of success in trailer repositioning tasks in practical settings.

### 3. Method

In this section, we start by presenting a MILP mathematical model to effectively tackle the problem of trailer repositioning in drayage operations. For modeling trailer repositioning in drayage operations with reinforcement learning, we begin by proposing a new concept, deficit, in order to formulate trailer repositioning with reinforcement learning, followed by an exposition of the complete dynamics of the environment necessary for training the reinforcement learning agent. This includes a detailed description of the problem. Subsequently, we examine the design of the reward function in detail and conclude with a comprehensive examination of the agent training algorithm.

#### 3.1. Mixed integer linear formulation

MILP formulation represents an attractive modeling approach for trailer repositioning problems. At its core, the implementation of the approach relies on the relaxation of two out of three key simplifications employed in prior studies:

1. **Limited Availability of Trailers**: The quantity of available trailers is constrained, indicating a finite and restricted supply.
2. **Dynamic Trailer Movements**: The repositioning of empty trailers to specific yards and time steps is contingent upon demand factors, rather than being predetermined by customer orders.

This model also assumes that trailers possess the same sizes and types, with travel times being deterministic. The importance of customers can be distinguished through the cost of delay for each customer. Commencing from the problem notations, the variables of the approach relies on the relaxation of two out of three key simplifications employed in prior studies:

- **Problem Notations**
- **Variables**: $i, j, t$
- **Constraints**: $g_{ij}(t + 1) = g_{ij}(t) + d_{ij}(t - \tau_{ij} - \beta_j) - f_{ij}(t + a_i)$
- **Objective Function**: $\max \sum_{i} \sum_{j} \sum_{t} p_{ij} f_{ij}(t) - ec_{ij} c_{ij}(t) - dc_{ij} g_{ij}(t)$

Inspired from Upadhyay and Bolia (2014) the MILP formulation for trailer repositioning in drayage operations is as follows:

$$\max \sum_{i} \sum_{j} \sum_{t} p_{ij} f_{ij}(t) - ec_{ij} c_{ij}(t) - dc_{ij} g_{ij}(t)$$

$$s.t. \quad g_{ij}(t + 1) = g_{ij}(t) + d_{ij}(t - \tau_{ij} - \beta_j) - f_{ij}(t + a_i) \quad \forall \quad i, j, t$$
when the optimal solution may be infeasible in some parameter settings. To address this issue, the majority of the literature has
et al., 2006). In a real-world scenario with large-scale drayage operations, it is impractical to obtain the precise solution, let alone
simplification assumption.
model, thereby facilitating real-time adaptation to changes. Consequently, the MILP model in this section does not relax the third
as noted in Section 1, one of the key benefits of utilizing RL is its capacity to rapidly generate an optimal policy through a trained
inherent structure of MILP models, handling the third issue, ad-hoc changes in order, is highly challenging, if not unfeasible, and
Additionally, constraints (7) specify the type of decision variables as non-negative integer variables.
minimize the total cost associated with empty trailer movements. Finally, it strives to minimize the total cost associated with service
revenues earned and the operational costs incurred in fulfilling demands) by satisfying customer requirements. Second, it seeks to
Constraints (2) ensure that unfulfilled demands are postponed but at a penalty cost. Constraints (3) guarantee the preservation
Constraints (4) ensure the availability of sufficient empty trailers at each destination yard, while accounting
Constraints (5) restrict the total number of trailers to be less than the maximum available trailers. Constraints (6) specify that the initial value for delay movement is zero.
\[ \sum e_{ij}(t) + f_{ij}(t) \leq mTTr \]  
\[ g_{ij}(0) = 0 \quad \forall \quad i,j \]  
\[ f_{ij}(t) \geq 0 \quad , e_{ij}(t) \geq 0 \quad , g_{ij}(t) \geq 0 \quad \forall i,j,t \]  
The objective function (1) comprises three components. First, it aims to maximize the overall profit (the difference between revenues earned and the operational costs incurred in fulfilling demands) by satisfying customer requirements. Second, it seeks to minimize the total cost associated with empty trailer movements. Finally, it strives to minimize the total cost associated with service quality in terms of delay and latency.
Constraints (2) ensure that unfulfilled demands are postponed but at a penalty cost. Constraints (3) guarantee the preservation of trailer flow at each location and time period. This encompasses the effects of trailer loading and unloading, as well as trailer travel times by trucks. Constraints (4) ensure the availability of sufficient empty trailers at each destination yard, while accounting for the impact of empty trailers that arrive at the destination and are loaded at that time. Constraints (5) restrict the total number of trailers to be less than the maximum available trailers. Constraints (6) specify that the initial value for delay movement is zero. Additionally, constraints (7) specify the type of decision variables as non-negative integer variables.
This model embodies the majority of pragmatic assumptions within the domain of drayage operations, however, owing to the inherent structure of MILP models, handling the third issue, ad-hoc changes in order, is highly challenging, if not unfeasible, and as noted in Section 1, one of the key benefits of utilizing RL is its capacity to rapidly generate an optimal policy through a trained model, thereby facilitating real-time adaptation to changes. Consequently, the MILP model in this section does not relax the third simplification assumption.
Furthermore, MILP problem formulations, while considered an optimal solution, are affected by the curse of dimensionality (Ileri et al., 2006). In a real-world scenario with large-scale drayage operations, it is impractical to obtain the precise solution, let alone when the optimal solution may be infeasible in some parameter settings. To address this issue, the majority of the literature has

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of notation.</td>
</tr>
<tr>
<td><strong>Indices and sets</strong></td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>i</td>
</tr>
<tr>
<td>j</td>
</tr>
<tr>
<td>t</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td>( l_{pj} )</td>
</tr>
<tr>
<td>( e_{ij} )</td>
</tr>
<tr>
<td>( d_{c_{ij}} )</td>
</tr>
<tr>
<td>( d_{ij}(t) )</td>
</tr>
<tr>
<td>( a_{ij} )</td>
</tr>
<tr>
<td>( \beta_{ij} )</td>
</tr>
<tr>
<td>( \tau_{ij} )</td>
</tr>
<tr>
<td>( \epsilon_{min} )</td>
</tr>
<tr>
<td>( mTTr )</td>
</tr>
<tr>
<td><strong>Decision variables</strong></td>
</tr>
<tr>
<td>( f_{ij}(t) )</td>
</tr>
<tr>
<td>( e_{ij}(t) )</td>
</tr>
<tr>
<td>( g_{ij}(t) )</td>
</tr>
</tbody>
</table>
The finite number of trailers available for redistributing orders among different yards poses a challenge in maintaining a smooth loading and dispatching process. This entails ensuring both the transportation of loaded trailers for deliveries and the presence of empty trailers at yards before scheduled pickup time windows. Achieving equilibrium in the quantity of empty trailers across various yards at different time intervals, contingent upon the orders received is crucial. This requires coordinating the allocation of limited empty trailers among all yards to ensure an adequate supply when needed. It involves meticulous monitoring of empty trailer movements and the statuses of other trailers at different temporal and spatial intervals to formulate an optimal dispatching strategy.

Reinforcement learning agents require relevant information within each state to understand the current situation, along with a dependable signal provided by a reward function to guide their decision-making process. By projecting this information in the state onto the appropriate actions, reinforcement learning agents strive to obtain an optimal policy that maximizes the expected reward over the entire episode. Of particular significance is the inclusion of key information pertaining to the yard’s status, specifically its trailer requirements or availability for dispatch to other yards.

To properly evaluate the availability status of trailers in each yard and integrate this information into the state space and reward function, it is essential to establish a suitable conceptual framework. In this regard, we employ the concept of Deficit, which is borrowed from the realm of inventory optimization, as a means to quantify trailer availability in a yard (Upadhyay and Bolia, 2014). The deficit represents the difference between the number of incoming trailers, the number of outgoing trailers, and the number of trailers currently stored in the yard over a given time interval. This is expressed in the following formulation for yard \( i \) at time step \( t_n \):

\[
\text{Deficit}_{i, t_n} = (\text{Incoming}_{i, t_n} - \text{Outgoing}_{i, t_n}) + \text{Capacity}_{i, t_n}.
\]

Within this context, a negative deficit value signifies an insufficient number of trailers within the yard, while a positive value indicates a surplus. The magnitude of the negative deficit value serves as an indicator of the level of urgency with which the yard requires additional trailers to fulfill existing orders.

A visual representation of the operational procedures, deficit level, and trailer types typically encountered within a yard is depicted in Fig. 2. On the right-hand side, there exist four yards exhibiting distinct deficits in available trailers at a specific time step. Within this context, deficit refers to the variance between incoming trailers (associated with delivering orders from other yards) and outgoing orders (related to dispatching orders to other yards), accounting for the currently available trailers that act as substitutes for meeting the minimum requirement of each yard. Ideally, an absence of surplus or deficit in available trailers is sought. Therefore, the objective entails balancing the number of trailers to maintain their proximity to zero.
To delve further into the interplay between trailer status and yard deficit levels, let us closely examine a selected yard. Within this specific context, considering a minimum threshold of two empty trailers for this yard, the current time step exhibits a deficit level of +2, indicating a surplus of available trailers within the yard in question. The yard currently accommodates three empty trailers (i.e., trailers 1,2,3). One of these trailers is presently undergoing the unloading process and will soon become available for subsequent assignments, which may entail an empty movement to another yard or loading in preparation for dispatching an order to another destination (trailer 4). Furthermore, there exists an additional trailer that has transitioned from an empty state to a loading state, as it is being prepared for delivery (trailer 5). In the vicinity of the yard’s front area, two trailers have recently arrived: an empty trailer dispatched from another yard, intended for storage in anticipation of future demands (trailer 6), and a loaded trailer that, upon unloading, will transform into another empty trailer (trailer 7). Additionally, three trailers are currently prepared for dispatch, consisting of one loaded trailer (trailer 8) and two empty trailers (trailer 9,10), all of which are scheduled to be sent to their respective destinations.

The main goal in this context is to guarantee the presence of a sufficient number of empty trailers at the necessary times for the timely dispatch of both present and future orders. It is crucial to avoid a surplus or deficit of empty trailers within a yard, as such imbalances can lead to undesirable outcomes such as delay in order delivery in the whole system of dispatching. To achieve this goal, it becomes imperative to adopt a comprehensive perspective encompassing all yards, orders with their respective pickup and delivery time windows, and the dynamic status changes of trailers. This comprehensive comprehension requires the creation of an optimized dispatching policy for empty trailers. Simultaneously, it involves updating the status changes at various time intervals to accurately monitor the movements of these trailers. Attaining such a policy necessitates the application of an optimization approach aimed at minimizing delays and ensuring the timely fulfillment of order deliveries in the domain of drayage operations.

3.3. Dray-Q framework as a Markov decision process

The problem we are considering is the trailer repositioning problem, which is abstracted using a MILP framework. This abstraction simplifies the dynamic nature of real-world scenarios, which are subject to frequent ad-hoc changes and delays affecting order dispatching. The dispatching process involves interactions among three entities: orders, trailers, and yards. Understanding how orders and trailers transition between yards over time is essential for grasping the system’s dynamics. Additionally, we aim to analyze the components of Dray-Q frameworks within the context of a Markov Decision Process (MDP).

The objective is to efficiently manage order pickup and delivery within specified time windows, using a limited number of trailers across various yards. Each order must be picked up from its source yard and delivered to its destination within designated time frames, minimizing delays. Orders can only be dispatched if an empty trailer is available in the source yard before the pickup time window begins, enabling the loading process. Therefore, the challenge lies in balancing trailer distribution across yards to ensure trailer availability at source yards before loading.

Let $M$ represent the set of trailers available in a drayage operation company. The deficit number $D$, representing the empty trailer level in each yard serves as a crucial indicator. Regardless of the number of time steps considered ahead of the current time step to compute the deficit level for yard $i$, the deficit levels of all yards, combined with the current status of all trailers (whether they are occupied or empty), form the state space of the problem.

Let $S$ represent the state space of the problem, defined as the set of all possible combinations of deficit levels for each yard and the status of all trailers. Mathematically, we can represent the state space as:

$$S = \{(d_1,d_2,...,d_n),(t_1,t_2,...,t_m)\}$$

where $n$ is the number of yards, and the $t_j$ refers to whether trailer $j$ is occupied or empty.

To show that the problem follows the Markov property, we need to prove that the probability distribution of the next state depends only on the current state, not on the entire history of states. Mathematically, for any two states $s_i$ and $s_{i+1}$ in the state space $S$, and any time step $t$, the following condition must hold:

$$P(s_{i+1}|s_t,s_{t-1},...,s_0) = P(s_{i+1}|s_t)$$

Given the definition of the state space $S$ and the dynamics of the trailer repositioning problem, the transition from one state to another depends only on the current state (i.e., the deficit levels of all yards and the status of all trailers), not on the entire history of states.

Therefore, the problem representation satisfies the Markov property, providing both necessary and sufficient conditions for modeling the problem as a Markov Decision Process.

In the subsequent sections, we will delve deeper into the examination of different components of MDP formulation including the action space, environmental dynamics, state space, and the reward function, which represents one of the notable contributions of the research and warrants further investigation.

3.4. Action space

Each instance of trailer movement corresponds to either the transportation of an order to a designated yard or the repositioning of an empty trailer to a different yard in preparation for an upcoming order. In the event that an empty trailer is present at the pick-up yard, load movements are transported automatically, without any decision-making. Conversely, to ensure the timely dispatch of orders, it is imperative to initiate preliminary action concerning empty trailers. Consequently, the action space $A$ for trailer
repositioning can be defined as a set of trailer–yard pairs with the goal of making the right decision about empty trailers at the right moment. This notation has been shown in

\[ A = \{(t, y) \mid t \in \text{Trailers}, y \in \text{Yards}\}. \]  

For instance, given a fleet of 3 trailers and 2 yards, the action space can be represented as 

\[ a = \{(t_1, y_1), (t_1, y_2), (t_2, y_1), (t_2, y_2), (t_3, y_1), (t_3, y_2)\}. \]

Within this context, the agent is capable of selecting an action that preserves the current empty trailer at its current yard.

3.5. Environment dynamism

The dynamic nature of the environment is illustrated in Fig. 3, which highlights the primary steps involved.

According to Fig. 3, the environment comprises four main modules namely: Empty Trailer Repositioning, Order to Packing, Order Status Updater, and Trailer Status Updater which are operating collaboratively. Like all environments in reinforcement learning, the input for this environment is a vectorized action representing trailer–yard pairs. This input is directed to the Empty Trailer Repositioning Module. At this stage, two scenarios may unfold: if the action is prohibited, indicating that the designated trailer is already engaged in another task, it will proceed directly to the reward function and trigger a Forbidden Action Penalty, resulting in a negative reward value that signals the agent to avoid such actions. Alternatively, if the action is permissible, it will be executed, involving the dispatching of an empty trailer from its current yard to the assigned yard. Concurrently, the accumulated deficit based on the n-step deficit will be computed and forwarded to the reward function.

After applying the action, the repositioning of orders and loaded trailers takes place, and their statuses are updated accordingly. Dispatching orders hinges on the availability of empty trailers, as determined in the Empty Trailer Repositioning Module. However, the process of loading orders and updating delay costs requires further attention, which is where the Order to Packing Module comes into play. This module arranges orders based on their delay costs and selects the most urgent one. If the packing time has elapsed and there is at least one empty trailer available at the source yard, the order begins loading; otherwise, it impacts the delay cost in the reward function, and the next order enters the workflow. The model employs a Time Window-based Delay calculation approach, wherein the discrepancy between the sum of loading and travel times (i.e., the first arrival time at the destination) and the last permissible delivery time determines the delay. Let \( L(t) \) denote the loading time for an order at time step \( t \), while \( T(t) \) and \( W(t) \)
represent, chronologically, the travel time for the order at time step \( t \) and the last permissible delivery time for the order at time step \( t \). The fuzzy delay calculation is formulated as follows:

\[
\text{Delay}(t) = \max\{0, (L(t) + T(t)) - W(t)\}
\]

(10)

Accordingly, the delay cost can be accumulated progressively at each time step as:

\[
\text{Delay Cost}(t) = \sum_{t=0}^{t} \text{Delay}(\tau)
\]

(11)

In situations where the difference is positive or zero, the delay cost accumulates gradually at each time step. This enables the agent to adjust its priorities in dispatching empty trailers based on the remaining time until a pickup or delivery time window ends, resulting in a higher reward. If an order was in the loading status in a previous time step and is now ready for dispatch, it is promptly sent to its assigned destination.

Following the dispatch of empty and loaded trailers, the next step involves updating the status of dispatched orders using the Order Status Updater Module. This module assigns a label to each order based on its current status and determines its end time step for future calculations. Similarly, the Trailer Status Updater Module performs the same function for trailers, labeling them based on their current status in the overall flow. At each time step, the environment returns the obtained reward and current state based on the trailer status and yard deficit levels, before transitioning to the next time step.

### 3.6. State space

The states of the three entities in question (i.e., orders, trailers, and yards) are subject to continual state transition as they progress through various yards over time. Consequently, the RL environment must represent this dynamicity and the status of all trailers, including whether they are empty or loaded, to present a current status quo to the agent. Hence, the state space is a combination of trailer status and an n-step forecast of the deficit level for all yards.

Let \( S \) represent the state space. The state space \( S \) is a combination of the trailer status and an n-step forecast of the deficit level for all yards. Each state \( s \) in \( S \) can be defined as a tuple \( S = (T, D) \), where \( T \) represents the trailer status and \( D \) represents the n-step forecast of the deficit level for all yards. Trailer status \( T \) includes information about whether each trailer is empty or loaded. For example, \( T = (t_1, t_2, \ldots, t_m) \) represents the status of \( m \) trailers, where \( t_i \) denotes the status of the \( i \)-th trailer (empty or loaded). The deficit level forecast \( D \) provides information about the projected deficit level for each yard if no adjustments are made. It considers the potential impact of preceding actions on the current load movements. The forecast can be represented as \( D = (d_1, d_2, \ldots, d_y) \), where \( d_i \) represents the forecasted deficit level for the \( i \)-th yard. Therefore, the state space can be formally expressed as:

\[
S = \{(T, D) \mid T = (t_1, t_2, \ldots, t_m), D = (d_1, d_2, \ldots, d_y)\}.
\]

(12)

The rationale behind the n-step ahead of the deficit level is to apprise the agent of the potential impact on the yards’ deficit level if no adjustments are made. Given that the presence of an empty trailer in a yard is a prerequisite for dispatching load movements (i.e., orders), the actions executed in preceding time steps can impact current load movements. Consequently, a mechanism is required to alert the agent of the potential deficit levels for each yard if no changes are made. In summary, the state space coordinates effectively with the action space, in which the state transmits two crucial signals to the agent regarding the trailer and yard statuses, and the agent selects the optimal action (i.e. trailer–yard pairs) accordingly.

### 3.7. Multi objective penalty-based reward function

The effectiveness of an reinforcement learning agent’s training heavily relies on the reward function. When it comes to dispatching both loaded and empty trailers, creating a reward function becomes crucial. It must balance the reduction of delays while ensuring connectivity and smooth operations. The representation of multiple objectives within a single numerical reward signal poses significant challenges. Thus, the formulation of a multi-objective reward function \( R \) demands diligent effort and attention to detail. For trailer repositioning, the single reward function is obtained as:

\[
R = w_a \sum_{t} L(D(t), t) \cdot Im - w_b \cdot \sum_{t} Dl - w_c \cdot (FP \cdot FA).
\]

(13)

To elaborate more on this reward function, our novel reward design consists of three main parts:

1. Deficit \( D_{i,t} \): the concept of deficit can be linked to its corresponding definition in inventory management in which deficit refers to the difference between the available resources and the desired or expected quantity in a given context. Specifically, for trailer repositioning, the deficit is defined as the expression in:

\[
D_{i,t} = (Im_{t,i} - Out_{t,i}) + Capa_{i} - \epsilon_{min}.
\]

(14)

Here, the deficit refers to the discrepancy between incoming and outgoing trailers, added to the current availability of empty trailers. Furthermore, we introduced the minimum yard level (or service level) as a novel term in the formal definition of the deficit concept, reflecting practical requirements in drayage operations.
It is important to note that in this study, the value of $t$ is computed for a period of four time steps ahead, where $n = 4$. The motivation behind this decision lies in the objective of providing a timely and informative signal to the agent regarding the deficit. By selecting time intervals of 15 min and taking into account the loading and travel durations, looking ahead to the next hour emerges as a practical choice for conveying the appropriate signal to the agent, thereby facilitating its learning process.

Moreover, the Laplace function $L$ of current deficit at yards multiplied by yard importance vector $I_{m_i}$. This vector, serving as a coefficient, has the capacity to either augment or diminish the current deficit depending on the yard’s significance. By incorporating this mechanism, the agent becomes capable of prioritizing the more significant yards, thereby enabling a more strategic decision-making process.

As depicted in the left side of Fig. 4. On the right side, another example shows a deficit of +2, indicating an excess of trailer levels in the yard. The main objective of this reward function is to maintain a balance of empty trailers at each yard, incentivizing the RL agent to achieve equilibrium in empty trailer availability across all yards, thereby maximizing the reward. With the state, action, and reward design established, the focus now shifts to the agent engine, which must learn to select the optimal actions at each time step.

3.8. Rainbow-DQN: Agent training algorithm for learning optimal trailer repositioning policy

Deep Q-Networks (DQN) faces issues in discrete action space reinforcement learning, including instability and slow convergence (Zeng et al., 2022). Rainbow-DQN, an advanced extension, addresses these with Double DQN, Dueling DQN, Prioritized Replay, Noisy Networks, N-step return, and Distributional networks, making it effective for complex tasks such as trailer repositioning (Hessel et al., 2017). The overall overview of Rainbow-DQN and its different components are illustrated in Fig. 5.

Accordingly, rainbow-DQN enhances traditional DQN by integrating several advanced techniques, each of which addresses specific challenges encountered in reinforcement learning tasks. In the context of the trailer repositioning problem, Rainbow-DQN offers several advantages over vanilla-DQN:

(1) Prioritized Experience Replay: Rainbow-DQN utilizes prioritized experience replay, which prioritizes experiences based on their temporal difference error. This technique allows the agent to focus more on learning from experiences that are more informative or surprising, leading to more efficient and effective learning. In the trailer repositioning problem, prioritized experience replay helps the agent to prioritize learning from critical events such as successful trailer repositioning actions or significant delays, thereby improving its decision-making process.
(2) Double Q-learning: Rainbow-DQN incorporates double Q-learning, which addresses the issue of overestimation of action values commonly observed in traditional Q-learning approaches. By using two separate value functions to estimate action values and alternating between them during updates, double Q-learning reduces the bias introduced by overestimation, leading to more accurate value estimation. In the context of trailer repositioning, accurate value estimation is crucial for the agent to make informed decisions about trailer movements, considering the dynamic nature of the environment and the trade-offs between different objectives.

(3) Dueling Architectures: Rainbow-DQN employs dueling architectures, which decouple the estimation of state values and action advantages, allowing the agent to better understand the value of taking specific actions in different states. By separating the estimation of state values and action advantages, dueling architectures enable more efficient learning and better generalization across different states. In the trailer repositioning problem, dueling architectures help the agent to better assess the potential benefits of repositioning trailers in different yards, considering the current deficit levels and order priorities.

(4) Distributional Value Estimation: Rainbow-DQN incorporates distributional value estimation, which represents the distribution of expected returns rather than a single expected value. By capturing the uncertainty in the expected returns, distributional value estimation provides a richer representation of the environment dynamics and helps the agent to make more robust decisions. In the trailer repositioning problem, distributional value estimation enables the agent to account for uncertainties such as variability in order arrival times and loading durations, leading to more adaptive and resilient decision-making.

(5) Multi-step Learning: Rainbow-DQN integrates multi-step learning, which reduces the variance of gradient updates and enhances sample efficiency by considering multiple future states and rewards when updating the Q-values. By incorporating information from multiple future time steps, multi-step learning enables the agent to make more informed decisions and learn more effectively from past experiences. In the trailer repositioning problem, multi-step learning allows the agent to consider the long-term consequences of trailer movements and delays, leading to more strategic decision-making and better overall performance.

(6) Noisy Networks: Rainbow-DQN incorporates Noisy Networks, which introduces noisy parameters directly into the neural network architecture, allowing the agent to learn stochastic policies without requiring explicit exploration strategies such as epsilon-greedy. By introducing noise into the network parameters, Noisy Networks encourage exploration during training while still allowing for deterministic behavior during inference. In the context of trailer repositioning, where the environment dynamics can be complex and uncertain, effective exploration is crucial for the agent to learn robust policies that can adapt to different scenarios. By incorporating Noisy Networks into Rainbow-DQN, the agent is better equipped to explore the diverse set of actions available, such as repositioning trailers to different yards or adjusting priorities based on changing deficit levels and order requirements.

The details of the Rainbow-DQN algorithm can be found in Algorithm 1.

**Initialization** The algorithm starts by initializing a replay buffer $D$ and two sets of network parameters: primary ($\theta$) and target ($\theta^{-}$). Noisy Networks add noise during training for exploration, and Prioritized Replay assigns priorities to transitions based on error, favoring higher-error transitions for sampling during training.

**Experiencing** During training, the algorithm follows a decaying $\epsilon$-greedy policy: it selects random actions with probability $\epsilon$ and the highest Q-value actions otherwise. After each episode, $\epsilon$ gradually decreases, promoting exploration initially and exploitation later, while recording actions and their experiential tuples in the prioritized response buffer.
Algorithm 1: Rainbow-DQN for learning optimal trailer repositioning policy

**Input**: Replay buffer $D$ with capacity $N$, Target net parameters $\theta^\rightarrow \leftarrow \theta$, Distributional net $Z$ with weights $\beta$

**Input**: Prioritized replay memory hyperparameters $\alpha, \beta$, Noise parameters $\mu \leftarrow 0$, $\sigma \leftarrow \sigma_0$

**Output**: Optimal Trailer Repositioning Policy

1. $t = 1$

2. **foreach** $t$ in Epochs **do**

3. **Experiencing:**

   - Observe tuple of deficit and trailer status as state $s_i$
   - Select action $a_i$ (trailer–yard unique index in action space) with decaying $\epsilon$-greedy policy
   - Execute action $a_i$ and observe multi-objective reward signal $r_i$ and next state $s_{i+1}$
   - Add transition tuple of $(s_i, a_i, r_i, s_{i+1})$ to reply buffer $D$

4. **Acting:**

   - Sample a batch of transitions $(s_i, a_i, r_i, s_{i+1})$ from $D$ with priority $p_i$
   - Computen-step returns $G_t : t + n - 1^{(t)}$
   - Compute importance-sampling weights $w_i = \left( \frac{1}{N} \cdot \frac{1}{p(i)} \right)^{\beta}$, where $P(i) = \frac{p}{\sum p_j} + \epsilon$

5. **Learning:**

   - Update distributional network $Z$ using minibatch $(s_i, a_i, r_i, G_t^{(t)}, w_i)$
   - Update online action–value network $Q$ using minibatch $(s_i, a_i, w_i)$
   - Compute noisy target values $y_i = r_i + \gamma \cdot \max_a Q^\leftarrow(s_{i+1}, a) + \eta_i$, where $\eta_i \sim \mathcal{N}(0, \sigma^2)$
   - Update the target network parameters $\theta^\rightarrow \leftarrow \tau \cdot \theta + (1 - \tau) \cdot \theta^\rightarrow$
   - Update priorities $P_i \leftarrow |y_i - Q^\leftarrow(s_i, a_i)| + \epsilon$
   - Update the noise parameters $\sigma \leftarrow \max(\sigma_{\text{min}}, \sigma_0 - \left( \frac{t}{t_{\text{decay}}} \right))$
   - Update the primary network parameters $\theta \leftarrow \theta - \frac{\alpha}{|B|} \sum_{i \in B} w_i \cdot \nabla_a Q^\leftarrow(s_i, a_i) \cdot \nabla_a Q^\leftarrow(s_i, a_i)$, where $B$ is the batch of transitions

6. Decay $\epsilon$ and $\beta$ according to schedule

7. $s \leftarrow s_{i+1}$

**end**

- **Acting** In this phase, the algorithm aligns with supervised learning by comparing real values from the target network to estimated values using the Double Dueling DQN technique, which partitions the Q-function into two streams, one for state value and another for action advantage, mitigating overestimation. It then selects a subset of transitions from the prioritized replay buffer, computes n-step returns denoted as $G^{(t)}_{t:t+n-1}$ using the target network for action selection and the online network for value estimation, and calculates importance-sampling weights $w_i$ to scale the loss function during gradient descent to account for non-uniform sampling from the replay buffer.

- **Learning** In this phase, the algorithm updates multiple networks. It starts by computing the TD error $\delta_i$ for the distributional network update, aiming to minimize the difference between expected and observed returns. This is done for each atom in the distribution. Then, the algorithm updates the online action–value network by computing another TD error $\delta_i$, which considers the reward, discounted expected values, and noise, encouraging exploration and preventing overfitting. These TD errors guide the network parameter updates through gradient descent.

- **Loss function** The algorithm performs gradient descent to update the action–value network parameters based on the loss function. The loss function is as:

\[
L_i = -w_i \sum_j Z(s_i, a_i, \beta, \theta) \log Z(s_i, a_i, \beta, \theta) + \text{Huber loss(}$\delta_i$, $y_i$).
\]  

(16)

The loss function in Eq. (16) comprises two parts: a cross-entropy loss for the distributional network and a Huber loss for the double Q-learning update. The cross-entropy term minimizes the negative log-likelihood of the observed return distribution, while the Huber loss penalizes deviations between predicted and observed returns, combining the benefits of distributional networks and double Q-learning to mitigate overestimation bias. This loss function is central to Rainbow’s state-of-the-art performance on benchmark tasks.
4. Experimental results

In this section, we present our experimental results and accompanying analysis, which aim to verify and validate the contributions discussed in Section 1.

4.1. Experimental setting

Real-world Dataset For our analysis, we utilized real-world data obtained from a well-known drayage operation company based in Canada. This data encompasses various aspects such as orders, yard coordinates, travel durations, distances, and loading/unloading times. The order data is further categorized into pickup and delivery, containing details about the yard name, pickup/delivery time windows, and loading/unloading durations. An example of data for a single order is illustrated in Table 3.

Each movement type in the dataset comprises several attributes, including sequentially the event type, yard id, time windows (in hours), and loading/unloading times (in minutes). In this study, the event types are restricted to PLT (Pickup Loaded Trailers) and DLT (Drop Loaded Trailers), which are standard codes used in drayage operations. These codes serve merely as indices and do not influence the operational processes. Each yard name corresponds to a specific yard location, represented by its geographical coordinates, and the travel distances are computed based on actual Earth distances within a realistic context. Within the environment, a profile is created for each order to monitor changes in orders and the availability status of trailers in different yards at various time steps, reflecting the dynamic flow of trailers for order fulfillment across different yards. As for trailers, since they are identical, a list of trailers is maintained based on available trailer numbers, and their current status in different locations and times is tracked based on the dynamic flow within the environment.

Furthermore, the importance of each yard was assessed by analyzing the number of orders processed in the preceding year, with each yard maintaining a minimum buffer of empty trailers necessary for drayage operations. Additionally, our scenarios were formulated in accordance with the company’s principal dispatching policies. The experiments in this study are structured around several key scenarios, as detailed below:

(1) Orders: 50, Trailers: 5, Yards: 5
(2) Orders: 60, Trailers: 10, Yards: 5
(3) Orders: 70, Trailers: 20, Yards: 10
(4) Orders: 80, Trailers: 30, Yards: 10
(5) Orders: 90, Trailers: 40, Yards: 15
(6) Orders: 100, Trailers: 50, Yards: 15

It is important to note that the number of orders fluctuates around the scenario’s order count, following a normal distribution of $N(\mu = \text{Order Count}, \sigma = 1)$ in each epoch. While this variation in order numbers within each epoch adds to the variability in the learning process, it enhances the model’s ability to generalize and provide policies for different order counts close to the primary order count for that scenario, without necessitating separate models for each order count. It should be highlighted that the yards and trailers remain fixed for each scenario, whereas the initial trailer locations vary randomly in the reinforcement learning environment with each iteration.

Problem Setting There exist two primary criteria for conducting experiments, namely the amount of reward function and the level of delay amount across yards. Depending on the specific conditions, one or both of these criteria may be employed in the conducted experiments. Prior to conducting experiments, it is essential to discuss experimental settings. The first concerns the reward normalization coefficients, which were determined through trial and error. In this setting, $w_a = 100$, $w_b = 70$, and $w_c = 30$. The selection of these choices has been meticulously crafted to ensure that all components of the reward function exhibit a closely aligned scale range for normalization purposes. Regarding time steps, the planning horizon is one day, which is typical for drayage operation companies. Each day begins at 6:00 AM and ends at 23:59 PM, with the entire available time being divided into 15-minutes intervals. As stated in Section 3.3, the deficit is calculated based on the n-step ahead deficit state. In our setting, n-step is set to 4, which represents a one-hour ahead interval.

Baseline models In addition to the proposed model, we introduce several baseline models to evaluate the performance of Dray-Q. These models are described as follows:

(1) Closest to Urgent (CUP) Heuristic Model
(2) Tabu Search Metaheuristic Model
(3) Hybrid of Tabu Search and Simulated Annealing Model (TS + SA)
(4) Vanilla-DQN Model

Table 3
An example for an instance of data.

<table>
<thead>
<tr>
<th>Movement type</th>
<th>List of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pickup</td>
<td>[PLT, Yard 1, [10, 11], 45]</td>
</tr>
<tr>
<td>Delivery</td>
<td>[DLT, Yard 2, [13, 14], 90]</td>
</tr>
</tbody>
</table>
The CUP heuristic model serves as a commonly employed dispatching policy in our case study drayage operations company. Under this policy, planners dispatch an empty trailer (if needed) from the nearest surplus yard to the most deficient one at each time step. As a comparison, two metaheuristic algorithms were utilized as baselines. To explain further how these heuristic baselines are constructed, it is worth noting that once we developed the environment for reinforcement learning agents, it required actions in a specific format to return the reward and next state. Thus, we adopted the overall structure of the environment and adjusted the action selection policy to align with the structures present in the baseline models. For example, in tabu search, the mechanism for action selection involves generating a solution in terms of pairs of trailer–yard and evaluating its fitness score before moving it to the tabu list temporarily. Similar procedures were followed for the hybrid of tabu search and simulated annealing, where the action selection mechanism relied solely on a pool of solutions. It is important to note that the fitness function for all baseline models is identical to the reward function for reinforcement learning agents.

**Parameters for Baseline Models** The first algorithm, tabu search, is extensively documented in the literature concerning dispatching and routing issues. The parameter termed Tabu Tenure is set to a value of 10. This signifies that after 10 iterative steps, a solution cannot be reconsidered for selection, following its initial exploration. After this proscription period, the solution becomes eligible for reconsideration and potential selection once again. The third baseline involves a hybrid of tabu search with another popular metaheuristic algorithm, Simulated Annealing. Here, the initial temperature is set at 10, and the temperature reduction schedule rate is 0.9 for this particular problem. It is worth noting that demand-dependent trailer repositioning is rarely addressed in existing literature. Consequently, all baseline models were developed from scratch without drawing inspiration from other papers. The final baseline model, Vanilla-DQN, employs the fundamental settings associated with DQN, without any additional modifications.

**Parameters in DQN Models** Several critical parameters are associated with the DQN agent models (Both Vanilla-DQN and Rainbow-DQN). The model architecture is implemented using the PyTorch framework, which is powered by an NVIDIA GeForce RTX 3080 GPU. The model comprises five hidden layers, each with 64 nodes except for the input and output layers. The capacity for the reply buffer is fixed to be 100,000.

In the context of Rainbow-DQN, the prioritized reply buffer is configured with an $\alpha$ value of 0.7, which serves the purpose of assigning greater priority to experiences characterized by higher TD errors. This priority mechanism enhances the likelihood of these experiences being sampled during the training process. Additionally, the importance sampling weights ($\beta$) are set to 0.4, contributing to the overall prioritization scheme. Moreover, the parameter $n$ plays a crucial role in determining the number of time steps used by the agent for bootstrapping to calculate the n-step return, and it is set to 4 in this scenario. The remaining parameters retain their default settings as described in the original paper.

The exploitation and exploration mechanism is a crucial component of the agent learning process. For this paper, we employed decaying epsilon-greedy exploration. After extensive trial and error, we determined that the optimal setting involves an initial epsilon value of 0.96, which decays to 0.01. The number of epochs is set to 10000 iterations. With these details in mind, we can now delve deeper into the experimental results.

### 4.2. Rainbow-DQN training curves in Dray-Q framework

The initial experiment endeavors to determine the convergence capability of the Rainbow-DQN model in the Dray-Q framework with regard to designed reward function. Fig. 6 displays training curves for six primary scenarios.

The line chart in red illustrates the learning curve for each scenario, whereas the blue shadow line denotes the 95% confidence interval of the learning reward function. The presented confidence intervals calculation methodology utilizes a statistical approach to calculate confidence intervals for rewards in order to assess the uncertainty associated with the reward values. The method involves iteratively processing multiple scenarios and epochs, extracting rewards for each scenario, and computing the mean and standard deviation. By considering the degrees of freedom and the desired confidence level, the critical value is determined, enabling the adjustment of the confidence interval bounds. The resulting lower and upper matrices provide a comprehensive representation of the confidence intervals for each scenario and epoch. The results indicate that the Dray-Q model exhibits a strong convergence capability across all examined cases. Although the convergence slope is notably steeper up to 80 orders, the model continues to learn even with a higher count of orders. Notably, when the order count exceeds 100, the model’s convergence persists, but the reward amount level remains lower than the 80 order count, which is to be expected given the inherent complexity of the scenario. These findings suggest that the Dray-Q model performs well, even in large-scale dispatching policies.

### 4.3. Reward comparisons on Dray-Q framework vs non-RL baselines

In the present section, we conduct an evaluation of the performance of the trained Rainbow-DQN for learning Dray-Q framework dynamicity and compare it with four baseline models. The results of this evaluation are presented in Fig. 7.

Our empirical analysis consistently shows that the Rainbow-DQN model outperforms all four baseline models across all scenarios examined. Moreover, we observe that Vanilla-DQN performs better than both metaheuristic and heuristic models. Notably, the performance of metaheuristic algorithms is quite similar. In simpler scenarios with fewer orders, yards, and trailers, all models achieve comparable reward amounts with minor differences. However, in more complex scenarios, the performance gap between superior and inferior models becomes more evident.
4.4. Dray-Q framework vs non-RL baseline for delay comparison across yards

In addition to examining the reward amount as a metric for understanding and comparing the effectiveness of different models, assessing delays provides another useful method for evaluating model performance. Delays refer to the total number of times each yard experiences delays in dispatching orders because there aren’t enough empty trailers available. This is important because the main purpose of the reward function is to reduce delays by balancing empty trailer levels across yards, while also considering cost efficiency as part of the reward function.

Primarily, this experiment has two objectives. The first objective is to scrutinize the performance of the model based on independent criteria. The reward function, being an engineering criterion, is one such metric, whereas delay is a criterion that more accurately reflects model performance. The second objective is to examine how the proposed model and baselines distribute empty trailers across yards. Boxplot in Fig. 8 illustrates the distribution of delay counts across all yards over the entire time horizon for each scenario in every model.

As evidenced by the findings, the proposed framework with Rainbow-DQN agent exhibits the lowest delay levels among all models, along with a comparatively lower delay variance. As anticipated, the complexity of scenarios is positively associated with a higher number of delays. Nevertheless, the proposed model demonstrates a remarkable level of performance even under highly intricate scenarios. Conversely, the baseline models reveal a greater frequency of outliers in the boxplot.

4.5. Trailer count ablation study

Upon scenario design, a critical inquiry arises regarding the optimal determination of the number of trailers. While the number of orders originates from the customer side and the yards are fixed for extended periods, the number of trailers is a decision variable that the drayage company must select based on the available orders, neither more nor less. In practice, the number of trailers...
involved is often chosen arbitrarily, thus highlighting the need for a systematic approach to assign an optimal number of trailers for each day.

Furthermore, this experiment encompasses several other objectives. Firstly, it is aim is to evaluate the relative robustness of different models in minimizing the number of trailers within the system without sacrificing reward significantly. The number of trailers serves as a crucial decision variable in drayage operations that needs to be determined based on the number of orders and the corresponding trailer movements. Hence, the study aims to ascertain the model that exhibits superior performance in effectively reducing the number of trailers while considering these important factors. Secondly, this experiment aims to determine whether increasing the number of trailers within the system leads to greater rewards or not. The selection of the number of trailers in each
scenario is based on the company's main policy. However, there exists a possibility that slightly increasing the number of trailers in the system could result in improved reward values and, consequently, contribute to smoother operations. Thirdly, it examines whether the reward function dynamically responds to the allocation of different numbers of trailers. In essence, if there is a surplus or deficiency of empty trailers, will the reward function reflect this in the reward amount? Thus, this study explores the potential benefits of adjusting the trailer count to optimize the reward outcomes and enhance overall operational efficiency.

In order to obtain the optimal number of trailers and address the aforementioned inquiries, a sensitivity analysis is performed. For each scenario, all models were compared using five different numbers that either started or fluctuated around the primary trailer count introduced in Section 4. The ablation study results are presented in Fig. 9.

It is evident that the CUP model displays minimal sensitivity to varying numbers of trailers. This suggests that if the potential for achieving a high-level reward is present, but the model lacks the capability to do so, it becomes indifferent to the change in the number of trailers. Furthermore, the Dray-Q framework exhibits greater robustness compared to other models and, in certain instances, maintains an equal or higher reward level for lower trailer counts. For a higher number of trailers, the decrease is less significant than that of other models, indicating the model's superior ability to distribute an excess of trailers among yards. This is particularly notable for more complex scenarios.

4.6. Reward component vs delay analysis

In the context of the research outlined in Section 3.7, it is essential to dissect the influence of each component within the reward function which comprises three primary elements. Firstly, it incorporates the Laplace function to evaluate the deficit level, which is then scaled by the yard's significance vector. Secondly, it factors in the cumulative delay costs accrued by all delayed orders across all yards. Lastly, there is a mandatory component inherent to the environment’s design. The delay metric, quantifying the delay in time steps, serves as the independent variable for the experiments.

The analytical methodology utilized in this investigation revolves around evaluating the consequences of removing individual components and replacing them with alternatives. The primary metric used for assessing the impact of such substitutions is the total count of delays incurred. The concept of delay refers to the overall number of instances where each yard experiences delays in order dispatching because of a shortage of empty trailers available.

The initial component consists of two distinct elements: the Laplace function and Yard importance. Additionally, the second component, delay cost, will be subject to an independent analysis. Consequently, we will firstly replace the Laplace function with its alternative, the non-negative Laplace function. Then, we will remove the yard importance component from the reward function to observe the impact on delay occurrences. Finally, we will eliminate the Delay cost component to assess its impact compared to the original proposed reward function. Thus, three components are earmarked for scrutiny:

1. **Laplace Function**: In this part of the study, we will substitute the Laplace function with a different function termed **Non-negative Laplace**. This new function will yield the same reward as the Laplace function only when the deficit level is non-negative. This adjustment encourages the model to avoid trailer shortages. In contrast, the Laplace function incentivizes the agent to maintain a balance of available empty trailers across all yards.
Table 4
Reward Components vs total delay in yards and percentage of change compared to the proposed reward function.

<table>
<thead>
<tr>
<th>Changes in reward function,</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
<th>Scenario 5</th>
<th>Scenario 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affected component:</td>
<td>Delay count</td>
<td>% of change</td>
<td>Delay count</td>
<td>% of change</td>
<td>Delay count</td>
<td>% of change</td>
</tr>
<tr>
<td>Non-negative Laplace</td>
<td>18</td>
<td>+38%</td>
<td>33</td>
<td>+32%</td>
<td>69</td>
<td>+25%</td>
</tr>
<tr>
<td>Remove yard importance</td>
<td>13</td>
<td>+0%</td>
<td>27</td>
<td>+8%</td>
<td>56</td>
<td>+1%</td>
</tr>
<tr>
<td>Remove delay cost</td>
<td>24</td>
<td>+84%</td>
<td>44</td>
<td>+76%</td>
<td>102</td>
<td>+85%</td>
</tr>
</tbody>
</table>

(2) Yard Importance: As an alternative, the yard importance vector will be removed from this component.

(3) Delay Cost: Alternatively, delay cost will be excluded from the reward function.

The delay amounts resulting from these modifications for each component will be reported, along with the percentage of change relative to the original reward function. A positive percentage indicates an increase in the delay amount, while a negative percentage indicates a decrease. The experimental outcomes are presented in Table 4.

The analysis reveals substantial impacts on delay amounts across all yards with the removal or alteration of each component. Particularly noteworthy are the Delay Cost and the Laplace function, the latter being a crucial contribution of this study. Moreover, as scenarios grow in complexity, the difference between the original reward version and the delay amounts resulting from manipulated components widens, underscores the effectiveness of the devised reward function.

4.7. Trained rainbow-DQN reliability and generalizability analysis in Dray-Q framework

In the field of reinforcement learning, deploying a trained model in production presents a notable hurdle. Creating a separate model for every possible combination of orders, yards, and trailers is neither cost-effective nor feasible. This challenge is especially pertinent on the ordering side of operations, where daily fluctuations in orders greatly affect planning. In contrast, yards and trailers exhibit more stability and can be planned ahead by the drayage company. Therefore, any proposed model for drayage operation planning must be adaptable to order fluctuations.

Moreover, the model’s ability to manage a variable number of orders is vital for handling ad-hoc orders in drayage operations that may occur during a typical operational day. These orders could arise due to breakdowns, changes in customer order time windows, and other unforeseen circumstances. Therefore, a model capable of accommodating such uncertainties is crucial for devising new plans, even for order numbers not covered during training.

Therefore, a model must demonstrate the ability to generalize effectively beyond the trained data. In the proposed model, this generalization primarily stems from the training process, where the model encounters diverse numbers of orders as well as different time windows and pickup and delivery locations.

To assess the robustness of the proposed Dray-Q framework across varying numbers of orders, an experiment is conducted. As mentioned earlier in Section 4.1, during the training phase for each scenario, the number of orders fluctuates around the scenario’s order count following a normal distribution of \( N(\mu = \text{Order Count}, \sigma = 1) \) in each epoch. However, to test the framework’s generalization capabilities and reliability, this experiment investigates the response in reward performance across a broad range of order counts, both above and below the main trained order count. The results of this experiment are illustrated in Fig. 10, with the main order counts for each scenario highlighted in bold.

The  \( x \)-axis of the graph represents the number of orders, while the  \( y \)-axis signifies the reward amount. Our study involves six scenarios, each trained with a specific number of orders. As seen in previous experiments, each scenario achieves a distinct reward amount with its dedicated trained model, highlighted by larger, bolded points in Fig. 10. However, to assess the model’s generalizability in handling varying order sizes, we tested it with order counts both lower and higher than the main order size used during training. For instance, if Scenario 1 is trained with 50 orders, we evaluated its performance with 25, 30, 35, 40, 45, 55, 60, 65, 70, and 75 orders.

The results suggest that, for the most part, significant fluctuations in order counts have minimal effects on the reward. It is only in the two most complex scenarios that a deviation of more than five orders from the main count leads to a notable decrease. However, for ad-hoc orders, the planner can leverage simpler scenarios over time, as some orders would have already been delivered. In general, the models exhibit impressive adaptability to changes in order counts.

4.8. Runtime analysis

An important aspect of model performance is its ability to respond in real time. One of the primary limitations of MILP models or metaheuristics is their inability to respond within an acceptable timeframe. However, this limitation does not apply to trained models such as those based on reinforcement learning.

In order to evaluate the computational efficiency of the proposed model and baseline models, we performed five independent runs for each scenario. Subsequently, we calculated the average response time and identified the best time achieved to obtain an
Fig. 10. Trained Dray-Q framework reliability and generalizability test for different order counts. The primary scenario order counts are highlighted.

Table 5
Run time analysis for the proposed models in Dray-Q framework and non-RL baselines.

<table>
<thead>
<tr>
<th>Models</th>
<th>Scenario 1 Avg (s)</th>
<th>Best (s)</th>
<th>Scenario 2 Avg (s)</th>
<th>Best (s)</th>
<th>Scenario 3 Avg (s)</th>
<th>Best (s)</th>
<th>Scenario 4 Avg (s)</th>
<th>Best (s)</th>
<th>Scenario 5 Avg (s)</th>
<th>Best (s)</th>
<th>Scenario 6 Avg (s)</th>
<th>Best (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUP</td>
<td>85</td>
<td>78</td>
<td>102</td>
<td>96</td>
<td>171</td>
<td>153</td>
<td>223</td>
<td>207</td>
<td>309</td>
<td>289</td>
<td>438</td>
<td>388</td>
</tr>
<tr>
<td>Tabu search</td>
<td>620</td>
<td>540</td>
<td>1080</td>
<td>995</td>
<td>1980</td>
<td>1760</td>
<td>2460</td>
<td>2310</td>
<td>4080</td>
<td>3870</td>
<td>4800</td>
<td>4740</td>
</tr>
<tr>
<td>TS + SA</td>
<td>720</td>
<td>640</td>
<td>1380</td>
<td>1190</td>
<td>2340</td>
<td>2150</td>
<td>2820</td>
<td>2630</td>
<td>4260</td>
<td>4125</td>
<td>5460</td>
<td>5410</td>
</tr>
<tr>
<td>Vanilla-DQN</td>
<td>2.23</td>
<td>1.24</td>
<td>5.83</td>
<td>2.26</td>
<td>5.54</td>
<td>3.59</td>
<td>5.71</td>
<td>4.32</td>
<td>7.11</td>
<td>5.45</td>
<td>11.43</td>
<td>9.42</td>
</tr>
<tr>
<td>Rainbow-DQN</td>
<td>3.40</td>
<td>1.28</td>
<td>4.69</td>
<td>2.44</td>
<td>5.26</td>
<td>4.51</td>
<td>6.39</td>
<td>5.12</td>
<td>6.44</td>
<td>5.54</td>
<td>13.97</td>
<td>8.26</td>
</tr>
</tbody>
</table>

Table 6
Comparison of MILP model with Rainbow-DQN and baselines based on time and number of delay occurrence.

<table>
<thead>
<tr>
<th>Instance: O:10, T:3, Y:3</th>
<th>CPLEX</th>
<th>CUP</th>
<th>TS</th>
<th>TS + SA</th>
<th>Vanilla-DQN</th>
<th>Rainbow-DQN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>7425</td>
<td>0</td>
<td>98</td>
<td>92</td>
<td>0.8</td>
<td>1.2</td>
</tr>
<tr>
<td>Delay (s)</td>
<td>15</td>
<td>1.0</td>
<td>203</td>
<td>203</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>Time (s)</td>
<td>195</td>
<td>2.0</td>
<td>234</td>
<td>234</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>Delay (s)</td>
<td>3.0</td>
<td>1.0</td>
<td>6.0</td>
<td>6.0</td>
<td>3.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

optimal dispatching policy for a specific number of orders and trailers in each scenario. It is important to note the response times for the Vanilla-DQN and Rainbow-DQN models are based on the trained models, not the training process. The training duration of these models significantly relies on factors like computational capacity and the number of epochs. In our case, the training time for various scenarios, given the described parameters, ranges between 19 and 23 h for each scenario.

The running time are presented in Table 5. It is evident that both Vanilla-DQN and Rainbow-DQN models exhibit nearly real-time responses to various scenarios. In contrast, baseline models require significant amounts of time, rendering them unsuitable for real-world applications.

The results obtained from our study reveal that the proposed framework possesses the capability to respond in real-time, enabling it to effectively manage ad-hoc changes during the planning phase. Consequently, drayage operations companies can leverage this framework to navigate uncertainties in the dispatching procedure without incurring additional burdens.

4.9. MILP model results

In the final experiment, we sought to evaluate the performance of the MILP model alongside the Rainbow-DQN and other baseline models. Given the MILP model’s complexity outlined in Section 3.1, where it involves an NP-hard problem with three decision variables, each having three dimensions, the CPLEX solver struggles to find an exact solution within a reasonable timeframe. Furthermore, the objective function of the MILP model differs from the reward structure utilized in the Dray-Q framework and other baselines. To facilitate comparison, we opted for a smaller-scale evaluation compared to previous experiments, akin to the approach detailed in Section 4.4. Specifically, we analyzed the number of instances where delays occurred throughout the entire episode using each model for three small-scale instances involving orders, trailers, and yards. The results are presented in Table 6.

The results demonstrate that DQN-based models, particularly Rainbow-DQN, achieve comparable performance to CPLEX in nearly real-time for the three instances where results were obtainable using the CPLEX solver. This underscores the remarkable performance of the proposed Dray-Q framework, which can achieve high-level performance akin to exact methods in real-time scenarios.
5. Conclusion

In this paper, we proposed a novel demand-dependent trailer repositioning and dispatching model (Dray-Q) for drayage operations. This study aims to address three simplifications identified in prior research. Firstly, the limitation on the total number of available trailers has been considered. Secondly, the dependency of trailer movement on dynamic demand, along with the handling of ad-hoc changes in order numbers and their corresponding time windows, has been taken into account. By relaxing these simplifications, the present paper seeks to provide a more comprehensive and realistic optimal solution.

The presented framework comprises a cutting-edge agent, namely Rainbow-DQN, integrated with an extensive environment featuring a distinctive and multi-objective reward function. This design enables the agent to acquire the optimal policy even in the context of sufficiently large-scale problems, while also effectively capturing the dynamic nature of the environment.

We utilized real-world data provided by a prominent drayage operations company located in Canada to evaluate the proposed model and compare it to several baseline models, including the commonly utilized CUP, some well-known metaheuristic models, and Vanilla-DQN Model. Our experimental results showed that the proposed model in Dray-Q framework outperformed all baseline models in terms of reward amounts and minimum delay achieved, even in large-scale dispatching policies. The model’s performance remained consistent across various numbers of orders, yards, and trailers. The variations in the number of orders within each epoch and the random initialization of trailer locations in the reinforcement learning environment enhance the generalizability of the Dray-Q framework and its ability to provide policies for various numbers of orders close to the primary order count for each scenario.

The computational efficiency of the trained model within the Dray-Q framework, coupled with its robust generalizability, empowers the planner to re-execute the agent at will. This enables the acquisition of new optimal trailer dispatching policies to accommodate ad-hoc changes in the system. Moreover, the incorporation of an innovative multi-objective reward function has been shown to effectively capture the dynamic nature of delays and prevent imbalances in the availability of trailers across yards, surpassing the performance of baseline models.

Overall, our findings demonstrate the effectiveness of the proposed Dray-Q framework for trailer repositioning in drayage operations. The Dray-Q can be applied in a practical setting to optimize trailer movements and reduce delays, which can result in significant cost savings and improved customer satisfaction.

CRediT authorship contribution statement

Hadi Aghazadeh: Conceptualization, Data curation, Formal analysis, Methodology, Project administration, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. Yunli Wang: Funding acquisition, Project administration, Resources, Supervision. Sun Sun: Funding acquisition, Investigation. Xin Wang: Conceptualization, Formal analysis, Methodology, Resources, Supervision, Validation, Writing – review & editing.

Data availability

The authors do not have permission to share data.

Acknowledgments

We would like to express our gratitude to Dr. Arash Tavakoli for his invaluable guidance with regard to designing the action space. His expertise and advice were instrumental in the successful implementation of our proposed model. This project was supported in part by collaborative research funding from the National Research Council of Canada’s Artificial Intelligence for Logistics Program. The authors also would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for a part of financial support.

References
