Transportation Research Part C 156 (2023) 104355

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Synchromodal freight transport re-planning under service time uncertainty: An online model-assisted reinforcement learning

Yimeng Zhang *, Rudy R. Negenborn, Bilge Atasoy

Department of Maritime and Transport Technology, Delft University of Technology, 2628 CD Delft, The Netherlands

ARTICLE INFO

Keywords:
Synchromodal transport
Uncertainty
Deep reinforcement learning
Online transport planning
Service time

ABSTRACT

The objective of this study is to address the issue of service time uncertainty in synchromodal freight transport, which can cause delays, inefficiencies, and reduced satisfaction for shippers. The proposed solution is an online deep Reinforcement Learning (RL) approach that takes into account the service time uncertainty, assisted by an Adaptive Large Neighborhood Search (ALNS) heuristic that provides state and reward information based on the routing and scheduling. The proposed planning approach re-plans in response to unexpected events and learns from real-time information from various transport modes, including road, railway, and inland waterways. The performance of the proposed planning approach is evaluated in the European Rhine-Alpine corridor under various scenarios with different types and severities of unexpected events. The results demonstrate that the RL approach consistently outperforms other strategies by effectively handling service time uncertainty, leading to reduced costs, emissions, and waiting time, as well as decreased transport delays and improved rewards through accurate decision-making and agile transport re-planning. This study also finds that incorporating event severity information improves the average reward obtained by the RL approach in scenarios involving multiple types of events.

1. Introduction

Synchromodal freight transport is an advanced version of intermodal freight transport, which has the potential to provide low-cost, reliable, and sustainable services by utilizing multiple modes (such as road, railway, and inland waterway) in combination with real-time updates (Tavasszy et al., 2017; Li et al., 2017; Zhang et al., 2022b,a). However, synchromodal transport planning is often faced with various uncertainties, such as service time uncertainty, which can significantly impact the transportation system's efficiency (SteadieSeifi et al., 2014; Delbart et al., 2021). Service time in synchromodal transport refers to the duration of picking up, delivering, or transferring goods at terminals, including time for loading/unloading and related activities. Synchromodal transport strives for seamless and efficient transfer of goods between modes, however, the service time uncertainty at terminals caused by unexpected events poses a significant challenge to achieving this goal. Unexpected events, such as congestion, bad weather, and equipment malfunctions, can cause long waiting times or changes in the duration of service, leading to uncertainty in service time. This uncertainty can trigger delays and infeasible transport plans, causing low efficiency, high cost, and request cancellation. One necessary task of synchromodal freight transport is to adapt to service time uncertainty at terminals (including ports, train/truck stations, and transshipment terminals).

To tackle transport planning problems under uncertainty, most of the existing approaches in the literature are based on robust optimization (Abbassi et al., 2019), re-planning (Hrušovskỳ et al., 2021), or stochastic programming (Guo et al., 2021).

* Corresponding author.
E-mail addresses: Yimeng.Zhang@tudelft.nl (Y. Zhang), R.R.Negenborn@tudelft.nl (R.R. Negenborn), B.Atasoy@tudelft.nl (B. Atasoy).

https://doi.org/10.1016/j.trc.2023.104355
Received 15 February 2023; Received in revised form 4 July 2023; Accepted 13 September 2023
Available online 25 September 2023

Robust optimization evaluates the solutions using worst-case realizations of uncertain parameters and may generate unused excess capacities (Gabrel et al., 2014). Re-planning refers to adjusting transport plans and schedules in response to unexpected events. The re-planning is usually carried out after significant delays without predicting when and how long a delay will be. Stochastic programming approaches rely on prior assumptions on probability distributions for travel times or demands. They do not account for possible deviations from an assumed distribution (Farahani et al., 2021). These approaches usually assume that distribution information about the uncertainty is available a priori before an action is taken. This is not always a realistic assumption because the information about uncertainty is usually incrementally revealed during the transport process. Moreover, using historical distributions without detecting changes in an environment, the planning performance may decline (Phiboonbanakit et al., 2021). Therefore, a dynamic learning ability that updates the planning model is required for dealing with uncertainty in the environment.

Online scheduling and routing problems arise naturally in many application areas and have received increasing attention in recent years. Contrary to offline optimization, data is not assumed available a priori in online optimization. Rather it is collected during algorithm execution (Bent and Van Hentenryck, 2005). Thanks to the development of digital platforms and the rise of concepts such as synchromodal transport, a carrier is more and more able to collect real-time information from the transport network (through port authorities, terminal operators and/or sensors) about uncertainties.

The complexity and size of a transport network make it difficult for carriers to retain and learn from events. Advanced models and algorithms, specifically deep Reinforcement Learning (RL), have the potential to be instrumental in handling unexpected events. RL has been proven to be able to achieve human or superhuman skill in tasks such as Atari games (Mnih et al., 2015) and the game Go (Silver et al., 2018). In synchromodal freight transport, the pattern of service time uncertainty refers to the regularities or associations that are related to factors that have an impact on the duration of service. Such factors include but are not limited to the mode of transport, current time, terminal, and type of event. The collected information from port authorities, terminal operators, and sensors can be used to learn the pattern of uncertainties by RL. By learning online, RL can handle the uncertainty and help operators take better decisions in a re-planning framework. As opposed to traditional methods of re-planning that lack an adaptive learning component, the utilization of RL for re-planning enables learning from experience and adjusting transport plans accordingly using continuously updated policy.

It is widely recognized that RL algorithms can be challenging to implement due to the "curse of dimensionality" (Gosavi, 2009). This term refers to the difficulty of training RL agents when the dimension of the environment state or control action is high. This challenge is compounded in the context of synchromodal transport planning, which involves a large state space due to the need to consider routing and scheduling across multiple transport modes, as shown in Fig. 1. The decisions in synchromodal transport are also complex, which include both discrete actions (e.g., vehicle selection) and continuous actions (e.g., shipment scheduling). To address these challenges and enhance the convergence of RL training, we propose a model-assisted RL approach in this paper. Instead of relying solely on the RL agent to make control decisions without guidance, we integrate a transport planning model to provide assistance. Specifically, we employ Adaptive Large Neighborhood Search (ALNS) to aid the RL, which helps to reduce the size of both actions and states and thus accelerates the training process. In this way, an optimization algorithm (ALNS) that has the domain knowledge for synchromodal freight transport and a machine learning technique (RL) for unexpected events are integrated to handle the synchromodal transport re-planning under service time uncertainty. While the methodologies suggested in this study have the potential to be applicable to passenger transport, this study focuses on freight transport.

As shown in Fig. 1, there are delays in terminals due to unexpected events, such as congestion, bad weather, and equipment malfunctions, and the carriers need to take appropriate actions when such delays occur. The problem is that a carrier usually does not know how long an unexpected event will last. Therefore, a model-assisted RL approach is proposed to find suitable actions by learning from the historical experiences of all vehicles in the transport network. The proposed RL is not provided with any transportation information in advance. It learns from nothing but the state input, the reward, and the taken actions—just as a carrier in practice would. In Fig. 1, when the RL is not implemented, requests r_1 and r_2 are scheduled for transportation from terminals A and C to terminals E and G, respectively. However, unexpected events at terminals A, B, C, and D result in service time uncertainty and both requests arrive with a delay. When RL is implemented, new requests r_1' and r_2' need to be transported with the same origin and destination as r_1 and r_2. The RL adjusts the transport modes and routes based on its experience with requests r_1 and r_2. The mode of transportation between terminals B and E for request r_1' is changed from train to truck for improved speed. The route for request r_2' is altered from C-D-G to C-F-G to avoid service time uncertainty in terminal D. Both requests r_1' and r_2' are eventually delivered without delay. By using RL, our approach is able to adapt to unexpected events and make decisions in real time based on the current state of the transportation system. We demonstrate the effectiveness of our approach through a series of simulation experiments, illustrating that the approach could significantly improve the efficiency of synchromodal transport re-planning compared to traditional non-learning methods.

The main contributions of this paper are summarized as follows: (a) we introduce a synchromodal transport re-planning problem under service time uncertainty; (b) we propose a synchromodal transport re-planning framework that can accommodate different strategies; (c) under the re-planning framework, we develop a model-assisted RL approach to handle the service time uncertainty online; (d) we evaluate the performance of the proposed approach under different scenarios using a realistic transport network, including scenarios with disturbance and disruptions, scenarios with multiple types of events, and scenarios with perfect and imperfect severity levels.

The remainder of this paper is organized as follows: Section 2 presents a brief literature review. Section 3 formalizes the studied problem. Section 4 proposes the synchromodal transport planning and the model-assisted RL approach for the re-planning. In Section 5, simulation experiments and results are provided, and the ability of the approach to handle unexpected events in different scenarios is evaluated. Section 6 concludes this paper and gives future research directions.

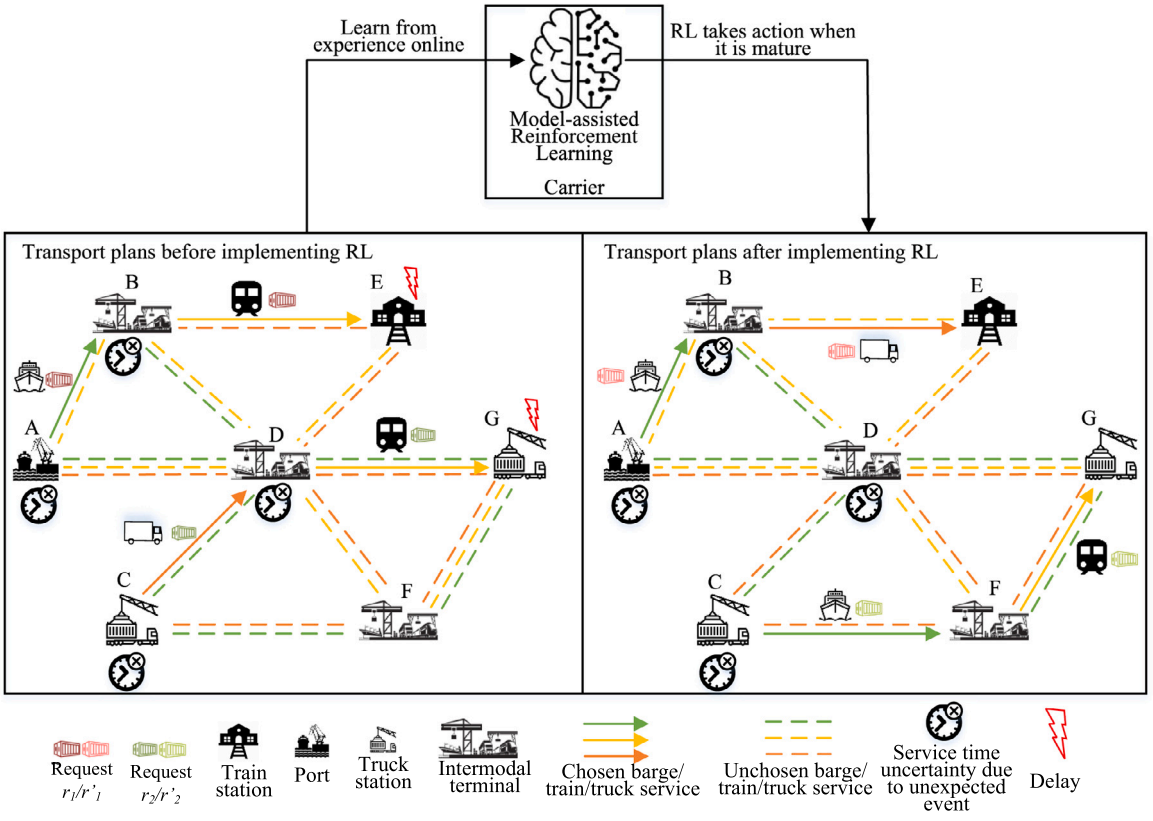


Fig. 1. Synchromodal transport planning under service time uncertainty.

2. Literature review

We first review existing studies for synchromodal transport planning under uncertainty and then review the learning approaches under uncertainty in vehicle routing problems.

2.1. Synchromodal transport planning under uncertainty

At the operational level, synchromodal transport is required to adapt to uncertainty in a dynamic environment (SteadieSeifi et al., 2014). Therefore online planning is needed based on real-time information that becomes available over time (Yee et al., 2021). In the literature, some studies do re-planning when an unexpected event occurs, while the uncertainty is not considered. For example, Van Riessen et al. (2015) measure the effect of a disturbance and update suitable paths in an intermodal transport network to adapt to occurring disturbances, such as early or late service departure and cancellation of services. Li et al. (2015) use the receding horizon intermodal container flow control approach to control and reassign intermodal container flows under disturbances in transportation demand and travel time. Qu et al. (2019) re-plan the synchromodal transport by shipment flow rerouting, service rescheduling, and transshipment when the release time, container volume, and travel time change.

Some studies consider demand uncertainty. For example, Van Riessen et al. (2016) adopt decision trees to make real-time container transport planning based on offline obtained optimal solutions. Rivera and Mes (2017) propose a look-ahead planning method for the intermodal long-haul round-trips under the uncertainty of the arrival of new orders.

Travel time uncertainty is an important issue related to the efficiency of transportation. Different approaches have been developed to handle it, including stochastic programming (Demir et al., 2016; Guo et al., 2021), Markov decision process (Yee et al., 2021), and RL (Guo et al., 2022). Most of them focus on travel time on transport arcs, including roads, railways, and waterways. In synchromodal transport, the service time uncertainty at terminals, including ports, train/truck stations, and transshipment terminals, is very common due to unexpected events, such as congestion, weather conditions, late arrivals of services, and late releases of empty containers. However, the service time uncertainty does not attract enough attention and few studies (Demir et al., 2016) propose approaches to handle it.

2.2. Learning approaches for vehicle routing problems under uncertainty

In vehicle routing problems with uncertainty, studies mainly use RL to handle the demand uncertainty. Basso et al. (2022) use RL to learn the stochastic demand and energy consumption offline for an electric vehicle routing problem. Pan and Liu (2023) design a real-time decision support system that consists of a deep neural network and an RL algorithm to control the value function of the VRP with demand uncertainty. Balaji et al. (2019) propose an RL benchmark for a VRP of an on-demand delivery driver, where orders are generated with a constant probability. Phiboonbanakit et al. (2021) use RL to discover strategies for VRP with delivery incidents and the results show that RL can quickly adapt to demand uncertainty by identifying patterns of abnormalities and rearranging shipments.

The Vehicle Routing Problem with Stochastic Travel Times (SVRP) has received considerable attention in the operations research (OR) community since its introduction by Laporte et al. (1992). Recently, the computer science (CS) community also found that RL is a potentially ideal approach to solve the SVRP, especially for dynamic SVRPs (Hildebrandt et al., 2021). The OR methodology can be used to model the SVRP with as much practical consideration as possible, such as time windows, capacity, precedence constraints, etc. The CS methodology can tackle the challenging stochastic part of the SVRP with a learning approach. In this way, a hybrid approach that combines methodologies from both OR and CS communities can provide a powerful tool to search the action space and evaluate actions in SVRP efficiently.

2.3. Summary

Table 1 provides a summary of the reviewed papers. In synchromodal/intermodal transport, only Demir et al. (2016) consider service time uncertainty, while they do not propose an online planning approach. Service time is crucial in synchromodal transport because a delay at one terminal could propagate to other terminals due to transshipment, hence causing numerous consequences, such as delays and reductions in shipper/customer satisfaction. Previous studies in synchromodal transport have focused on predefined arcs or paths, without considering the flexibility of vehicle routing (Zhang et al., 2022b). However, in this study, the ability to choose routes and switch to available vehicles under uncertainty freely is taken into account, which is a critical characteristic of synchromodal transport (Tavasszy et al., 2017; Giusti et al., 2019). In VRP, although integrating RL from CS and approaches from OR is promising, handling the uncertainties of the transport environment using RL has not been well-addressed, and existing studies mainly focus on dealing with demand uncertainty (Hildebrandt et al., 2021; Phiboonbanakit et al., 2021). Moreover, most studies require prior information, such as distribution and historical demands, while the proposed approach learns online and does not need such information.

Guo et al. (2022) is the most similar study to our work. Guo et al. (2022) use the Q-learning algorithm to learn the policy of matching a shipment with a service in synchromodal transport. There are five differences that distinguish our work from Guo et al. (2022): (a) we tackle the service time uncertainty at terminals, while Guo et al. (2022) consider the travel time uncertainty on arcs; (b) Guo et al. (2022) assume that probability distributions of uncertainties are available, while the proposed model in our work does not need the distribution to train the RL; (c) Guo et al. (2022)'s RL approach uses offline simulation to learn, while our model utilizes online learning, allowing it to adapt and improve as new information is revealed during transportation. This enables our model to better handle uncertainty in real-world scenarios; (d) Guo et al. (2022) use a tabular Q-learning approach and the obtained policy cannot be generalized to events that have never been encountered before, while our model can handle events with similar features by using a deep neural network as a function approximator to estimate the action-value function; (e) our study proposes a model-assisted RL, which integrates a heuristic with RL to let RL only focus on the uncertainty part, and the size of the state is reduced compared to Guo et al. (2022).

3. Problem description

The notation used in the mathematical model is provided in Table 2. We consider a transport network $G = (N, A)$ (N/A represents the set of nodes/arcs) with multiple modes $w \in W$, and the carrier transports containers in the network. The nodes (terminals) N include ports, train/truck stations, and transshipment terminals. The arcs A include waterway, railway, and road. A carrier usually owns multiple vehicles $k \in K$ and serves multiple requests $r \in R$. The pickup and delivery terminals of request $r \in R$ are designated by $p(r)$ and $d(r)$, respectively. A request r needs to be picked up in time window $[a_{p(r)}, b_{p(r)}]$ at terminal $p(r)$ and delivered in time window $[a_{d(r)}, b_{d(r)}]$ at terminal $d(r)$. Let $o(k) \in O \subseteq N$ and $o'(k) \in O \subseteq N$ represent the starting and the ending depot of vehicle k. Request r can be served by a single vehicle k or by multiple vehicles $k_1, k_2, \ldots \in K$ via transshipment terminals $T \subseteq N$.

During the transportation, unexpected events ue may occur with starting time t_{ue} and ending time $\bar{t}_{ue}$. The starting and ending times are unknown when designing the initial transportation plan. Due to unexpected events, the duration of service time at each terminal $i \in N$ is uncertain. If a vehicle k arrives at terminal i and cannot transport request r as planned, the request r is an affected request. If the vehicle k continued as originally planned, the delivery time of request r could exceed $b_{d(r)}$ and a delay penalty will be charged. To avoid delay, the best action needs to be taken. This study addresses the following research question: How does an agent learn online to plan better under service time uncertainty in the context of synchromodal transportation? Specifically, the following questions need to be considered:

1. Should the affected requests be served by the current vehicle?
2. If not, which vehicles are suitable for serving them?

Table 1
Comparison between the proposed model and existing approaches in the literature.

Article	Problem characteristics					Methodologies			
	Mode	Problem	Vehicle routing	Uncertainty	Event location	Approach	Learning	Re-planning	Required prior information
Synchromodal transport									
Li et al. (2015)	road, railway, inland waterway	STP		–	–	RHC	–	periodical	–
Van Riessen et al. (2016)	road, railway, inland waterway	STP		demand	–	DT	✓, offline	real-time	historical requests
Demir et al. (2016)	road, railway, inland waterway	STP		travel/service time, demand	arcs	SO		–	–
Rivera and Mes (2017)	road, inland waterway	STP		demand	–	MDP		periodical	distribution
Yee et al. (2021)	road, railway, inland waterway	STP		travel time	arcs	MDP		periodical	distribution
Qu et al. (2019)	road, railway, inland waterway	STP		–	–	LP		real-time	–
Guo et al. (2021)	road, railway, inland waterway	STP		travel time and demand	arcs	SO		periodical	distribution
Guo et al. (2022)	road, railway, inland waterway	STP		travel time	arcs	RL	✓, offline	periodical	distribution
Vehicle routing problems									
Balaji et al. (2019)	road	VRP	✓	demand	–	DRL	✓, online	–	none
Pan and Liu (2023)	road	VRP	✓	demand	–	DRL	✓, online	real-time	none
Basso et al. (2022)	road	VRP	✓	demand	–	RL	✓, offline	real-time	historical data
This research	road, railway, inland waterway	STP	✓	service time	terminals	model-assisted DRL	✓, online	real-time	none

–: in the "Required prior information" column, it means that no information is required as uncertainty is not taken into account; in other columns, it means that the relevant item is not mentioned in the article.
RHC: Receding horizon control; STP: Synchromodal transport planning; VRP: Vehicle Routing Problem; MDP: Markov decision process; DT: Decision trees; LP: Linear programming model; SO: Stochastic optimization.

In question 1, if request r is served by one vehicle, only the schedule of the current vehicle needs to be evaluated. If two or more vehicles are used, the schedules of subsequent vehicles also need to be considered. If the request is removed from the schedule of a vehicle, then question 2 is considered. After inserting the removed request into a new route of vehicle k', the schedules of vehicle k' and vehicles that have transshipment operations with k' need to be re-evaluated. Since multiple requests could be influenced by the same unexpected event, the above re-evaluation needs to be iterated until all affected requests have either been confirmed to keep the original plan or have been re-planned.

The severity of unexpected events may differ. Some events may cause severe disruptions and some may only disturb the schedules of vehicles. For different terminals, the factors that influence the duration of unexpected events are various, such as weather conditions, equipment malfunctions, or traffic congestion. Therefore, the duration of unexpected events at different terminals may be of different types. Multiple events of different types may happen in a single terminal. Moreover, the severity level of the event may be provided by the port authority or terminal operator (for example based on the source of the event), and the severity level may be inaccurate. The performance of the model under severity level with inaccurate information needs to be evaluated. Therefore, different scenarios need to be considered to evaluate the effectiveness of the proposed approach.

4. Proposed planning approach

Solving vehicle routing problems by RL is challenging because the size of the state is very large, especially for synchromodal transport with multiple modes and transshipment (Guo et al., 2022). RL can be computationally expensive and may not always find the optimal solution, especially in such large and complex environments. Different from approaches that solely use RL to solve vehicle routing problems (Nazari et al., 2018; James et al., 2019; SteadieSeifi et al., 2021), this study integrates RL and ALNS to make use of the strengths of both, namely the data-driven strength of the former and the domain knowledge from the latter, as shown in Fig. 2. ALNS is a metaheuristic optimization algorithm that is widely used to solve vehicle routing problems, and the ALNS used in this study is extended from our previous work (Zhang et al., 2022b). It works by iteratively constructing and improving solutions by making changes to the current solution, called "neighborhood moves". ALNS can provide efficient search and optimization capabilities for the static problem, while RL can provide real-time adaptability and decision-making capabilities under uncertainty. This can potentially allow the integrated approach to find good solutions quickly and adapt to unexpected events in real-time. Different from the ALNS in our previous paper (Zhang et al., 2022b), in this study, ALNS is used to build schedules, provide information on states, check feasibility to provide rewards and guide the RL agent by prioritizing vehicles. Benefiting from

Table 2
Notation.

Sets:	
W	Set of modes indexed by w.
R	Set of requests indexed by r.
N	Set of terminals indexed by i and j. $O \subseteq N$, set of depots. $P/D/T \subseteq N$, set of pickup/delivery/transshipment terminals. $T_{w_1}^{w_2}$, set of terminals that allows transshipment between mode w_1 and mode w_2.
K	Set of vehicles indexed by k and l. $K_{\text{b\&t}} \subseteq K$, set of barges and trains. $K_{\text{truck}} \subseteq K$, set of truck fleets. $K_w \subseteq K$, set of vehicles of mode w. $K_{\text{fix}} \subseteq K$, set of vehicles that have predefined routes and schedules. $K_{ue} \subseteq K$, set of vehicles that affected by unexpected event ue.
A	Set of arcs. For $i, j \in N$, the arc from i to j is denoted by $(i, j) \in A$. $A_p/A_d \subseteq A$ represents the set of pickup/delivery arcs. For $(i, j) \in A_p$, $i \in P$. For $(i, j) \in A_d$, $j \in D$. $A_w \subseteq A$ represents the set of arcs for mode w. $A_{\text{fix}}^k \subseteq A$ represents the set of arcs for a fixed vehicle $k \in K_{\text{fix}}$.
Parameters:	
u_k	Capacity (TEU) of vehicle k.
q_r	Quantity (TEU) of request r.
τ_{ij}^k	The travel time (in hours) on arc (i, j) for vehicle k.
$[a_{p(r)}, b_{p(r)}]$	The pickup time window for request r.
$[a_{d(r)}, b_{d(r)}]$	The delivery time window for request r.
$[a_i^k, b_i^k]$	The open time window for fixed vehicle k at terminal i.
t''^k_i	The loading (or unloading) time (in hours) for vehicle k at terminal i.
v_k	Speed (km/h) of vehicle k.
d_{ij}^k	Distance (km) between terminals i and j for vehicle k.
e_k	CO_2 emissions (kg) per container per km of vehicle k.
c_k^n	$c_k^1/c_k^{1'}$ is transport cost (euro) per hour/km per container using vehicle k. c_k^2 is the loading (or unloading) cost per container. c_k^3 is the storage cost per container per hour. c_k^4 is the carbon tax coefficient per ton. c_k^5 is the cost per hour of waiting time.
c_r^{delay}	The delay penalty per container per hour of request r.
M	A large enough positive number.
Variables:	
x_{ij}^k	Binary variable; 1 if vehicle k uses the arc (i, j), 0 otherwise.
y_{ij}^{kr}	Binary variable; 1 if request r transported by vehicle k uses arc (i, j), 0 otherwise.
z_{ij}^k	Binary variable; 1 if terminal i precedes (not necessarily immediately) terminal j in the route of vehicle k, 0 otherwise.
s_{ir}^{kl}	Binary variable; 1 if request r is transferred from vehicle k to vehicle $l \neq k$ at transshipment terminal i, 0 otherwise.
$t_i^{kr}/t'^{kr}_i/\bar{t}_i^{kr}$	The arrival time/service start time/service finish time of request r served by vehicle k at terminal i.
$t_i^k/t'^k_i/\bar{t}_i^k$	The arrival time/last service start time/departure time of vehicle k at terminal i.
t_{ki}^{wait}	The waiting time of vehicle k at terminal i.
t_r^{delay}	The delay time of request r at delivery terminal.

combining ALNS, the RL approach can focus on the uncertainty in real-time and the size of the state in RL can be reduced by only keeping critical factors that influence decisions.

Section 4.1 presents the mathematical model for synchromodal transport planning. Section 4.2 introduces a re-planning framework that considers different strategies, one of which is the model-assisted RL. Section 4.3 presents the details of the model-assisted RL.

4.1. Synchromodal transport planning

The mathematical model for synchromodal transport planning is extended from Zhang et al. (2022b). The objective of synchromodal transport planning is minimizing cost (Euros), which consists of transit cost (F_1), transfer cost (F_2), storage cost (F_3), carbon tax (F_4), waiting cost (F_5), and delay penalty (F_6), as shown in Eqs. (1)–(7).

$$\min\ F = F_1 + F_2 + F_3 + F_4 + F_5 + F_6 \tag{1}$$

$$F_1 = \sum_{k\in K}\sum_{(i,j)\in A}\sum_{r\in R}(c_k^1\tau_{ij}^k + c_k^{1'}d_{ij}^k)q_r y_{ij}^{kr} \tag{2}$$

$$F_2 = \sum_{k,l\in K,k\neq l}\sum_{r\in R}\sum_{i\in T}(c_k^2 + c_l^2)q_r s_{ir}^{kl} + \sum_{k\in K}\sum_{(i,j)\in A_p}\sum_{r\in R}c_k^2 q_r y_{ij}^{kr} + \sum_{k\in K}\sum_{(i,j)\in A_d}\sum_{r\in R}c_k^2 q_r y_{ij}^{kr} \tag{3}$$

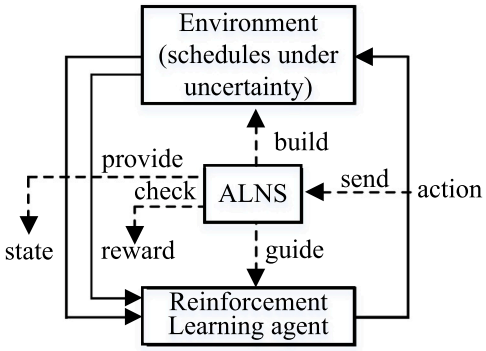


Fig. 2. Model-assisted RL.

$$F_3 = \sum_{k,l\in K,k\neq l}\sum_{r\in R}\sum_{i\in T} c_k^3 q_r s_{ir}^{kl}(t'^{lr}_i - \bar{t}_i^{kr}) + \sum_{k\in K}\sum_{(i,j)\in A_p}\sum_{r\in R} c_k^3 q_r y_{ij}^{kr}(t'^{kr}_i - a_{p(r)}) \tag{4}$$

$$F_4 = \sum_{k\in K}\sum_{(i,j)\in A}\sum_{r\in R} c_k^4 e_k q_r d_{ij}^k y_{ij}^{kr} \tag{5}$$

$$F_5 = \sum_{k\in K}\sum_{i\in N} c_k^5 t_{ki}^{\text{wait}} \tag{6}$$

$$F_6 = \sum_{r\in R} c_r^{\text{delay}} q_r t_r^{\text{delay}} \tag{7}$$

Constraints (8)–(15) are temporal constraints at terminals. Other constraints, e.g., temporal constraints on arcs, spatial constraints, capacity constraints, etc., are presented in Appendix A.

$$t_i^{kr} \leqslant t'^{kr}_i \quad \forall i \in N,\ \forall k \in K,\ \forall r \in R \tag{8}$$

$$t'^{kr}_i + t''^{kr}_i \sum_{j\in N} y_{ij}^{kr} \leqslant \bar{t}_i^{kr} \quad \forall i \in N,\ \forall k \in K,\ \forall r \in R \tag{9}$$

$$t_i^k \leqslant t_i^{kr} \quad \forall i \in N,\ \forall k \in K,\ \forall r \in R \tag{10}$$

$$t'^{k}_i \geqslant t'^{kr}_i \quad \forall i \in N,\ \forall k \in K,\ \forall r \in R \tag{11}$$

$$\bar{t}_i^k \geqslant \bar{t}_i^{kr} \quad \forall i \in N,\ \forall k \in K,\ \forall r \in R \tag{12}$$

Constraints (8) and (9) are time constraints on service start and finish time, respectively. Constraints (10), (11), and (12) take care of the vehicle's arrival, service, and departure time, respectively.

$$\bar{t}_i^{kr} - t'^{lr}_i \leqslant M(1 - s_{ir}^{kl}) \quad \forall r \in R,\ \forall i \in T,\ \forall k,l \in K,\ k \neq l \tag{13}$$

$$t_{ki}^{\text{wait}} \geqslant t'^{k}_i - t_i^k \quad \forall i \in N,\ \forall k \in K \tag{14}$$

$$t_r^{\text{delay}} \geqslant (\bar{t}_{d(r)}^{kr} - b_{d(r)}) \sum_{i\in N} y_{id(r)}^{kr} \quad \forall r \in R,\ \forall k \in K \tag{15}$$

Constraints (13) are time constraints for transshipment. If there is a transshipment from vehicle k to vehicle l, but vehicle l arrives before vehicle k departs, vehicle l can wait until vehicle k completes its unloading. Constraints (14) and (15) calculate waiting time and delay time, respectively.

4.2. Synchromodal transport re-planning framework

This section presents a re-planning framework that accommodates different strategies: (a) waiting strategy, (b) average duration strategy, and (c) model-assisted RL strategy. When an unexpected event ue occurs prior to the planned service start time ($t_{ue} < t'^{kr}_i$), the service should start when the unexpected event ue is resolved, as shown in Constraints (16):

$$t'^{kr}_i \geqslant \bar{t}_{ue} \quad \forall i \in N,\ \forall k \in K_{ue},\ \forall r \in R \tag{16}$$

However, the event ending time $\bar{t}_{ue}$ in Constraints (16) is uncertain, which influences requests served by vehicle k at terminal i. If appropriate action is not taken, it may cause a long waiting time t_{ki}^{wait} at terminal i and hence severe delay t_r^{delay} at the delivery terminal $d(r)$.

With an event-triggered mechanism, the framework consists of two phases: re-planning when the unexpected event occurs at time step t_{ue} and evaluation/learning when the unexpected event ends at time step $\bar{t}_{ue}$. In order to address the two questions outlined in Section 3, the re-planning phase contains two sub-phases: the removal phase and the insertion phase. Three strategies (a), (b), or (c) are employed to determine the actions in these sub-phases. In the removal phase, the actions are to either wait or remove a request from the vehicle's schedule. In the insertion phase, the actions are to either insert a request into the schedule or not insert it.

As presented in Algorithm 1, in strategy (a), all vehicles just wait during the unexpected event mimicking the traditional planning in practice. When an unexpected event finishes, if there is a delay and re-planning is possible, the affected request will be re-planned. As presented in Algorithm 2, strategy (b) collects the duration of unexpected events online and then assumes that the current expected event's duration equals the average duration of these records and is updated as more information is received. This strategy mimics carriers who also learn from experience, but in a simpler way compared to RL.

Strategy (c) integrates re-planning and learning using a model-assisted RL, as presented in Algorithm 3. The re-planning phase stores information about the situation at time step t_{ue}. Since RL cannot immediately receive a reward for its actions when events occur, we allow the RL agent to learn when events end. As illustrated in Fig. 3, when an event begins, the situations faced by all vehicles at different terminals are stored. When the event ends, RL is trained by simulating the situation when the event occurred. Since the duration of the event is known at the end, the reward for the taken action can be calculated. During the learning phase, the re-planning process relies on ALNS. For strategy (c), we also consider RL with and without severity level in the state.

Detailed procedures of these three strategies are introduced in Sections Section 4.2.1, Section 4.2.2, and Section 4.2.3.

Algorithm 1: Waiting strategy

Input: K, R, G; **Output:** X, R_{pool}; // X/R_{pool} represents the solution/request pool.
obtain the initial routes by the static ALNS's solution for the mathematical model in Section 4.1 (Zhang et al., 2022b);
if *an event ue finishes* **then**
  get affected requests R_{ue};
  add the event duration to schedules of R_{ue} and check feasibility of R_{ue}'s schedules;
  if there are delays for requests R_{ue} and re-planning is possible, re-plan using Algorithms 4 and 5 and update solution X
end

Algorithm 2: Average duration strategy

Input: K, R, G; **Output:** X, R_{pool}; // X/R_{pool} represents the solution/request pool.
obtain the initial routes by the static ALNS's solution for the mathematical model in Section 4.1 (Zhang et al., 2022b);
set the average duration $\overline{DU}$ as zero;
if *an event ue occurs and enough historical durations are collected* **then**
  get affected requests R_{ue};
  for *r in* R_{ue} **do**
    add average duration $\overline{DU}$ to *r*'s schedule;
    check feasibility of request *r*'s schedule;
    if *r's schedule is infeasible* **then**
      remove and reinsert request *r* using Algorithms 4 and 5 and update solution X
    end
  end
end
if *an event ue finishes* **then**
  the average duration $\overline{DU}$ is updated with the event duration;
  get affected requests R_{ue};
  add the event duration to schedules of R_{ue} and check feasibility of R_{ue}'s schedules;
  if there are delays for requests R_{ue} and re-planning is possible, re-plan using Algorithms 4 and 5 and update solution X;
end

4.2.1. Re-planning when the unexpected event occurs

The initial solution for the synchromodal transport re-planning is generated using ALNS (Zhang et al., 2022b). However, unexpected events during transportation may require the initial solution to be modified through re-planning. The synchromodal transport re-planning process proceeds as follows:

1. Find affected requests R_{ue}: When an unexpected event ue occurs at terminal i at time t_{ue}, the first step is to determine which transport mode w_{ue} is affected by the event. Then, for each vehicle k in the set $K_{w_{ue}}$, the process checks whether the vehicle passes terminal i. If it does, the process identifies all requests R_k^i that have operations at i. For each request r in R_k^i, if the planned service start time is larger than the event occurring time ($t'^{kr}_i > t_{ue}$), then the request is added to the set of requests R_{ue} that are affected by the unexpected event.

Algorithm 3: RL strategy

Input: K, R, G; **Output:** X, R_{pool}; // X/R_{pool} represents the solution/request pool.
obtain the initial routes by the static ALNS's solution for the mathematical model in Section 4.1 (Zhang et al., 2022b);
if *an event ue occurs and RL is mature* **then**
  get affected requests R_{ue};
  for r *in* R_{ue} **do**
    send the state of request r to RL and obtain the action from RL;
    if *the action is removal* **then**
      remove request/segment r using Algorithms 4 and update solution X;
      for k *in suitable vehicles* K' *for request/segment* r **do**
        try to insert r to vehicle k using Algorithm 5; send the state of k and r to RL and receive action from RL;
        if *the action is insertion* **then**
          keep the insertion, update solution X, and break the loop;
        end
      end
    end
  end
end
if *an event ue finishes* **then**
  update the RL's policy using the approach in Section 4.3;
  get affected requests R_{ue};
  add the event duration to schedules of R_{ue} and check feasibility of R_{ue}'s schedules;
  if there are delays for requests R_{ue} and re-planning is possible, re-plan using Algorithms 4 and 5 and update solution X;
end

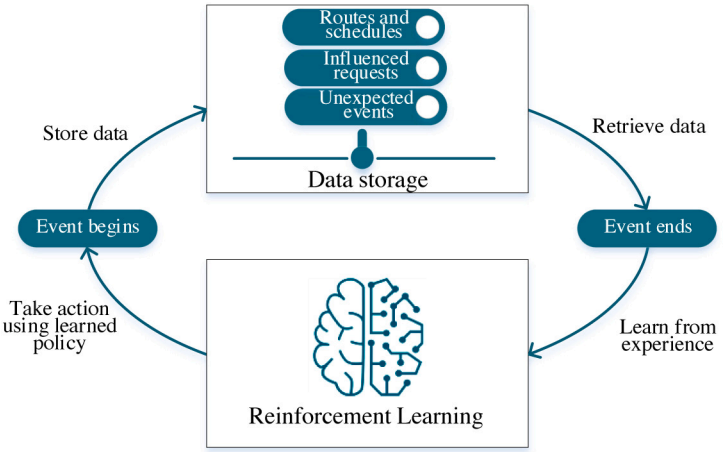


Fig. 3. Data storage and learning.

2. Collect RL state information: If the RL strategy is being used, for each request r in R_{ue}, the process collects the state information in Section 4.3.
3. Take action: For strategy (a), the waiting action is always taken when an unexpected event occurs and the schedules are not changed. For strategy (b), the average duration is added to the schedules, and the feasibility is checked. If the request is infeasible, it is removed and re-inserted using Algorithms 4 and 5 in Section 4.2.3. For strategy (c), RL is used to make decisions (see details in Section 4.3). The ALNS sends the state to RL and waits for the action from RL. If the action is waiting, the process evaluates the next affected request. If the action is removal, the process uses Algorithms 4 and 5 to remove and insert the request/request segment, respectively. When strategies (b) and (c) are not implemented, waiting action is taken.
4. The above steps are repeated until all requests have been delivered.

4.2.2. *Evaluation/learning when the unexpected event finishes*

When unexpected event *ue* finishes, the duration is known. The RL agent can then learn from the experience of affected requests R_{ue} during the event. All routes, unserved requests, and state s_t are retrieved from the data storage to simulate the same situation

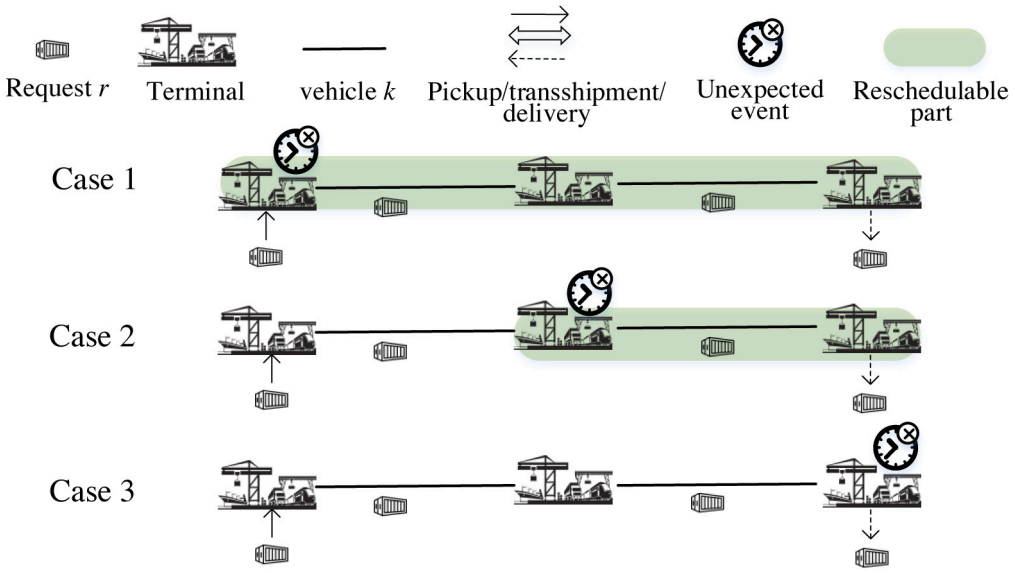


Fig. 4. Reschedulable part when there is no transshipment.

at time step t. The ALNS sends the state s_t to RL and RL gives the action. Similar to the procedures in Section 4.2.1, if the action is removal, the request/request segment is removed and re-inserted using Algorithms 4 and 5 in Section 4.2.3. If the action is non-removal, the vehicle will wait until the event finishes. After the action is taken, the reward is determined by the methods in Section 4.3 and sent to RL for learning.

If RL is implemented, the performance of the action a_t taken by RL is evaluated. For each affected request r in R_{ue}, the routes, removal action a_t and insertion action a'_t (if any) at time step t are restored to recreate the same situation. The reward is then determined by checking the feasibility after taking the action a_t using the approaches in Section 4.3.

If strategy (b) is used, the duration is collected. The performance of strategies (a) and (b) are evaluated in a similar way to the evaluation of RL.

4.2.3. Removal and insertion methods

There are two types of synchromodal transport re-planning: re-planning for requests with and without transshipment. Requests without transshipment involve the transportation of goods using only one mode of transportation. If an unexpected event occurs at a terminal along the route of a vehicle transporting such a request, the vehicle may need to wait at the terminal until the event is resolved. This can cause delays in the delivery of the goods and may result in decreased efficiency and increased costs. In this case, re-planning may involve adjusting the route or waiting at the terminal until the event is resolved, depending on the specific situation. Requests with transshipment in synchromodal transport involve the transfer of goods from one mode of transportation to another at a specific terminal. If an unexpected event occurs at the transshipment terminal, it may affect the availability of the next mode of transportation or the transfer of goods between modes. Therefore, the re-planning for requests with transshipment also needs to consider the case where only a segment of the request is affected, meaning that only the request segment after the transshipment terminal needs to be re-planned. This helps to minimize the impact of the re-planning on the initial plan. The cases of these two types are shown in Figs. 4 and 5, respectively.

The key to successful re-planning is to identify the current location i_k of the vehicle k and the terminal i_{ue} with the unexpected event ue, and determine the possible actions. If i_k is a terminal after i_{ue}, vehicle k is not affected by the event ue. Except in the case where ue occurs at the delivery terminal, only the re-planning from i_{ue} is considered in order to minimize changes to the initial plan.

On the route of vehicle k, if i_{ue} is a previous terminal of the pickup terminal $p(r)$ or is $p(r)$, then it is case 1 in Fig. 4. In this case, the entire request r will be removed to R_{pool} and re-planned. For case 2, i_{ue} is in the middle of $p(r)$ and $d(r)$, and the request can be segmented by i_{ue} and delivery time $t^r_{i_{ue}}$ at i_{ue} if the request cannot be delivered on time. This results in the request r being segmented into two segments: $r^1_{i_{ue}}$, which needs to be picked up in the time window $[a_{p(r)}, b_{p(r)}]$ at terminal $p(r)$ and delivered in the time window $[a_{p(r)}, t^r_{i_{ue}}]$ at transshipment terminal i_{ue}, and $r^2_{i_{ue}}$, which needs to be picked up in the time window $[t^r_{i_{ue}}, b_{d(r)}]$ and delivered in the time window $[a_{d(r)}, b_{d(r)}]$. The planning for $r^1_{i_{ue}}$ remains unchanged, while $r^2_{i_{ue}}$ is removed to R_{pool} and re-inserted. In case 3, the unexpected event occurs at the delivery terminal $d(r)$. If i_k is a previous terminal of $d(r)$, the request can be removed or segmented by i_k in a similar way as in cases 1 and 2. Otherwise, the request cannot be rescheduled.

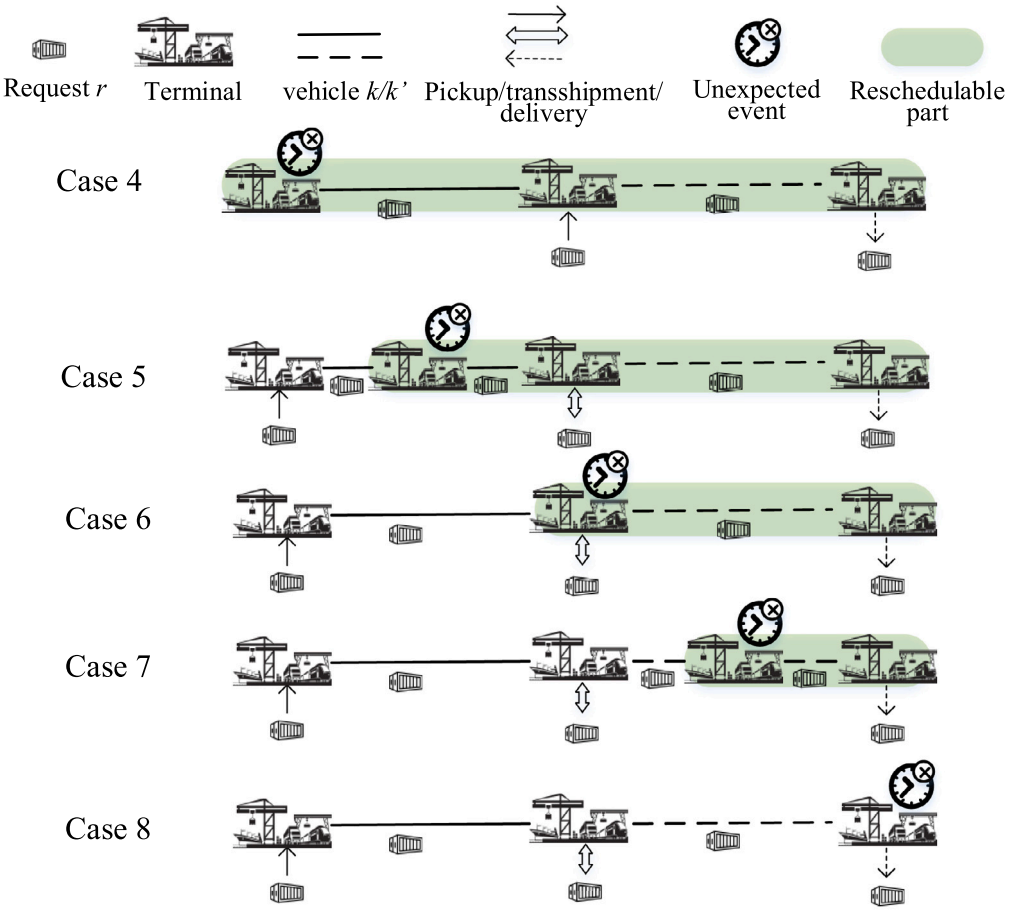


Fig. 5. Reschedulable part when there is transshipment.

There are five cases to consider when request r is transferred at transshipment terminal j, as shown in Fig. 5. In case 4, similar to case 1, the unexpected event influences the entire transportation of request r, and the request can be fully removed and re-planned. Cases 5 and 6 are similar to case 2, with the unexpected event occurring at a terminal between $p(r)$ and j (case 5) or at transshipment terminal j (case 6), resulting in the request segment from the affected terminal to $d(r)$ being removable. In case 7, the unexpected event occurs at a terminal between j and $d(r)$, and the request segment from the affected terminal to $d(r)$ can be re-planned, potentially requiring the use of three or more vehicles to serve the request. In case 8, the unexpected event occurs at the delivery terminal, and the request can be removed or segmented as in cases 4-7, depending on the location of vehicle k.

The removal and insertion algorithms are presented in Algorithms 4 and 5. Algorithm 4 removes the request or request segment based on the case it belongs to, while Algorithm 5 inserts the request into a route until the feasibility, as evaluated by ALNS/RL, is achieved.

4.3. Model-assisted reinforcement learning

The RL agent interacts with an environment $\mathcal{E}$ at each of a sequence of discrete time steps, $t = 0, 1, 2, 3, \ldots$. Besides the planning of all vehicles and requests, $\mathcal{E}$ contains the uncertain duration of the service time at terminals. For the unexpected event ue occurring at each time step t_{ue}, the RL agent receives a state s_t and chooses an action a_t from a set of possible actions $\mathcal{A}$ according to its policy $\pi = P(a_t|s_t)$. When the event finishes, the actual duration is known. ALNS checks the feasibility of schedules after adding the duration and gives the RL agent a scalar reward r_t. The goal of the RL agent is to maximize cumulative rewards R_t by selecting appropriate actions from each state s_t:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{17}$$

where γ is a discount factor.

Algorithm 4: Removal algorithm

Input: K, r, X_{current}, R_{pool}, *case*; **Output:** X_{removal}, R_{pool}; // $X_{\text{current}}/X_{\text{removal}}$ means the current solution/the solution after removal.
if *case == 1 or case == 4* **then**
  for *k in K* **do**
    if *k serves r* **then**
      remove r from k's schedule in X_{current} and obtain X_{removal};
    end
  end
  add r to R_{pool};
else
  remove the request segment from related vehicles and obtain X_{removal};
  add the request segment to R_{pool}.
end

Algorithm 5: Insertion algorithm

Input: k, r, X_{current}, R_{pool} ; // k is the vehicle that is trying to be used, and r could be a request or a request segment.
Output: $X_{\text{insertion}}$, R_{pool}; // $X_{\text{current}}/X_{\text{insertion}}$ means the current solution/the solution after insertion.
for *position pos in all possible positions in k's route in* $X_{current}$ **do**
  insert r to the position *pos*;
  check feasibility of the inserted route by the ALNS or RL;
  if *the solution after insertion is feasible* **then**
    keep the insertion and obtain $X_{\text{insertion}}$;
    remove r from R_{pool};
    stop;
  else
    remove r from position *pos*.
  end
end

In this RL approach, the state includes important information about the current time t, passed terminals that have unexpected events N_{ue}, the travel time τ_{ij}^{k} between adjacent terminals $i, j \in N_{ue}$, and the delay tolerance $t_r^{\text{tolerance}}$ of the request. The current time t helps the RL agent to evaluate how long the unexpected event will last. The decision for one request must consider not only the unexpected event at the current terminal but also events at later terminals. Therefore, we include all passed terminals that have unexpected events N_{ue} and the travel time τ_{ij}^{k} in the state. The delay tolerance $t_r^{\text{tolerance}}$ represents the maximum possible delay time and should not be smaller than the duration of the unexpected event, otherwise, there will be a delay in delivering the request and the request may need to be switched to another vehicle. As shown in Fig. 6, there are two cases when calculating the delay tolerance. In both cases, the delay tolerance includes the duration between the latest delivery time and the planned delivery time $b_{d(r)} - t_r^{\text{delivery}}$. In case 1, the event begins after the service start time t'^{kr}_i, so the delayed time is equal to the duration of the event. In case 2, the event begins before t'^{kr}_i, and part of the duration $t'^{kr}_i - t_{ue}$ does not affect the service, which needs to be added to the delay tolerance. Therefore, the delay tolerance is calculated by:

$$t_r^{\text{tolerance}} = (t'^{kr}_i - t_{ue})^{+} + (b_{d(r)} - t_r^{\text{delivery}}). \tag{18}$$

In order to provide more information about the event that is causing the service time uncertainty, we also consider the severity level of the event as a part of the state in the RL approach. The severity level is a measure of the impact of the event on the transport operation and can be obtained from the terminal operator or other sources. The severity level may not always be accurate due to incomplete information and measurement errors, therefore, the consideration of imperfect severity level is also incorporated. The inclusion of the severity level is only applied in complex scenarios, as demonstrated in Section 5.2.

Fig. 7 shows the flowchart of the model-assisted RL. When a request is influenced, firstly the RL agent decides whether the current vehicle is suitable to serve it or not in the removal phase. The first step involves generating the route for the request and sending the relevant state information to the RL agent. The RL agent then makes a decision about whether the request should be removed or if the vehicle should wait until the event finishes. ALNS then evaluates the feasibility of the original route based on the action taken by the RL agent and sends a reward to the RL agent based on whether or not the action avoided delay. If the action is removal and there is a delay when the event finished, or if the action is waiting and there is no delay when the event finishes, the reward is 1. Otherwise, the reward is 0.

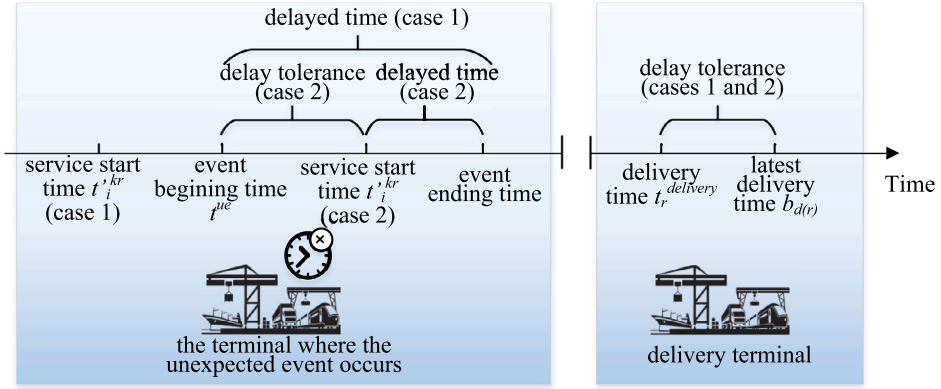


Fig. 6. Delay tolerance in the state.

If the action in the removal phase is removal, RL will determine which vehicle is the most suitable for the affected request in the insertion phase. The insertion phase includes the following steps:

1. ALNS ranks the vehicles based on their unit cost. For each vehicle, it inserts the removed request into the route using the ALNS greedy insertion operator (Zhang et al., 2022b), and then sends the resulting state to RL. RL then returns an action, which can be either non-insertion (1) or insertion (0).
2. ALNS evaluates the action by checking the feasibility of the original route and sends a reward to RL. If the action is non-insertion (1) and there is a delay, or if the action is insertion (0) and there is no delay, the reward is 1. Otherwise, the reward is 0.
3. If the action is an insertion, the insertion phase is stopped and the affected request is inserted into the chosen vehicle. If the action is non-insertion, the process continues with the next vehicle until the request is inserted or there is no suitable vehicle left.

The actions and rewards in the insertion phase have a similar meaning to those in the removal phase but are named differently. Therefore, the same RL approach can be utilized for both phases. If the action in the removal phase is waiting, then the insertion phase is not necessary. If the action is removal, it may be necessary to perform additional iterations in the insertion phase to identify a suitable vehicle.

Once the RL approach has reached a certain level of maturity or a predetermined number of iterations, it will be used to make decisions for requests that are affected by uncertainty, while the ALNS heuristic will continue to handle constraint checking. As the RL approach continues to interact with the environment, it will continue to learn and improve its decision-making capabilities.

The proposed model-assisted RL framework can be built upon any RL algorithm. In this paper, we use the deep Q-network (DQN) (Mnih et al., 2015), a representative RL technique, as the RL algorithm. The action value $Q^{\pi}(s,a) = \mathbb{E}[R_t|s_t = s, a]$ is the expected return for selecting action a in state s and following policy π. The optimal value function $Q^*(s,a)$ gives the maximum action value for state s and action a for any policy. The $Q^*(s,a)$ obeys the Bellman equation:

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a')|s,a]. \tag{19}$$

where s' and a' are the state and action at the next time step. The Bellman equation means that the optimal policy is to choose the action maximizing the expected value of $r+\gamma \max_{a'} Q^*(s',a')$ if $Q^*(s',a')$ of s' is known for all possible actions a'. In practice, finding $Q^*(s,a)$ is computationally expensive. Therefore, the DQN uses deep neural networks, called Q-network, as a nonlinear function approximator with parameters θ to estimate the action value function: $Q(s,a;\theta) \approx Q^*(s,a)$.

The algorithm for training DQN to approximate the optimal action-value function $Q^*(s,a)$ is presented in Algorithm 6. A target network $\hat{Q}(s,a;\theta^-)$ is cloned from Q using an older set of parameters θ^- in every C iterations and is used to generate the Q-learning targets y for the following C iterations. In the beginning, both Q and $\hat{Q}$ are initialized with random parameters θ. Then, for each iteration t, the DQN receives state s_t and selects an action a_t according to an ε-greedy policy ($\varepsilon = 0.05$). The action is sent to ALNS and reward r_t and state s_{t+1} is received. The target Q-value y is calculated using the Bellman equation (19) with $\hat{Q}$:

$$y = r_t + \gamma \max_{a'} \hat{Q}(s',a';\theta_t^-). \tag{20}$$

The predicted Q-value is obtained using the current parameters θ_t in the network $Q(s,a;\theta_t)$. At each iteration t, the $Q(s,a;\theta_t)$'s parameters θ_t are updated to minimize the mean-squared error between the target and predicted Q-values, i.e., the loss function:

$$L_t(\theta_t) = \mathbb{E}_{s,a,r,s'}[(y - Q(s,a;\theta_t))^2] \tag{21}$$

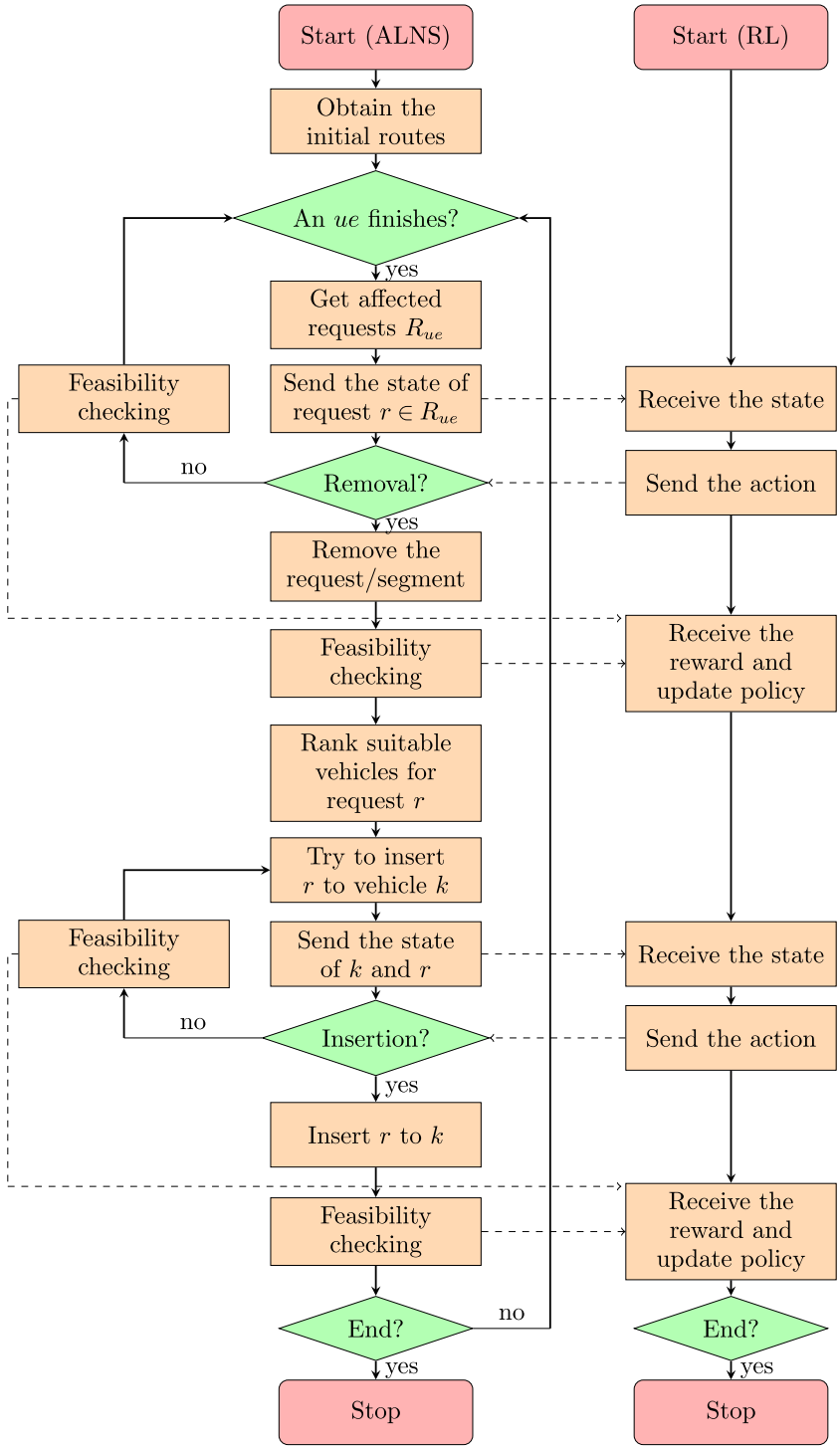


Fig. 7. The flowchart of the model-assisted RL. The dashed lines represent the interactions between ALNS and RL.

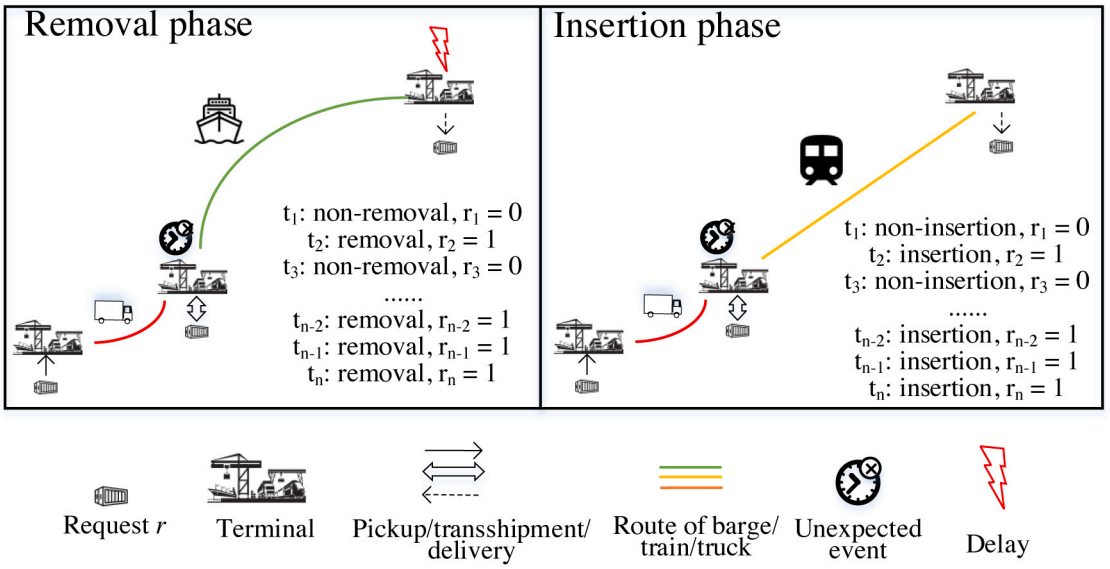


Fig. 8. An example of how RL learns.

Differentiating (21) with respect to θ_t, we obtain the following gradient:

$$\nabla_{\theta_t} L(\theta_t) = \mathbb{E}_{s,a,r,s'}[(y - Q(s,a;\theta_t))\nabla_{\theta_t} Q(s,a;\theta_t)]. \tag{22}$$

These gradients are then used by optimization algorithms like stochastic gradient descent used in this study to update the parameters in a direction that minimizes the loss.

Algorithm 6: Deep Q-network

Initialize the Q-network Q parameters θ randomly;
Initialize the target Q-network $\hat{Q}$ parameters $\theta^- = \theta$;
repeat
  Receive state s_1 from ALNS;
  for $t = 1, 2, ..., T$ **do**
    Select $a_t = \text{argmax}_a\, Q(s_t, a; \theta)$ or a random action a_t with probability ε;
    Send action a_t to ALNS and receive reward r_t and state s_{t+1};
    Calculate the target Q-value using the Bellman equation (20) and the predicted Q-value using $Q(s, a; \theta)$;
    Compute the loss function (21) as the mean squared error between target and predicted Q-values;
    Perform a gradient descent step using equation (22) and update θ using stochastic gradient descent to minimize the loss function;
    Reset $\hat{Q} = Q$ in every C iterations;
    if *the RL is mature* **then**
      Return: Trained Q-network
    end
  end
until *the number of episodes is reached*;
Return: Trained Q-network

Besides, the DQN also makes use of different techniques to stabilize the learning with neural networks, including the replay buffer and gradient clipping, as introduced by Mnih et al. (2015).

The process of how the RL agent learns to make decisions in the presence of unexpected events is illustrated in Fig. 8. A request is initially planned to be transported by truck and then transferred to a barge via transshipment. However, an unexpected event occurs at the transshipment terminal, requiring the RL agent to determine whether to remove the request from the barge and whether to insert it onto the train service as an alternative. If the request is removed from the barge and inserted onto the train, there will be no delay in its transportation. In the removal phase, the RL agent receives a reward for removing the request from the barge service, as remaining on the barge would result in a delay. Through training, the RL agent continually tries different actions and receives rewards, eventually learning to make the decision to remove the request. Similarly, in the insertion phase, the RL agent is trained to ultimately make the decision to insert the request onto the train service.

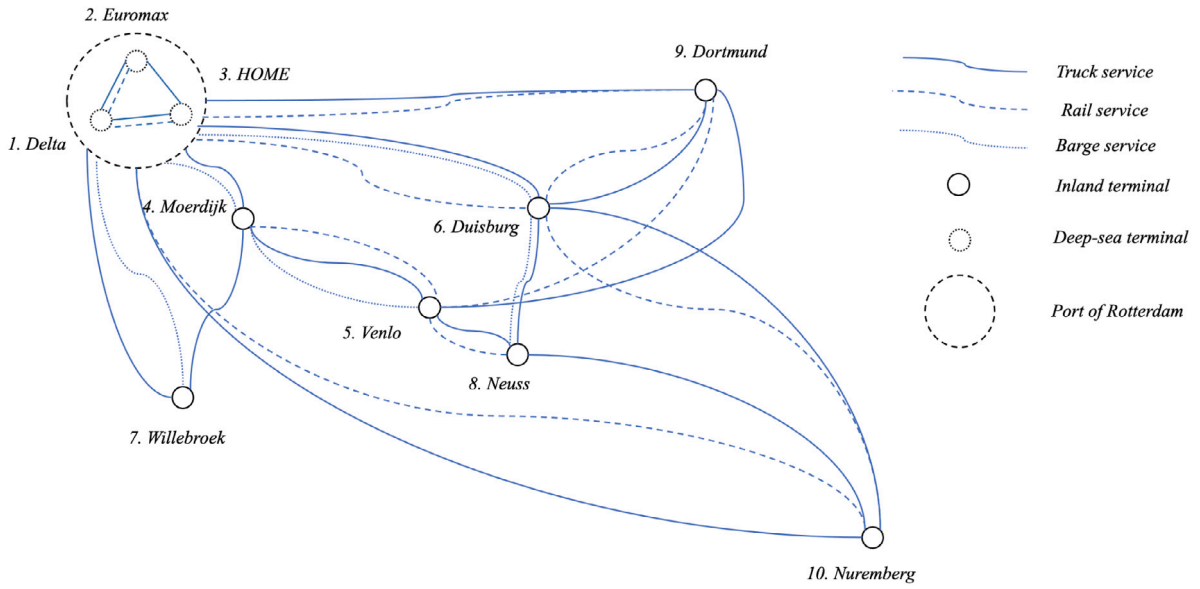


Fig. 9. Transport network of EGS,
Source: Zhang et al. (2022d).

5. Case study

The network shown in Fig. 9 inspired by the European Gateway Services (EGS) network located in the Rhine-Alpine corridor, is selected as a real-world case study to evaluate the effectiveness of the proposed planning approach. This network includes three terminals in the Port of Rotterdam and seven inland terminals in The Netherlands, Belgium, and Germany, and offers a total of 116 services including 49 barges, 33 trains, and 34 truck services, according to the EGS website (EGS, 2021). We assume a truck service is a fleet with an unlimited number of trucks and truck services are available between any of the two terminals. To evaluate the approach, instances with 5, 10, 20, 30, 50, and 100 shipment requests are designed. Requests have random origins and destinations among deep-sea and inland terminals. Container volume is uniformly distributed in [10, 30] TEUs. Earliest pickup $a_{p(r)}$ is uniformly distributed in $[1, 120]$ and latest delivery $b_{d(r)}$ is $a_{p(r)}+LD_r$, where LD_r is the lead time and is independently and identically distributed among {24, 48, 72} hours (probabilities {0.15, 0.6, 0.25}). Further information can be found in Zhang et al. (2022d).

We consider scenarios with different types of unexpected events, which moreover follow different duration distributions. A duration distribution is a statistical representation of the distribution of time periods for a specific type of event. It describes the likelihood of the event taking a certain amount of time. In order to generate realistic unexpected events, we use truncated normal distributions to exclude negative durations, which are commonly used in the literature (Srinivasan et al., 2014; Soltani-Sobh et al., 2016). These distributions are not known to the RL algorithm. In terms of the severity level information in the state of RL approach, we have three cases: no severity level information, perfect severity level information (the level is accurate), and imperfect severity level (where some levels are not as expected). In Section 5.1, there is no severity level information. In Section 5.2, because the scenario with multiple events is complex, perfect and imperfect severity levels are considered.

Unless otherwise specified, the maximum number of iterations in the learning phase is set to 5000, and the RL is evaluated every 100 iterations. During the evaluation, the RL is tested 10 times and the rewards are recorded. If the average reward is greater than 0.9 for five consecutive evaluations, we consider the RL to be mature and ready for implementation. During the implementation phase, the RL is used to make decisions for 200 iterations.

The performance is evaluated using two indicators: average reward of all iterations and total delay over all requests in the implementation phase. The performance indicators are evaluated from the carrier's perspective and consider all vehicles in the transport network. Rewards are given for actions that avoid delay (referred to as "correct actions"). A higher reward and lower delay indicate better performance. However, a high reward does not necessarily mean a low delay, as incorrect actions can result in substantial delays even if the majority of actions are correct and result in a high average reward. We also evaluate the proportion of rewards obtained through removal, waiting, non-insertion, and insertion actions, with higher proportions being favorable.

5.1. Results under disruptions and disturbances without severity level information

To test the model under different types of unexpected events, several scenarios are designed, including (a) disturbances, (b) severe disturbances, (c) disruptions, and (d) a mix of disruptions and disturbances. The distributions used in each scenario are

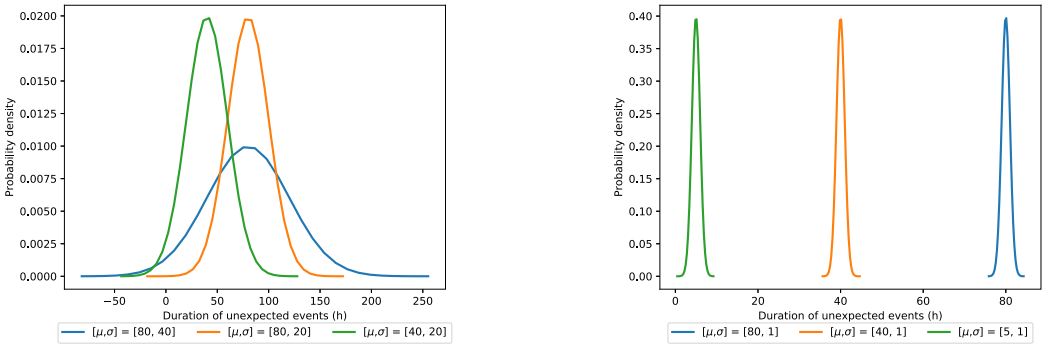


Fig. 10. Normal distributions used in different scenarios (unknown to RL). These distributions are truncated at zero to avoid negative duration.

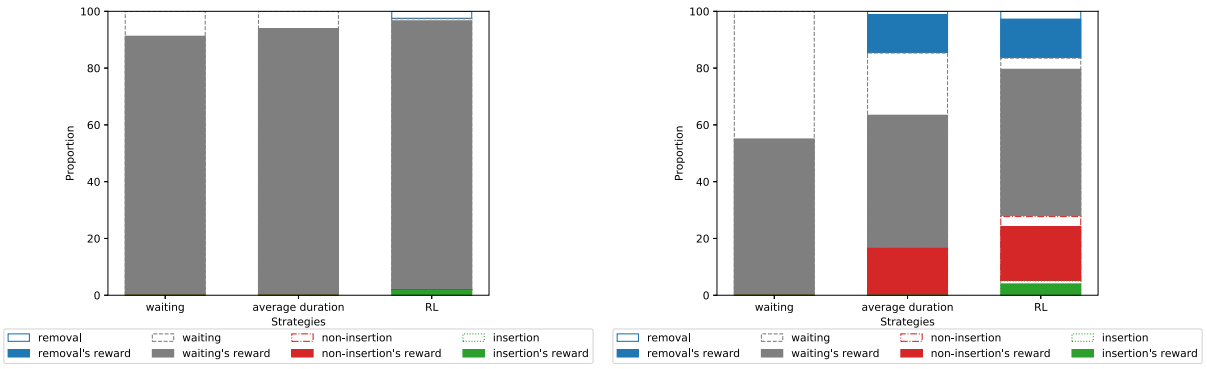
illustrated in Fig. 10. In scenario (a), the mean value μ of the duration distribution is set to a small value of 5 h, and the standard deviation σ is set to 1. This represents a situation where the duration of unexpected events is generally short but still somewhat variable. In scenario (b), the distribution is defined by the parameters $[\mu, \sigma] = [40, 20]$ or $[40, 1]$. In scenario (c), the mean value μ of the distribution is set to 80 h, and the standard deviation σ is set to 40, 20, or 1. This allows us to evaluate the performance of the proposed approach under different levels of variability in the duration of unexpected events. In scenario (d), the terminals are divided into two groups. The first group (terminals 1-5) and the second group (terminals 6-10) experience different types of events in scenarios (a), (b), and (c). The findings from scenario (d) are presented in Appendix B, demonstrating similar insights as the other scenarios. The information on these distributions is unknown to RL.

Fig. 11 shows the proportion of actions and rewards among various strategies. Each rectangle of a distinct color represents a specific action, with the size of the rectangle indicating the proportion of that action. The filled portion of each rectangle represents the proportion of rewards obtained through the corresponding action, while the blank portion represents the proportion of the action that does not result in a reward. Across all scenarios, the RL strategy (strategy (c)) consistently performs the best, while the waiting strategy (strategy (a)) consistently performs the worst. The waiting strategy performs well only in the presence of disturbances in Fig. 11(a), where the wait time is sufficient in most cases. As the severity of unexpected events increases, the waiting strategy performs increasingly poorly. The average duration strategy (strategy (b)) performs worse as the variation in the unexpected events becomes larger, as it becomes more difficult to utilize average duration to determine the optimal action in such circumstances. In the presence of disruptions, the proportion of non-insertion and removal actions increases as the strategy attempts to mitigate the disruptions and subsequent delays. The RL strategy uses more insertion and waiting actions compared to the average strategy, even in the presence of disruptions, because it is able to identify situations in which requests can still be serviced by vehicles despite the disruptions occurring frequently at terminals. This capability allows the RL strategy to earn more rewards compared to the other two strategies. The RL strategy also exhibits superior performance in terms of its ability to accurately recognize and execute non-insertion and removal actions.

Fig. 12 compares the delay (in hours) of different strategies under various numbers of requests and scenarios where the duration of unexpected events at all terminals follows the same distribution. It is observed that the delay of the waiting strategy is higher than other strategies in 75% of the cases. The RL strategy is relatively insensitive to increases in the variations or stochasticity of the duration of the events, and the total delay is the lowest in all scenarios, including disturbances, severe disturbances, and disruptions. As the severity of the events increases, the maximum delay for the waiting and average duration strategies increases significantly, while the maximum delay for the RL strategy remains below half of the maximum delay for the other strategies in the majority of cases. The delay of the RL strategy is lower than the other two strategies in 80% of the cases. In the remaining cases, the RL strategy performs better than at least one of the other two strategies in five out of seven cases. On average, the RL strategy reduces delay compared to the average duration strategy by 9.6% and the waiting strategy by 53.8%. This suggests that the RL approach is more effective at handling unexpected events and minimizing the delay compared to the waiting and average duration strategy.

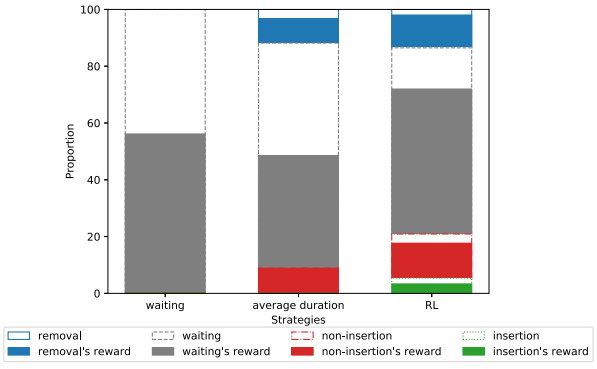
5.2. Results under multiple events with perfect and imperfect severity levels

In previous experiments, we assumed that there is only one type of event at each terminal. However, multiple types of events can occur at the same terminal, with some events causing disruptions and others causing disturbances. To tackle this issue, we created five scenarios incorporating 2-6 types of events occurring at the same terminal. In each scenario, 12 cases are generated with different events, randomly chosen from the following types ($[\mu, \sigma]$): disturbance ([5, 1]), severe disturbance $[\mu, \sigma]$, severe disturbances with a higher standard deviation ([40, 20]), disruption ([80, 5]), and disruptions with a higher standard deviation ([80, 40]). In the majority of the cases, the types of events differ, but some cases contain the same type of events. This section exclusively displays the results of the simplest (two events) and most complex (six events) scenarios, while the results of other scenarios can be found in Appendix B.

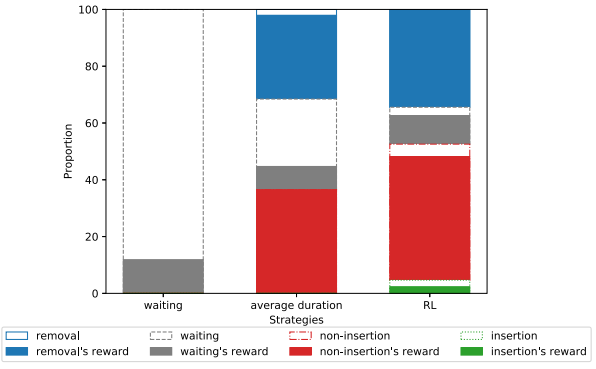


(a) disturbances ([5,1])

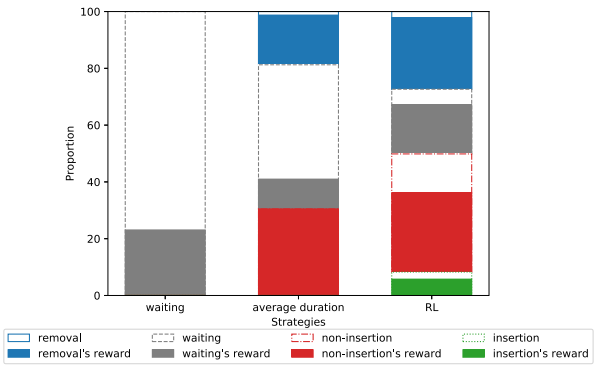
(b) severe disturbances ([40,1])



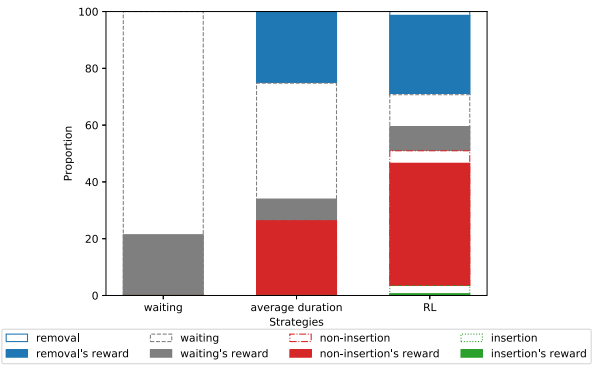
(c) severe disturbances with medium variations ([40,20])



(d) disruptions ([80,1])



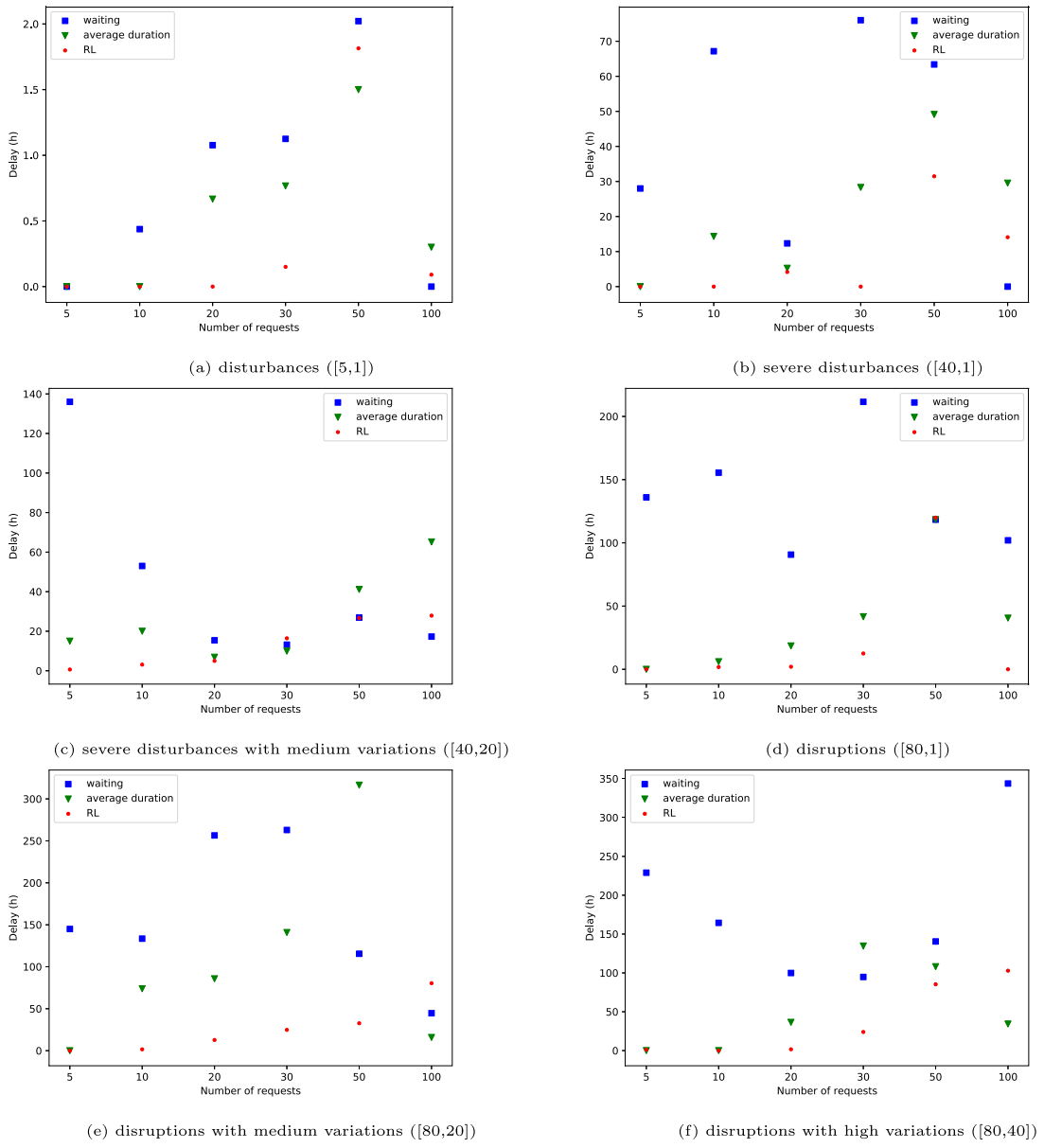
(e) disruptions with medium variations ([80,20])



(f) disruptions with high variations ([80,40])

Fig. 11. Proportion of actions and rewards under disturbances or disruptions.

Fig. 13 presents the average rewards obtained by all actions of the RL strategy with varying numbers of training iterations under scenarios involving different numbers of events occurring at the same terminal. It is observed that in the scenario with two events (Fig. 13(a)), the RL's average rewards reach 0.9 when the number of training iterations is 10000, indicating that the RL is able to choose correct actions in more than 90% of cases. However, as the number of events increases, the performance of the RL declines, with the average reward unable to reach 0.8 in scenarios with six events. This suggests that the problem becomes increasingly complex as the number of events increases, and the RL is unable to effectively solve it without additional information. In order to measure the performance of RL in complex scenarios with multiple events at the terminal, the state has been augmented with the inclusion of a severity level. The severity level is labeled from 1 to 6 and is defined as follows: Level 1: duration time ≤ 20, Level 2: duration time $\in (20, 40]$, Level 3: duration time $\in (40, 60]$, Level 4: duration time $\in (60, 80]$, Level 5: duration time $\in (80, 100]$, Level 6: duration time > 100. The RL is only informed about the level as a label but does not know the duration. Fig. 13 also presents the average rewards for scenarios after adding a severity level to the state. The results indicate that the average reward is able to reach 0.8 in most cases when the number of training iterations is 1000, and approaches or exceeds 0.9 when the number of training



(a) disturbances ([5,1])

(b) severe disturbances ([40,1])

(c) severe disturbances with medium variations ([40,20])

(d) disruptions ([80,1])

(e) disruptions with medium variations ([80,20])

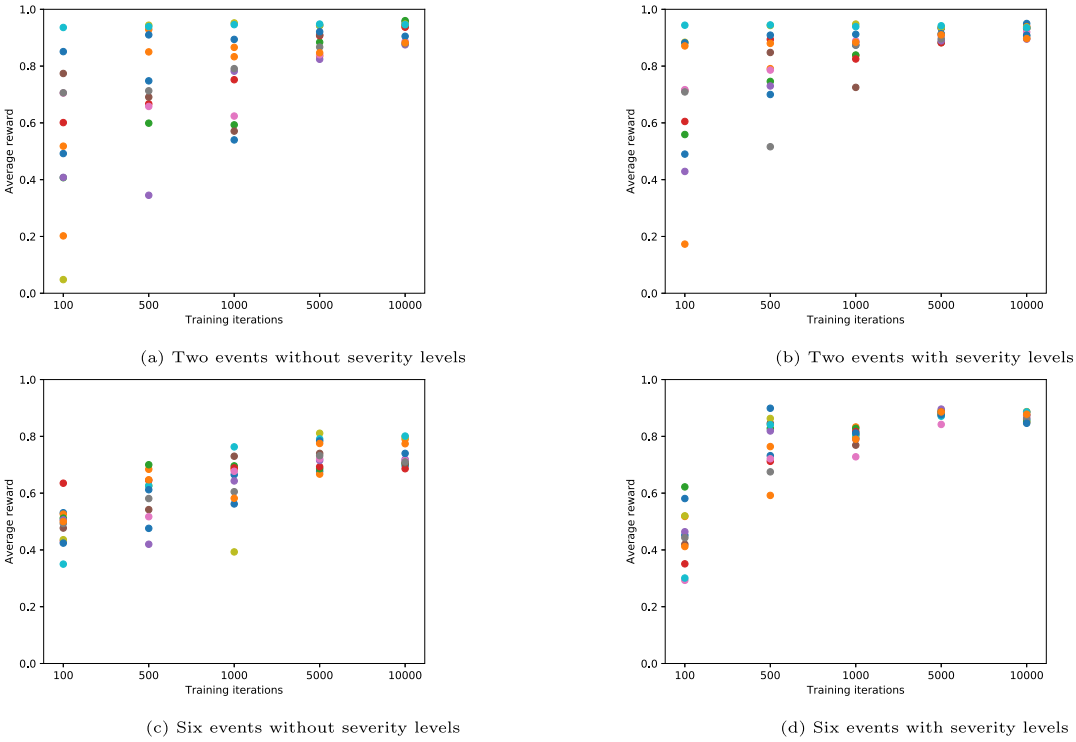
(f) disruptions with high variations ([80,40])

Fig. 12. Total delay over all requests under disturbances or disruptions.

iterations is 5000. This suggests that incorporating a severity level into the state is beneficial in enabling the RL to choose correct actions, as it can help to differentiate between events with different levels of impact and allow for more informed decision-making.

Fig. 14 presents the average rewards of the three strategies under various numbers of requests in scenarios with multiple events and severity levels. Across all cases, the RL strategy's average rewards of handling all requests are higher than the waiting and average duration strategies. Fig. 15 provides the proportions of actions and the corresponding rewards obtained by each action. This figure more clearly demonstrates the RL's ability to accurately utilize different actions in complex cases involving up to six events at a single terminal. Fig. 16 compares the delay experienced by the different strategies. In 25 out of 30 cases, the RL strategy performs the best among the three strategies, and in the remaining four out of five cases, the RL's performance is similar to that of the other two strategies. On average, the RL strategy reduces delay compared to the waiting and average duration strategies by 52.8% and 29.0%, respectively.

While incorporating severity levels can improve the performance of the RL, it is important to recognize that such knowledge may be imperfect, potentially incomplete or outdated, subject to interpretation, or prone to measurement errors. In these cases, the RL may make suboptimal decisions or take longer to learn an optimal policy. To assess the RL's performance under imperfect knowledge



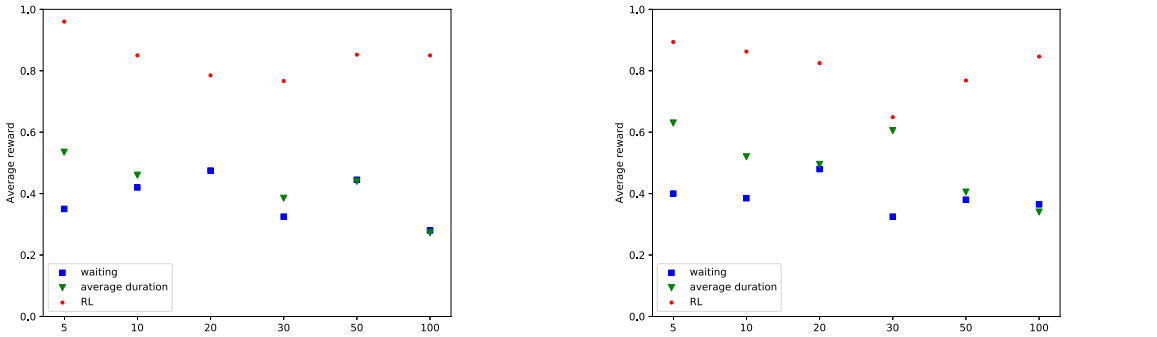
(a) Two events without severity levels

(b) Two events with severity levels

(c) Six events without severity levels

(d) Six events with severity levels

Fig. 13. Average rewards under multiple types of events at the same terminal (12 cases with different colors).



(a) two events [5, 1], [80, 5]

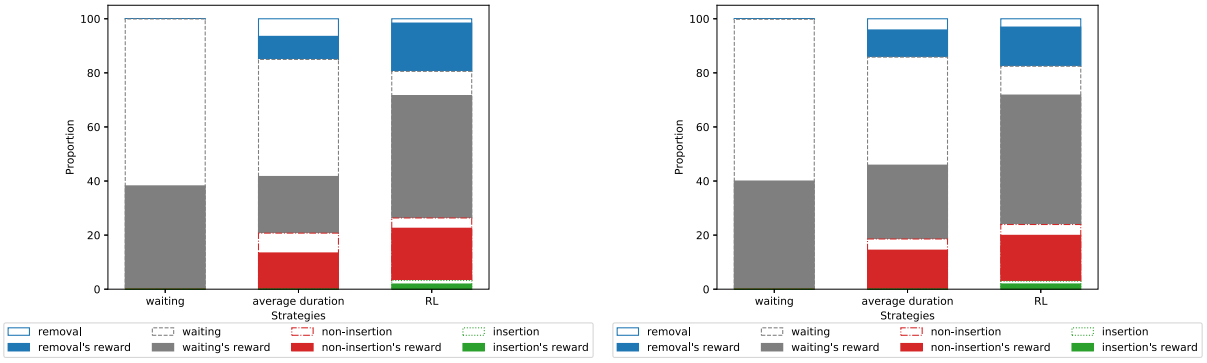
(b) six events [5, 1], [80, 5], [40, 5], [5, 1], [40, 20], [80, 40]

Fig. 14. Average rewards of different strategies under multiple events at the same terminal.

of severity levels, we designed scenarios that include random severity levels with a probability ranging from 0.2 to 0.5. This section specifically presents the results corresponding to probabilities 0.2 and 0.5 in Fig. 17, while the results associated with probabilities 0.3 and 0.4 can be found in Appendix C. For the scenario with two or six events, as the probability increases, the average reward decreases, but still reaches 0.8 or 0.7 with sufficient training iterations when half of the severity levels are randomly provided. The incorporation of imperfect knowledge can increase the complexity of the problem for the RL, requiring it to consider multiple potential states and incorporate uncertainty into its decision-making process. However, the use of deep neural networks in the RL allows the agent to adapt to changes in the environment, even with imperfect knowledge of the state, making it particularly useful in complex synchromodal transport environments where other methods may be ineffective.

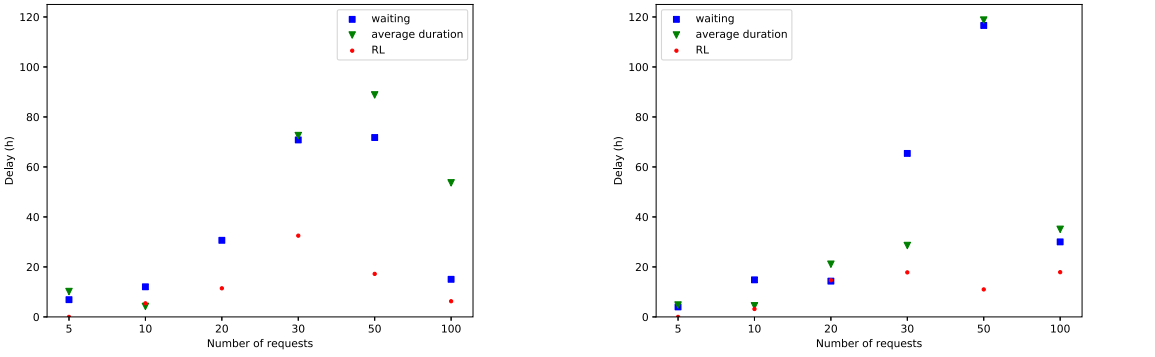
5.3. Analysis of other performance indicators: served requests, costs, emissions, waiting time, and training time

Besides delay and reward, there are several additional performance indicators that need to be considered, such as the number of served requests, costs, emissions, waiting time, and training time. The waiting strategy serves all requests even at the cost of a high



(a) two events [5, 1], [80, 5] (b) six events [5, 1], [80, 5], [40, 5], [5, 1], [40, 20], [80, 40]

Fig. 15. Proportion of actions under multiple events at the same terminal.

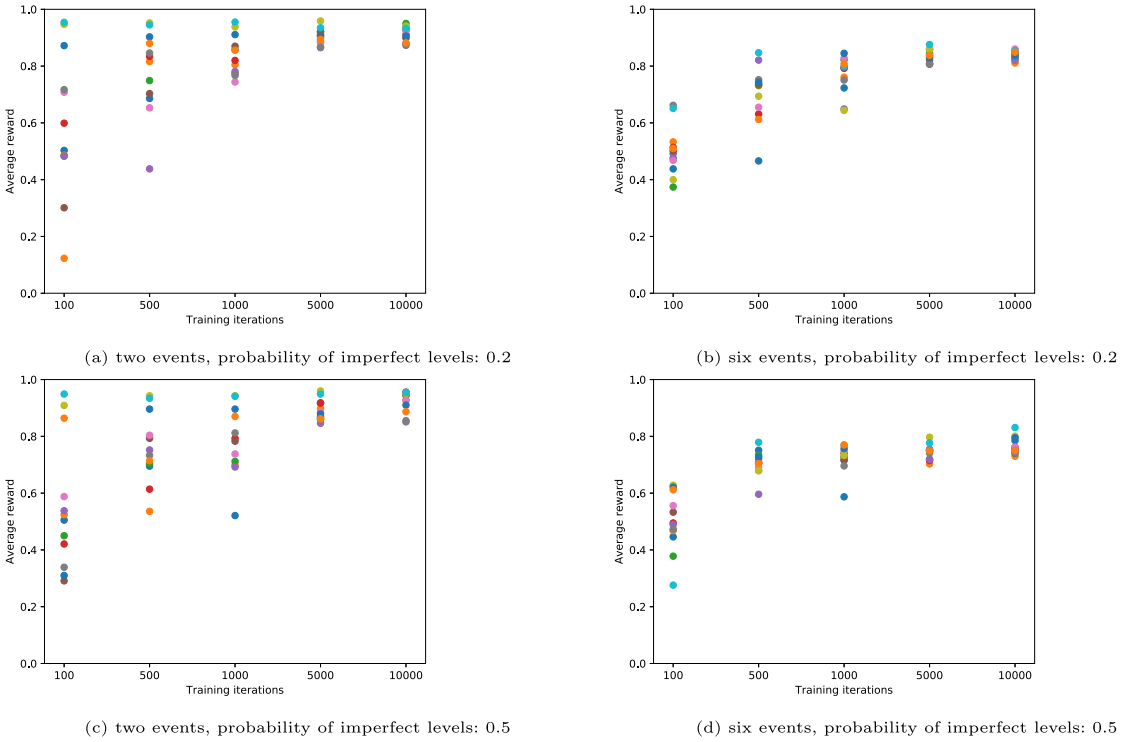


(a) two events [5, 1], [80, 5] (b) six events [5, 1], [80, 5], [40, 5], [5, 1], [40, 20], [80, 40]

Fig. 16. Total delay over all requests under multiple events at the same terminal.

delay penalty. The average duration and RL strategies may selectively unserve a limited number of requests in order to optimize overall performance in instances where delays are unavoidable and alternate services are more appropriate for other requests. In the average duration strategy, all requests are served in 91.2% of the experiments, in 7% of the experiments only one request is not served, and in the remainder 1.8% of the cases two requests are left unserved. The RL strategy has a higher rate of served requests, with all requests being served in 93.9% of the experiments, only one request being unserved in 5.7% of the experiments and two being unserved only in 0.4% of the cases. It is noteworthy that the experiments with unserved requests are mostly the larger ones such as those with 100 requests.

The performance indicators including average cost per request, average emissions, and average waiting time are presented in Figs. 18, 19, and 20, respectively. These results are derived from the evaluation of different strategies under a variety of scenarios, including disturbance, severe disturbance, disruption, and mixed events in various terminals as discussed in Section 5.1, as well as multiple events at a single terminal in Section 5.2. In the scenario with multiple events, the RL strategy with severity level is used. The performance of the different strategies in terms of cost is shown in Fig. 18. The average duration and RL strategies have demonstrated an improvement over the waiting strategy, reducing costs by 26.8% and 44.0%, respectively. This is attributed to the better handling of service time uncertainty, leading to a reduction in delay penalties and the effective adjustment of transport plans, avoiding the use of more expensive trucks in the late stages. Handling service time uncertainty not only leads to cost reduction, but also results in a decrease in emissions, particularly under scenarios with disruptions, mixed events, and multiple events, as shown in Fig. 19. The waiting strategy, which only implements re-planning upon the occurrence of a significant delay, often leads to high-cost, high-emission vehicles to mitigate the delay at the last minute. On the other hand, the average duration and RL strategies reduce emissions by switching the shipment request to a suitable vehicle in the presence of unexpected events and reducing the need for high-emission vehicles at the last minute. As illustrated in Fig. 20, the waiting time is significantly reduced when compared to the waiting strategy. The average duration and RL strategies have resulted in a reduction of 13.2% and 24.5%, respectively. The efficient handling of service time uncertainty allows for a more agile and flexible allocation of resources, leading to the avoidance of unnecessary wait times and the prompt adjustment of shipment requests to suitable vehicles. These results highlight the benefits of efficient handling of service time uncertainty, as it reduces the risk of missing the best time to switch vehicles and reduces costs, emissions, and waiting time.



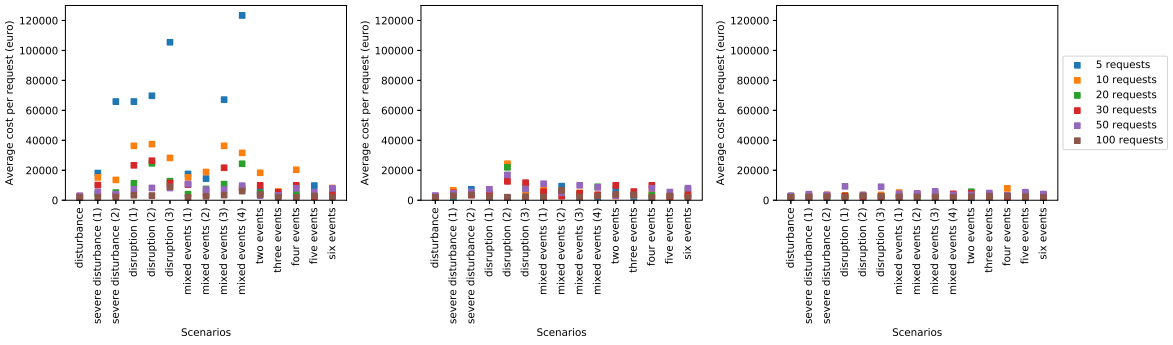
(a) two events, probability of imperfect levels: 0.2

(b) six events, probability of imperfect levels: 0.2

(c) two events, probability of imperfect levels: 0.5

(d) six events, probability of imperfect levels: 0.5

Fig. 17. Average rewards under multiple events with imperfect severity levels (12 cases with different colors).



(a) waiting strategy

(b) average duration strategy

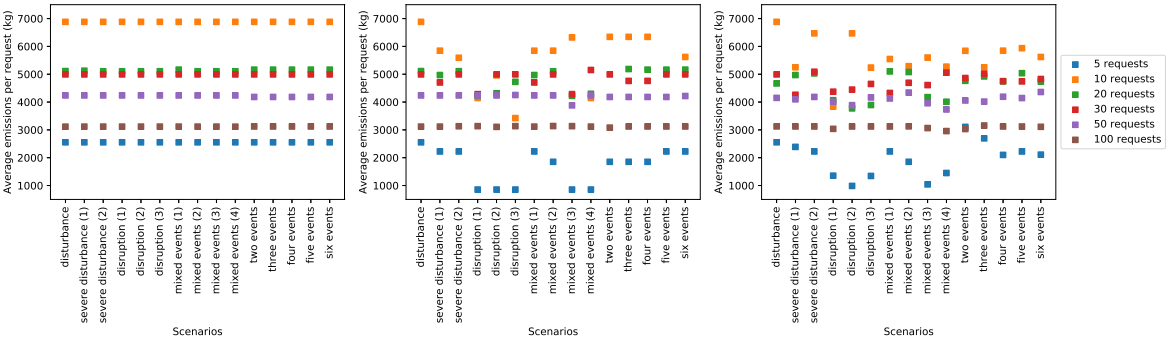
(c) RL strategy

Fig. 18. Average cost per request in different scenarios with different strategies.

The training time for the RL strategy is presented in Fig. 21. The total duration of training is no more than one hour when the size of the instance is small, such as the instance with 5 requests. The total training time increases proportionally with the size of the instance. The average training time per iteration is calculated to be a few seconds, with the longest being less than three minutes for the largest instance. As the duration of service time in the field of synchromodal transport often requires several hours, the training can be completed during this period and can be done online. Additionally, the time required for the RL approach to make a decision is less than 1 ms when the RL approach is implemented, making it an efficient solution for real-time decision-making.

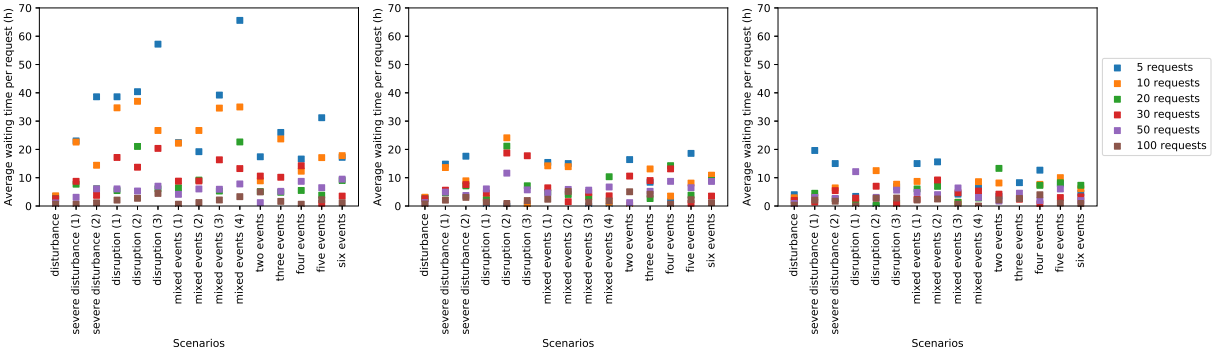
5.4. Ablation study

To evaluate the impact of individual components within the neural network of DQN, we conduct an ablation study by selectively modifying the number of layers, the number of neurons in each layer, and the activation function, and observing the resulting performance changes. Our current DQN configuration employs a neural network architecture with two hidden fully connected layers, each consisting of 64 neurons. The Rectified Linear Unit (ReLU) activation function is used. In our ablation study, we investigate the



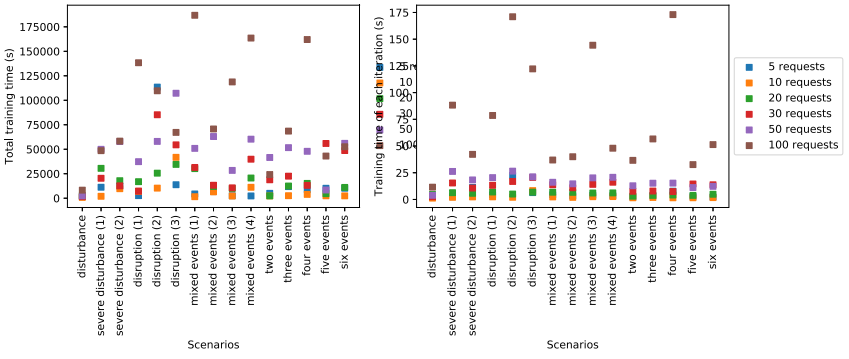
(a) waiting strategy (b) average duration strategy (c) RL strategy

Fig. 19. Average emissions per request in different scenarios with different strategies.



(a) waiting strategy (b) average duration strategy (c) RL strategy

Fig. 20. Average waiting time per request in different scenarios with different strategies.

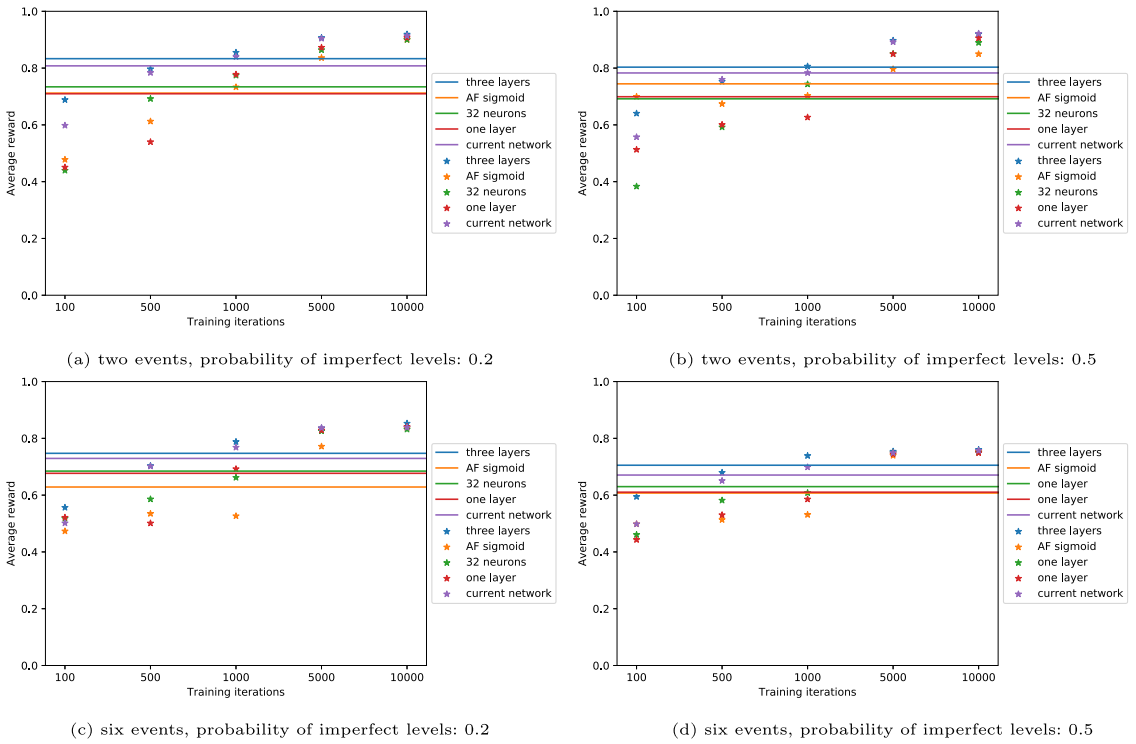


(a) total training time (b) average training time of each iteration

Fig. 21. Training time of RL strategy in different scenarios.

following cases: (a) 32 neurons: We reduce the number of neurons in each layer to 32. (b) One layer: We retain only one layer with 64 neurons. (c) Sigmoid activation function: We replace the ReLU activation function with the sigmoid function, which is another commonly used activation function (Dubey et al., 2022). (d) Three layers: We add an extra layer with 64 neurons.

We conduct the ablation study using scenarios that involve two or six events and imperfect levels. Fig. 22 shows the average reward achieved for each case. Upon analysis, we find that the current network configuration consistently outperforms the other cases, except for the scenario with three layers. However, the difference in average rewards between the current network and the three-layer network is less than 0.01 when a sufficient number of training iterations are performed. Additionally, we compare the computation times of the current network and the three-layer network. It is observed that the three-layer network requires



(a) two events, probability of imperfect levels: 0.2

(b) two events, probability of imperfect levels: 0.5

(c) six events, probability of imperfect levels: 0.2

(d) six events, probability of imperfect levels: 0.5

Fig. 22. Ablation study. The stars represent the reward with different training iterations and the lines are the average rewards across all training iterations..

approximately 14% more time to train compared to the current network. Considering the trade-off between reward and computation time, we conclude that the current network configuration is the most suitable option.

5.5. Performance of trained policy on different transport networks

In previous sections, we conduct training and testing of our proposed approach on the EGS transport network. To further explore the transferability of the trained policy, we evaluate its performance on another distinct transport network. In this case, we chose Contargo's transport network, a renowned intermodal container hinterland logistics network operating extensively in Europe (Zhang et al., 2022a,c). Contargo plays a pivotal role in facilitating container transport between western seaports, Germany's North Sea ports, and the European hinterland. Fig. 23 visually illustrates both the Contargo and EGS transport networks on the same map, providing a comparative perspective. Contargo's transport network includes 20 terminals/ports and it operates 38 barge services, 23 train services, and 95 truck services. To establish the distances between terminals of different modes within Contargo's network, we utilize the same data sources as in Shobayo et al. (2021). We obtain service-related information from schedules available on Contargo's websites (Contargo, 2021). Similarly, we generate an instance with 30 requests for Contargo's network following a similar approach used for the EGS network.

We deploy the trained policy from EGS directly on Contargo's transport planning, which means there is no training using Contargo's transport network information. Our evaluation includes scenarios encompassing both disturbances (Scenario 1: $[\mu, \sigma] = [40, 1]$, Scenario 2: $[\mu, \sigma] = [40, 20]$) and disruptions (Scenario 3: $[\mu, \sigma] = [80, 20]$, Scenario 4: $[\mu, \sigma] = [80, 40]$). We select these scenarios as they cannot be easily addressed by always employing the same action. Each scenario is repeated three times to obtain average results. Table 3 presents the results in terms of the proportion of actions, average reward, delay, and computation time per iteration. The proportion of actions and success rates demonstrate that the trained policy can consistently select the appropriate action for a new transport network in the majority of cases. The average reward for scenarios with disturbances (Scenarios 1 and 2) surpasses that of scenarios with disruptions (Scenarios 3 and 4), owing to the higher variability in the disruption scenarios. In Scenario 1, the average reward reaches 0.9, while even in the scenario with the lowest performance (Scenario 4), the reward remains above 0.7. As the severity levels escalate from Scenario 1 to 4, the delay also increases. Notably, the computation time for each iteration is completed in approximately half a second for all experiments, affirming the feasibility of implementing the proposed model in real-time settings. These results indicate that despite differing terminals, distances, and network architectures, a learned policy from one transport network can be effectively applied to another transport network. If further performance is desired, the RL needs to be trained on the new network as well.

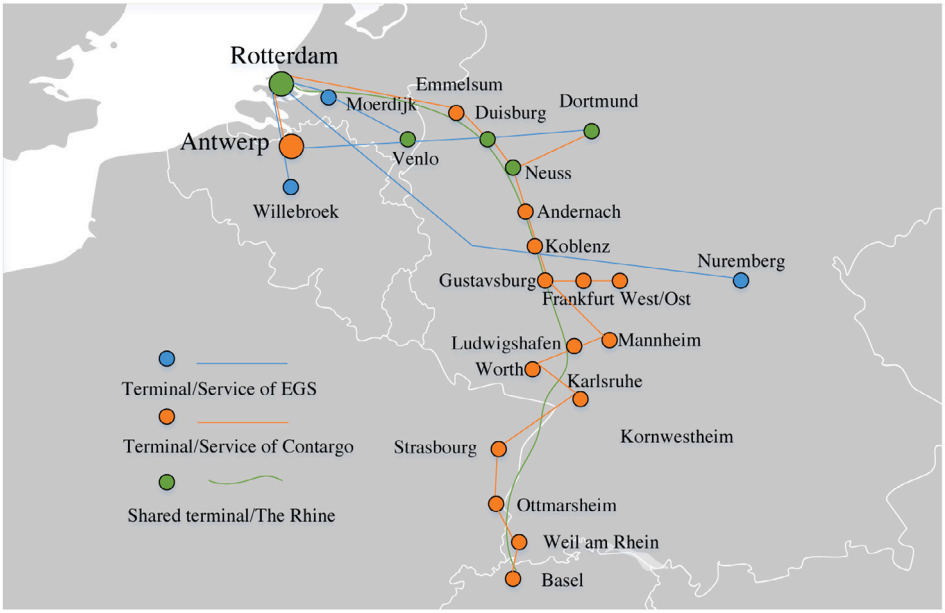


Fig. 23. Transport networks of Contargo and EGS.

Table 3
Performance of the trained policy on Contargo's transport network.

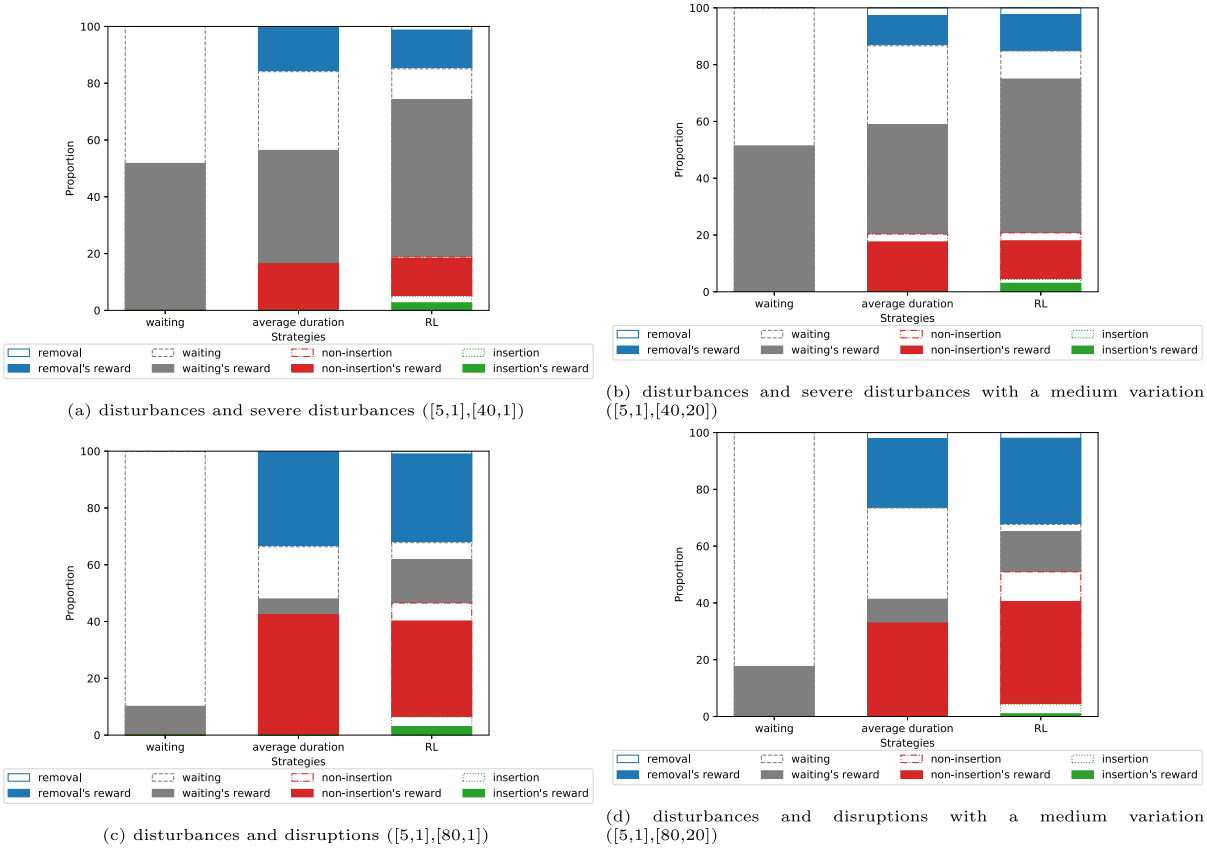
Scenario	Proportion of action (success rate)				Reward	Delay (h)	Computation time (s)
	Removal	Waiting	Non-insertion	Insertion			
1	6.8% (1.4%)	83.6% (81%)	2.9% (1.9%)	6.8% (5.5%)	0.90	0.80	0.50
2	12.3% (5.7%)	72.6% (68.6%)	6.3% (4.4%)	8.8% (5.3%)	0.84	2.50	0.53
3	32.2% (26.0%)	22.7% (19.3%)	27.9% (23.3%)	17.3% (7.4%)	0.76	10.04	0.55
4	36.3% (23.7%)	20.8% (17.3%)	24.4% (20.1%)	18.5% (11.1%)	0.72	20.25	0.53

6. Conclusions and future research

It is important to consider the challenges of managing synchromodal transport operations in the presence of service time uncertainty due to unexpected events at terminals. Unexpected events, such as disruptions or disturbances, can have a significant impact on the efficiency and effectiveness of transportation processes, resulting in delays, high costs, and high emissions. These events are often difficult to predict and can be caused by a variety of factors, including weather, accidents, or maintenance issues. As a result, it is crucial for transport operators to have tools and strategies in place to mitigate the impact of such events and maintain the smooth operation of the transportation system.

We have proposed a Reinforcement Learning (RL) approach for online synchromodal transport planning that can handle uncertainty and determine whether requests should be switched to different vehicles in case of delays. The RL approach is assisted by an Adaptive Large Neighborhood Search (ALNS) heuristic, which provides state and reward information, makes changes on the transport plans, and checks the feasibility of schedules. The model-assisted RL approach learns in real time and adapts its recommendations for carriers dynamically based on the uncertain service time conditions in the environment. This approach can be used by synchromodal transport carriers through a digital platform, where the carrier receives information about unexpected events from port authorities and terminal operators.

Several scenarios that varied in the type and severity of unexpected events and the level of variability in their duration are investigated. The performance of each strategy is measured in terms of average reward and total delay. The results of this study indicate superior performance of RL on unexpected events, as it is able to adapt to unexpected events and effectively handle complex scenarios, resulting in significantly reduced delays and higher rewards compared to other strategies. The waiting strategy, on the other hand, is unable to effectively mitigate the impact of disruptions or severe disturbances. The RL strategy outperforms the waiting and average duration strategies in the majority of cases, particularly when dealing with disruptions, a mix of disruptions and disturbances, and multiple events in a single terminal. The efficient handling of service time uncertainty, as demonstrated by the RL approach, leads to a reduction in costs, emissions, and waiting time by reducing delay penalties, avoiding the use of more



(a) disturbances and severe disturbances ([5,1],[40,1])

(b) disturbances and severe disturbances with a medium variation ([5,1],[40,20])

(c) disturbances and disruptions ([5,1],[80,1])

(d) disturbances and disruptions with a medium variation ([5,1],[80,20])

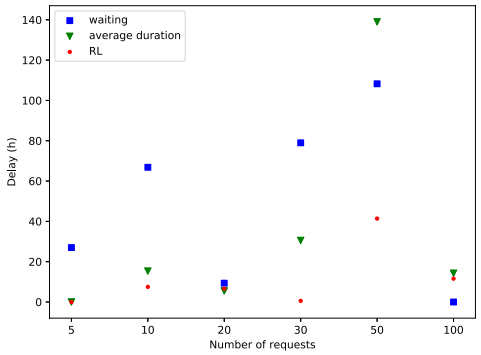
Fig. 24. Proportion of actions and rewards under different types of events occurring at different terminals.

expensive and high-emission trucks, and allowing for a more agile and flexible allocation of resources. Therefore, transportation managers may consider implementing the RL strategy in their decision-making process to reduce delays and increase efficiency in their operations. It is worth noting that the RL strategy requires a longer training period compared to the other two strategies, but this is compensated by its superior performance in the long run. Therefore, transportation managers should also consider investing sufficient training time to fully optimize the RL strategy's performance.

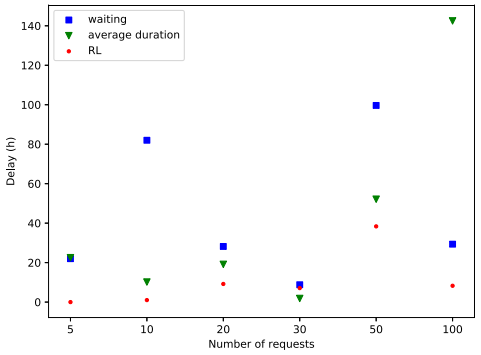
The potential of incorporating knowledge of event severity into the decision-making process is a key managerial insight from this study. The results indicate that providing this type of information to the RL algorithm can significantly improve its performance. Transportation managers should prioritize regularly updating and accurately assessing event severity information in order to optimize their management systems. However, it is important to note that imperfect information on event severity is inevitable in complex synchromodal transport systems due to various factors such as outdated or incomplete information, subjective interpretation, or measurement errors. Despite this, the proposed RL approach is able to handle imperfect information and still achieve good performance.

To better evaluate the performance of the proposed approach, we conduct the ablation study and test the performance of trained policy on different transport networks. The ablation study demonstrates that the current DQN configuration consistently outperforms alternative configurations in terms of average reward. The addition of an extra layer yields minimal improvement but increases training time by 14%, making the current configuration the most suitable choice. Our evaluation also demonstrates the transferability of the trained policy. Despite differences in terminals, distances, and network architectures, the policy achieves appropriate action selection for the new transport network.

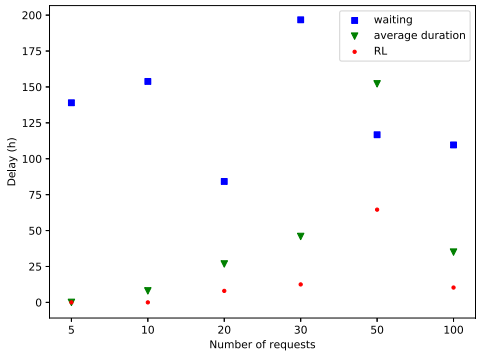
The following directions for future research are suggested: (a) The re-planning model can be extended to consider uncertainties in travel time and demand. (b) The proposed planning approach can be extended for multiple carriers. When used by multiple carriers, the experiences of these carriers can be shared, allowing each carrier to learn more quickly. Multi-agent Reinforcement Learning and Federated Learning are promising approaches for this research direction.



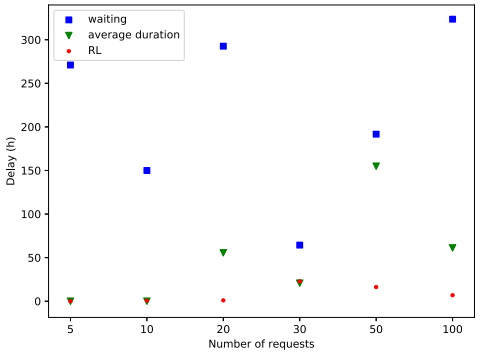
(a) disturbances and severe disturbances ([5,1],[40,1])



(b) disturbances and severe disturbances with a medium variation ([5,1],[40,20])



(c) disturbances and disruptions ([5,1],[80,1])



(d) disturbances and disruptions with a medium variation ([5,1],[80,20])

Fig. 25. Total delay over all requests under different types of events occurring at different terminals.

CRediT authorship contribution statement

Yimeng Zhang: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Rudy R. Negenborn:** Writing – review & editing, Supervision. **Bilge Atasoy:** Conceptualization, Methodology, Writing – review & editing, Supervision.

Acknowledgments

This research is supported by the China Scholarship Council (CSC) under Grant 201906950085. This research is also supported by the project "Novel inland waterway transport concepts for moving freight effectively (NOVIMOVE)". This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 858508.

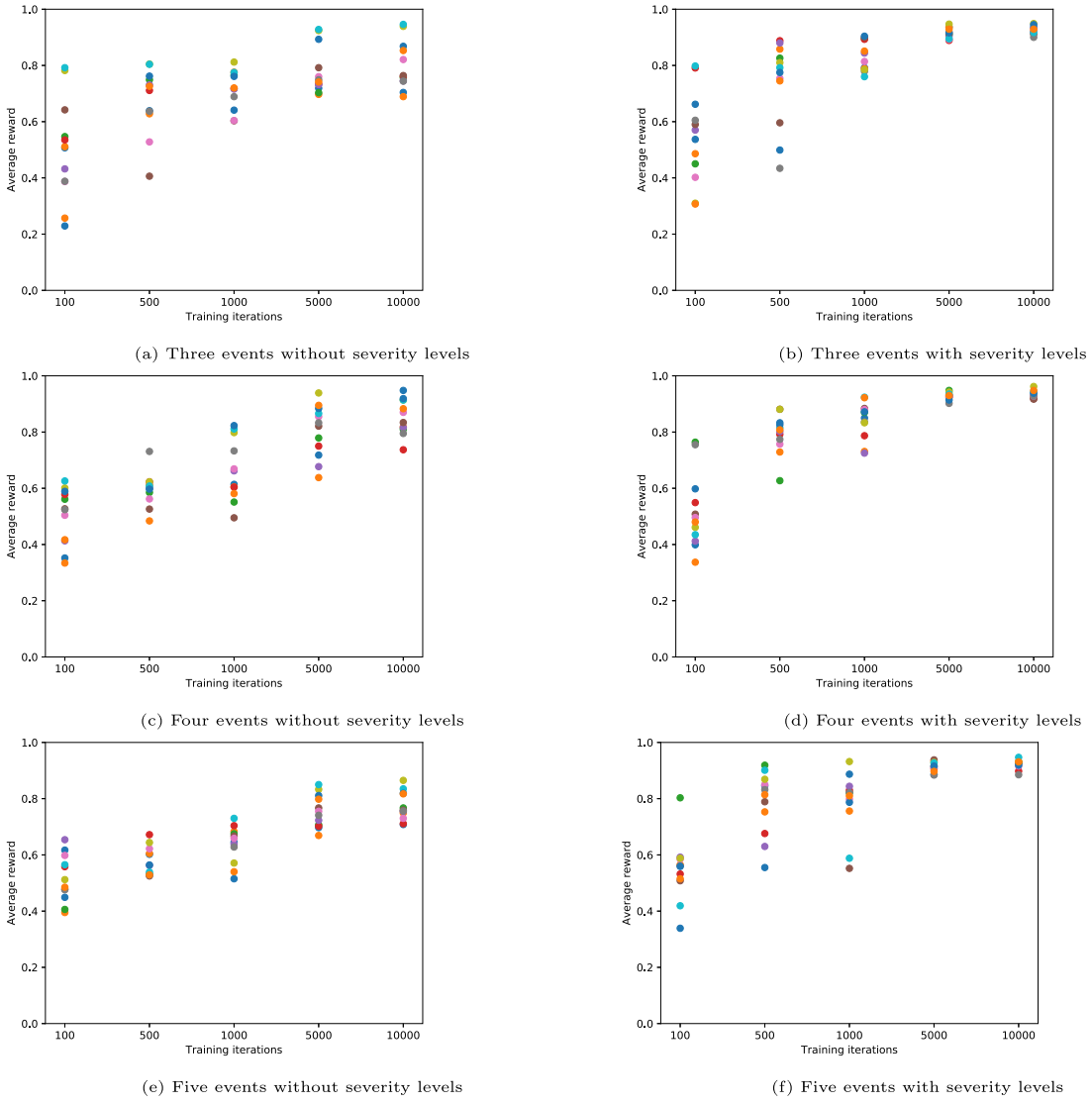
Appendix A. Constraints in the synchromodal transport planning model

Constraints (23)–(41) are the spatial constraints. Constraints (23) enforce that each vehicle may initiate at most one route from its beginning depot; Constraints (24) ensure that the same vehicle ends the route at its end depot. Not all of the available vehicles may have to be used in synchromodal transport, therefore we use "$\leqslant 1$" instead of "$= 1$" in constraints (23). Virtual depots ($\overline{o}(k)/\overline{o'}(k) \in \overline{O}$) are used to avoid the constraints on pickup/delivery terminals also work on depots. These virtual depots have the same location as depots, but with different names. Constraints (25) and (26) ensure that containers for each request are picked and delivered at their pickup and delivery terminals, respectively.

$$\sum_{j\in N} x^k_{\overline{o}(k)j} \leqslant 1 \quad \forall k \in K_{\text{b\&t}} \tag{23}$$

$$\sum_{j\in N} x^k_{\overline{o}(k)j} = \sum_{j\in N} x^k_{j\overline{o'}(k)} \quad \forall k \in K_{\text{b\&t}} \tag{24}$$

$$\sum_{k\in K}\sum_{j\in N} y^{kr}_{p(r)j} \leqslant 1 \quad \forall r \in R \tag{25}$$



(a) Three events without severity levels

(b) Three events with severity levels

(c) Four events without severity levels

(d) Four events with severity levels

(e) Five events without severity levels

(f) Five events with severity levels

Fig. 26. Average rewards under multiple types of events at the same terminal (supplementary results, 12 cases with different colors).

$$\sum_{k\in K}\sum_{j\in N} y^{kr}_{jd(r)} \leqslant 1 \quad \forall r \in R \tag{26}$$

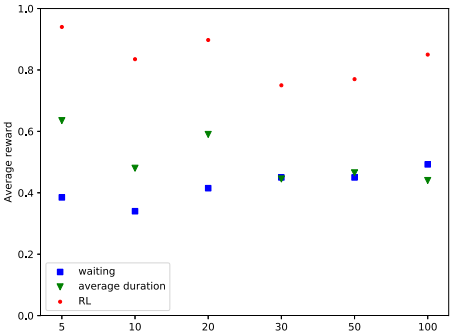
Constraints (27) and (28)–(31) represent flow conservation for vehicle and request flows, respectively. Constraints (28) and (29) are for regular and transshipment terminals, respectively. If request r is not transferred at terminal $i \in T$ but vehicle k passes terminal i due to operations for other requests, Constraints (28) do not work on request r. Therefore, additional flow conservation of requests (Constraints (30) and (31)) are added. Constraints (32) link y^{kr}_{ij} and x^{k}_{ij} variables in order to guarantee that for a request to be transported by a vehicle, that vehicle needs to traverse the associated arc.

$$\sum_{j\in N} x^{k}_{ij} - \sum_{j\in N} x^{k}_{ji} = 0 \quad \forall k \in K_{\text{b\&t}},\ \forall i \in N \setminus \overline{o}(k), \overline{o'}(k) \tag{27}$$

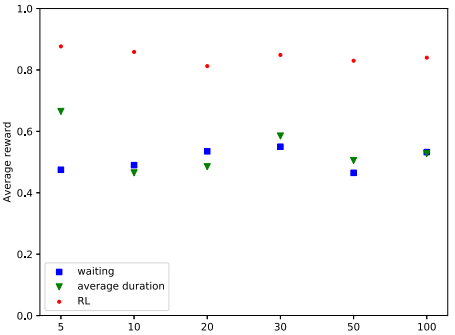
$$\sum_{j\in N} y^{kr}_{ij} - \sum_{j\in N} y^{kr}_{ji} = 0 \quad \forall k \in K,\ \forall r \in R,\ \forall i \in N \setminus T, p(r), d(r) \tag{28}$$

$$\sum_{k\in K}\sum_{j\in N} y^{kr}_{ij} - \sum_{k\in K}\sum_{j\in N} y^{kr}_{ji} = 0 \quad \forall r \in R,\ \forall i \in T \setminus p(r), d(r) \tag{29}$$

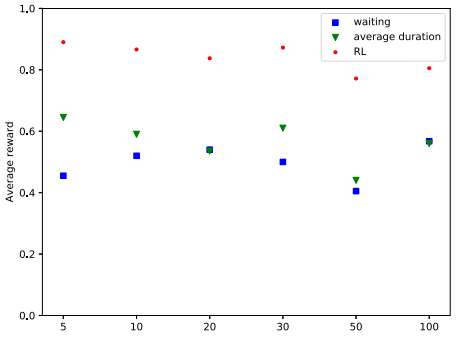
$$\sum_{j\in N} y^{kr}_{ij} - \sum_{j\in N} y^{kr}_{ji} \leqslant \sum_{l\in K} s^{lk}_{ir} \quad \forall k \in K,\ \forall r \in R,\ \forall i \in T \setminus p(r), d(r) \tag{30}$$



(a) three events [5, 1], [80, 5], [40, 5]



(b) four events [5, 1], [80, 5], [40, 5], [5, 1]



(c) five events [5, 1], [80, 5], [40, 5], [5, 1], [40, 20]

Fig. 27. Average rewards of different strategies under multiple events at the same terminal (supplementary results).

$$\sum_{j\in N} y_{ji}^{kr} - \sum_{j\in N} y_{ij}^{kr} \leqslant \sum_{l\in K} s_{ir}^{kl} \quad \forall k \in K,\ \forall r \in R,\ \forall i \in T \setminus p(r), d(r) \tag{31}$$

$$y_{ij}^{kr} \leqslant x_{ij}^{k} \quad \forall (i,j) \in A,\ \forall k \in K,\ \forall r \in R \tag{32}$$

Constraints (33) ensure that the transshipment occurs only once per transshipment terminal. Constraints (34) forbid transshipment between the same vehicle k.

$$\sum_{j\in N} y_{ji}^{kr} + \sum_{j\in N} y_{ij}^{lr} \leqslant s_{ir}^{kl} + 1 \quad \forall r \in R,\ \forall i \in T,\ \forall k,l \in K \tag{33}$$

$$s_{ir}^{kk} = 0 \quad \forall r \in R,\ \forall i \in T,\ \forall k \in K \tag{34}$$

Constraints (35)–(37) are the subtour elimination constraints and provide tight bounds in relatively short computation time among several polynomial-size versions of subtour elimination constraints (Öncan et al., 2009). Constraints (38) are the capacity constraints.

$$x_{ij}^{k} \leqslant z_{ij}^{k} \quad \forall i,j \in N,\ \forall k \in K_{\text{b\&t}} \tag{35}$$

$$z_{ij}^{k} + z_{ji}^{k} = 1 \quad \forall i,j \in N,\ \forall k \in K_{\text{b\&t}} \tag{36}$$

$$z_{ij}^{k} + z_{jp}^{k} + z_{pi}^{k} \leqslant 2 \quad \forall i,j,p \in N,\ \forall k \in K_{\text{b\&t}} \tag{37}$$

$$\sum_{r\in R} q_r y_{ij}^{kr} \leqslant u_k x_{ij}^{k} \quad \forall (i,j) \in A,\ \forall k \in K \tag{38}$$

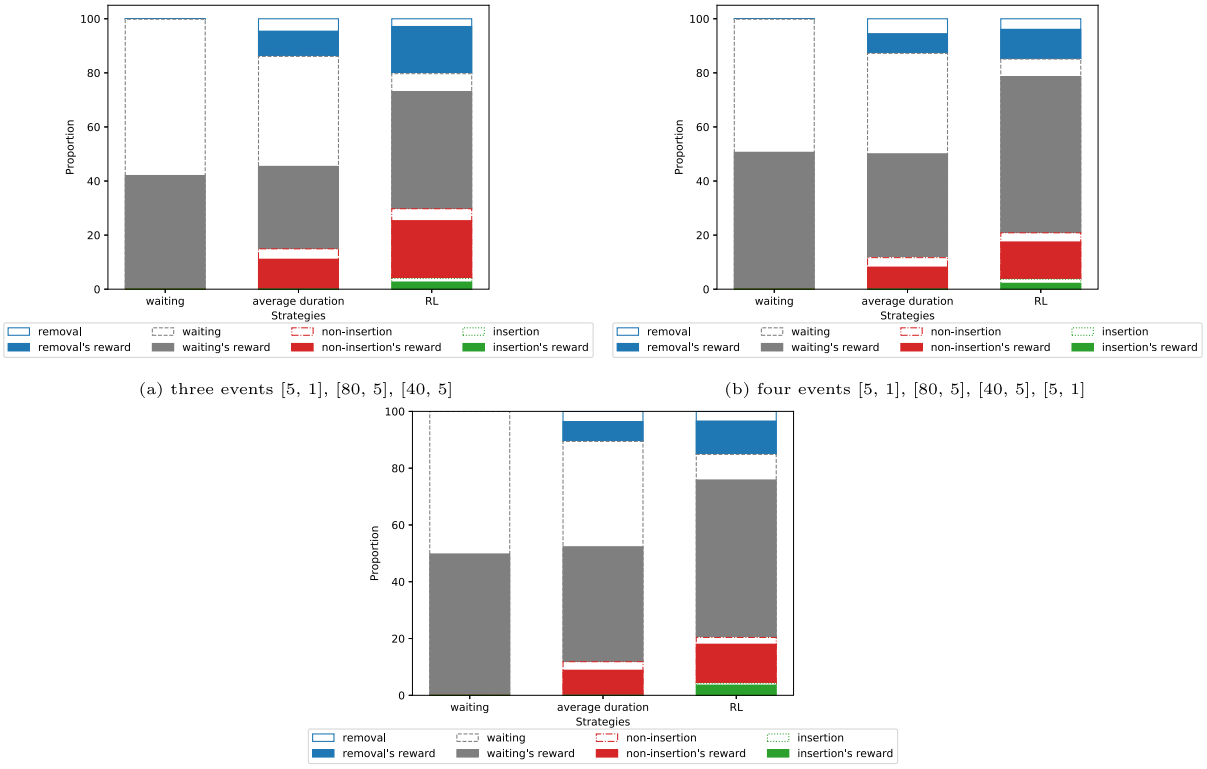
Constraints (39) and (40) ensure vehicles running on suitable and predefined routes, respectively. Constraints (41) ensure the transshipment occurs in the right terminal.

$$x_{ij}^{k} = 0 \quad \forall k \in K_w,\ \forall (i,j) \in A \setminus A_w,\ \forall w \in W \tag{39}$$

$$x_{ij}^{k} = 0 \quad \forall k \in K_{\text{fix}},\ \forall (i,j) \in A \setminus A_{\text{fix}}^{k} \tag{40}$$

$$s_{ir}^{kl} = 0 \quad \forall k \in K_{w_1},\ \forall l \in K_{w_2},\ \forall i \in T \setminus T_{w_1}^{w_2},\ \forall r \in R,\ \forall w_1, w_2 \in W \tag{41}$$

Constraints (42) and (43) ensure that the time on route of barges and trains is consistent with the distance traveled and speed, and Constraints (44) and (45) ensure the time on route of trucks. Constraints (46) and (47) take care of the time windows for pickup



(a) three events [5, 1], [80, 5], [40, 5]

(b) four events [5, 1], [80, 5], [40, 5], [5, 1]

(c) five events [5, 1], [80, 5], [40, 5], [5, 1], [40, 20]

Fig. 28. Proportion of actions under multiple events at the same terminal (supplementary results).

terminals and fixed terminals, respectively.

$$\bar{t}_i^k + \tau_{ij}^k - t_j^k \leqslant M(1 - x_{ij}^k) \quad \forall (i,j) \in A,\ \forall k \in K_{\text{b\&t}} \tag{42}$$

$$\bar{t}_i^k + \tau_{ij}^k - t_j^k \geqslant -M(1 - x_{ij}^k) \quad \forall (i,j) \in A,\ \forall k \in K_{\text{b\&t}} \tag{43}$$

$$\bar{t}_i^{kr} + \tau_{ij}^k - t_j^{kr} \leqslant M(1 - y_{ij}^{kr}) \quad \forall (i,j) \in A,\ \forall k \in K_{\text{truck}} \tag{44}$$

$$\bar{t}_i^{kr} + \tau_{ij}^k - t_j^{kr} \geqslant -M(1 - y_{ij}^{kr}) \quad \forall (i,j) \in A,\ \forall k \in K_{\text{truck}} \tag{45}$$

$$t'^{kr}_{p(r)} \geqslant a_{p(r)} y_{ij}^{kr},\ \bar{t}^{kr}_{p(r)} \leqslant b_{p(r)} + M(1 - y_{ij}^{kr}) \quad \forall (i,j) \in A, \forall r \in R,\ \forall k \in K \tag{46}$$

$$t_i^{kr} \geqslant a_i^k y_{ij}^{kr},\ \bar{t}_i^{kr} \leqslant b_i^k + M(1 - y_{ij}^{kr}) \quad \forall (i,j) \in A,\ \forall r \in R,\ \forall k \in K_{\text{fix}} \tag{47}$$

Constraints (48) to (50) set variables x, y, and z as binary variables.

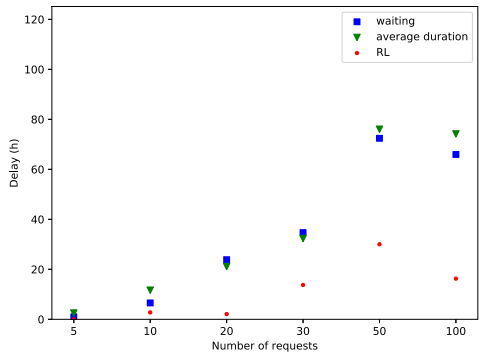
$$x_{ij}^k \in \{0,1\} \quad \forall (i,j) \in A,\ \forall k \in K \tag{48}$$

$$y_{ij}^{kr} \in \{0,1\} \quad \forall (i,j) \in A,\ \forall k \in K,\ \forall r \in R \tag{49}$$

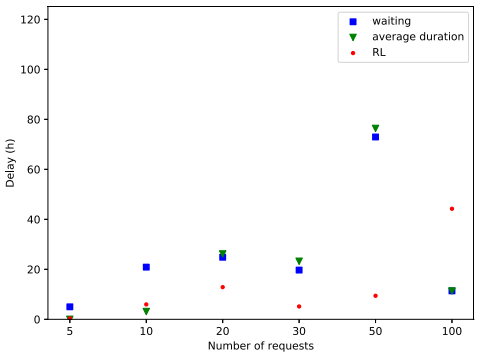
$$z_{ij}^k \in \{0,1\} \quad \forall (i,j) \in A,\ \forall k \in K \tag{50}$$

Appendix B. Results with mixed events at different terminals

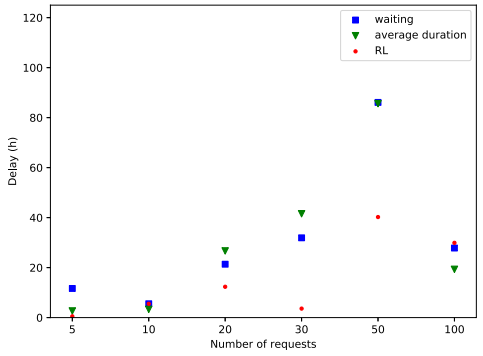
Fig. 24 depicts the distribution of actions and rewards under scenarios in which different types of events occur at different terminals. In Fig. 24(a), the waiting strategy only effectively handles half of the cases in the scenario with both disturbances and severe disturbances. The average duration strategy performs better than the waiting strategy as it uses historical information, although it is still less accurate and yields fewer rewards compared to the learning strategy. In Fig. 24(b), the performance of the waiting and average duration strategies is similar to that observed in Fig. 24(a). In Figs. 24(c) and 24(d), when disruptions occur at some terminals, the waiting strategy's performance deteriorates significantly as it is unable to avoid delays caused by disruptions in the majority of cases. In contrast, the learning strategy is able to handle a mixture of disruptions and disturbances effectively, utilizing a range of actions appropriately based on the specific circumstances it encounters, and consistently earning the highest rewards.



(a) three events [5, 1], [80, 5], [40, 5]

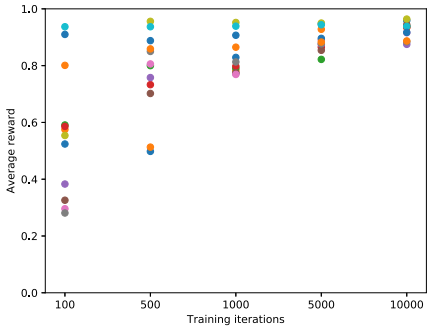


(b) four events [5, 1], [80, 5], [40, 5], [5, 1]

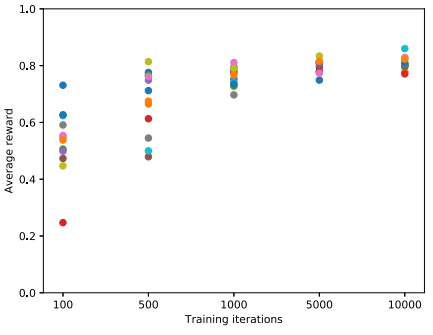


(c) five events [5, 1], [80, 5], [40, 5], [5, 1], [40, 20]

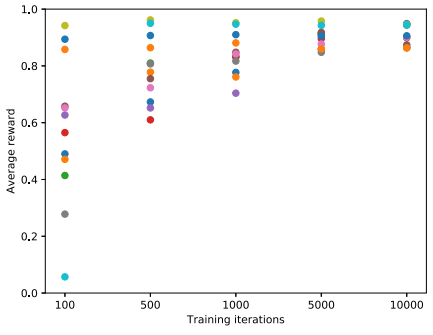
Fig. 29. Total delay over all requests under multiple events at the same terminal (supplementary results).



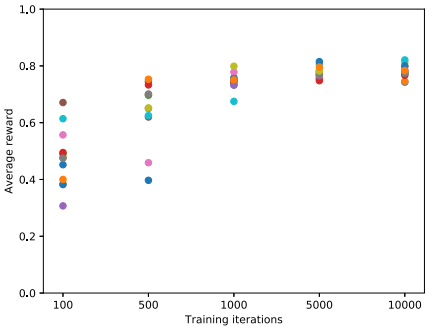
(a) two events, probability of imperfect levels: 0.3



(b) six events, probability of imperfect levels: 0.3



(c) two events, probability of imperfect levels: 0.4



(d) six events, probability of imperfect levels: 0.4

Fig. 30. Average rewards under multiple events with imperfect severity levels (supplementary results, 12 cases with different colors).

Fig. 25 presents the total delay over all requests under scenarios in which different terminals experience different types of events. In 21 out of 24 cases, the RL strategy outperforms the other two strategies, while in the remaining three cases, the RL strategy performs comparable to the waiting or average duration strategy that has a better performance. Despite the presence of various types of events, including disturbances, severe disturbances, and disruptions, at different terminals, the RL strategy is able to effectively identify and implement strategies to avoid delay based on the terminal and its accumulated experiences at that terminal. The results indicate that when either the waiting or average duration strategy is the best-performing strategy, the RL strategy is able to obtain similar results. In cases where these strategies are not optimal, the RL strategy is able to devise a superior approach. On average, the RL strategy reduces delay compared to the average duration strategy by 22.1% and the waiting strategy by 73.8%.

Appendix C. Supplementary results for Section 5.2

Figs. 26 to 30 provide supplementary results for Figs. 13 to 17, respectively.

References

Abbassi, A., El hilali Alaoui, A., Boukachour, J., 2019. Robust optimisation of the intermodal freight transport problem: Modeling and solving with an efficient hybrid approach. J. Comput. Sci. 30, 127–142.

Balaji, B., Bell-Masterson, J., Bilgin, E., Damianou, A., Garcia, P.M., Jain, A., Luo, R., Maggiar, A., Narayanaswamy, B., Ye, C., 2019. ORL: reinforcement learning benchmarks for online stochastic optimization problems. arXiv preprint arXiv:1911.10641.

Basso, R., Kulcsár, B., Sanchez-Diaz, I., Qu, X., 2022. Dynamic stochastic electric vehicle routing with safe reinforcement learning. Transp. Res. Part E: Logist. Transp. Rev. 157, 102496.

Bent, R., Van Hentenryck, P., 2005. Online stochastic optimization without distributions. In: Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS), Vol. 5. Monterey, California, USA, pp. 171–180.

Contargo, 2021. Transport services of contargo. https://www.contargo.net/en/transport/, [Online; accessed 20-June-2023].

Delbart, T., Molenbruch, Y., Braekers, K., Caris, A., 2021. Uncertainty in intermodal and synchromodal transport: Review and future research directions. Sustainability 13 (7), 3980.

Demir, E., Burgholzer, W., Hrušovskỳ, M., Arıkan, E., Jammernegg, W., Van Woensel, T., 2016. A green intermodal service network design problem with travel time uncertainty. Transp. Res. B 93, 789–807.

Dubey, S.R., Singh, S.K., Chaudhuri, B.B., 2022. Activation functions in deep learning: A comprehensive survey and benchmark. Neurocomputing 503, 92–108.

EGS, 2021. Transport services of EGS. https://www.europeangatewayservices.com/en/services/intermodal-services, [Online; accessed 22-February-2022].

Farahani, A., Genga, L., Dijkman, R., 2021. Tackling uncertainty in online multimodal transportation planning using deep reinforcement learning. In: Proceedings of the International Conference on Computational Logistics. Springer, Enschede, the Netherlands, pp. 578–593.

Gabrel, V., Murat, C., Thiele, A., 2014. Recent advances in robust optimization: An overview. European J. Oper. Res. 235 (3), 471–483.

Giusti, R., Manerba, D., Bruno, G., Tadei, R., 2019. Synchromodal logistics: An overview of critical success factors, enabling technologies, and open research issues. Transp. Res. Part E: Logist. Transp. Rev. 129, 92–110.

Gosavi, A., 2009. Reinforcement learning: A tutorial survey and recent advances. INFORMS J. Comput. 21 (2), 178–192.

Guo, W., Atasoy, B., Beelaerts Van Blokland, W., Negenborn, R.R., 2021. Global synchromodal transport with dynamic and stochastic shipment matching. Transp. Res. Part E: Logist. Transp. Rev. 152, 102404.

Guo, W., Atasoy, B., Negenborn, R., 2022. Global synchromodal shipment matching problem with dynamic and stochastic travel times: A reinforcement learning approach. Ann. Oper. Res. 2022, 1–32.

Hildebrandt, F.D., Thomas, B., Ulmer, M.W., 2021. Where the action is: let's make reinforcement learning for stochastic dynamic vehicle routing problems work!. arXiv preprint arXiv:2103.00507.

Hrušovskỳ, M., Demir, E., Jammernegg, W., Van Woensel, T., 2021. Real-time disruption management approach for intermodal freight transportation. J. Clean. Prod. 280, 124826.

James, J., Yu, W., Gu, J., 2019. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. 20 (10), 3806–3817.

Laporte, G., Louveaux, F., Mercure, H., 1992. The vehicle routing problem with stochastic travel times. Transp. Sci. 26 (3), 161–170.

Li, L., Negenborn, R.R., De Schutter, B., 2015. Intermodal freight transport planning–A receding horizon control approach. Transp. Res. C 60, 77–95.

Li, L., Negenborn, R.R., De Schutter, B., 2017. Distributed model predictive control for cooperative synchromodal freight transport. Transp. Res. Part E: Logist. Transp. Rev. 105, 240–260.

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

Nazari, M., Oroojlooy, A., Snyder, L., Takác, M., 2018. Reinforcement learning for solving the vehicle routing problem. Adv. Neural Inf. Process. Syst. 31, 1–11.

Öncan, T., Altınel, İ.K., Laporte, G., 2009. A comparative analysis of several asymmetric traveling salesman problem formulations. Comput. Oper. Res. 36 (3), 637–654.

Pan, W., Liu, S.Q., 2023. Deep reinforcement learning for the dynamic and uncertain vehicle routing problem. Appl. Intell. 53, 405–422.

Phiboonbanakit, T., Horanont, T., Huynh, V.-N., Supnithi, T., 2021. A hybrid reinforcement learning-based model for the vehicle routing problem in transportation logistics. IEEE Access 9, 163325–163347.

Qu, W., Rezaei, J., Maknoon, Y., Tavasszy, L., 2019. Hinterland freight transportation replanning model under the framework of synchromodality. Transp. Res. Part E: Logist. Transp. Rev. 131, 308–328.

Rivera, A.E.P., Mes, M.R., 2017. Anticipatory freight selection in intermodal long-haul round-trips. Transp. Res. Part E: Logist. Transp. Rev. 105, 176–194.

Shobayo, P., Nicolet, A., Van Hassel, E., Atasoy, B., Vanelslander, T., 2021. Conceptual development of the logistics chain flow of container transport within the rhine-alpine corridor. In: Proceedings of the European Transport Conference (ETC), 13-15 September, 2021. pp. 1–17.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al., 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 362 (6419), 1140–1144.

Soltani-Sobh, A., Heaslip, K., Stevanovic, A., El Khoury, J., Song, Z., 2016. Evaluation of transportation network reliability during unexpected events with multiple uncertainties. Int. J. Disaster Risk Reduct. 17, 128–136.

Srinivasan, K.K., Prakash, A., Seshadri, R., 2014. Finding most reliable paths on networks with correlated and shifted log–normal travel times. Transp. Res. B 66, 110–128.

SteadieSeifi, M., Dellaert, N.P., Nuijten, W., Van Woensel, T., Raoufi, R., 2014. Multimodal freight transportation planning: A literature review. European J. Oper. Res. 233 (1), 1–15.

SteadieSeifi, M., Dellaert, N., Van Woensel, T., 2021. Multi-modal transport of perishable products with demand uncertainty and empty repositioning: A scenario-based rolling horizon framework. EURO J. Transp. Logist. 10, 100044.

Tavasszy, L., Behdani, B., Konings, R., 2017. Intermodality and synchromodality. In: Ports and Networks. Routledge, pp. 251–266.

Van Riessen, B., Negenborn, R.R., Dekker, R., 2016. Real-time container transport planning with decision trees based on offline obtained optimal solutions. Decis. Support Syst. 89, 1–16.

Van Riessen, B., Negenborn, R.R., Lodewijks, G., Dekker, R., 2015. Impact and relevance of transit disturbances on planning in intermodal container networks using disturbance cost analysis. Marit. Econ. Logist. 17 (4), 440–463.

Yee, H., Gijsbrechts, J., Boute, R., 2021. Synchromodal transportation planning using travel time information. Comput. Ind. 125, 103367.

Zhang, Y., Atasoy, B., Negenborn, R.R., 2022a. Preference-based multi-objective optimization for synchromodal transport using adaptive large neighborhood search. Transp. Res. Rec. 2676 (3), 71–87.

Zhang, Y., Guo, W., Negenborn, R.R., Atasoy, B., 2022b. Synchromodal transport planning with flexible services: Mathematical model and heuristic algorithm. Transp. Res. C 140, 103711.

Zhang, Y., Heinold, A., Meisel, F., Negenborn, R.R., Atasoy, B., 2022c. Collaborative planning for intermodal transport with eco-label preferences. Transp. Res. Part D: Transp. Environ. 112, 103470.

Zhang, Y., Li, X., Van Hassel, E., Negenborn, R.R., Atasoy, B., 2022d. Synchromodal transport planning considering heterogeneous and vague preferences of shippers. Transp. Res. Part E: Logist. Transp. Rev. 164, 102827.