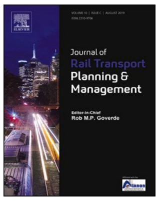
Journal of Rail Transport Planning & Management 32 (2024) 100483

Contents lists available at ScienceDirect

Journal of Rail Transport Planning & Management

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



A reinforcement learning approach to solving very-short term train rescheduling problem for a single-track rail corridor

Jin Liu [*], Zhiyuan Lin, Ronghui Liu

Institute for Transport Studies, University of Leeds, Leeds, LS2 9JT, United Kingdom

A R T I C L E I N F O

Keywords:
Reinforcement learning
Q learning
Train rescheduling
Single-track
Railway traffic management

A B S T R A C T

Railway operations are regularly affected by incidents such as disturbances and disruptions, which cause temporary operational restrictions to the trains in the network. Compared to real-time disturbances and disruptions, sometimes these incidents can be known at a short notice, e.g., 24–48 h beforehand, which is known as the Very-Short-Term-Planning in British rail operations. This paper presents a novel reinforcement learning based approach for rescheduling train services for in a single-track corridor with bi-directional traffic. As an important subfield of machine learning, reinforcement learning offers an alternate strategy for tackling the NP-hard train (re)scheduling problems and shows its advantages in balancing computational efficiency and solution quality. We propose a Q-learning approach with a tiered rewarding strategy and lightweight train representation in state vectors, which enables more efficient learning and knowledge sharing among homogeneous trains. Compared with an existing reinforcement learning approach, our proposed method can find better quality solutions due to its unique representation of state vectors and a novel tiered rewarding/punishing mechanism, overcoming certain disadvantages in existing approaches. Knowledge reusability is another advantage of the proposed approach, as prior knowledge obtained from training one instance can significantly enhance the performance of another, potentially more challenging, instance on the same corridor with substantially reduced computational time and effort on algorithm development. We also discuss the potential applications of the knowledge reusability feature inherent in reinforcement learning algorithms, which we believe will benefit the entire industry in addressing NP-hard problems through data-driven technologies.

1. Introduction

The railway timetable is usually designed well in advance, prioritizing system robustness and resilience while optimizing the scheduling of its resources (Carey and Lockwood, 1995). However, during daily operations, the planned schedules are not always strictly followed due to nearly inevitable disturbances/disruptions of varying scales, causing trains to run behind schedule. Without prompt and proper handling, the delay of a single train would affect the operation of other trains and rapidly spread across the whole corridor and/or network. Under these circumstances, the objective of train rescheduling is to generate a workable (and likely near-optimal) schedule based on the original timetable and disturbance/disruption information to minimize the impact of disturbance/disruption as much as possible. The rescheduling action can be accomplished by adjusting the arrival and departure times of

* Corresponding author.
E-mail address: J.Liu12@Leeds.ac.uk (J. Liu).

https://doi.org/10.1016/j.jrtpm.2024.100483
Received 7 May 2024; Received in revised form 20 July 2024; Accepted 20 September 2024
Available online 25 September 2024

the timetabled trains, and if necessary, may also involve extra train services, cancellation of existing train services (Hong et al., 2021), modification of stopping patterns (Zhan et al., 2022), and subsequent tasks such as the reallocation of rolling stock (Lin and Kwan, 2016) and crew (Veelenturf et al., 2014), adjustments on train speed profiles and energy consumption (Dong et al., 2023, 2024), etc.

In general, rescheduling must be performed in short time beforehand, and the affordable time window for presenting reschedule plan may range from days (short-term), 24–48 h (very short-term) and a few hours to minutes (real-time), necessitating varying degrees of computation speeds and optimality requirements in operational principles of British railway (RSSB, 2022). Depending on the attributes of a railway network and train services the goal of train rescheduling may involve minimizing train terminating delay, minimizing the number of delayed trains, reducing the propagation of train delay, minimizing the absolute, peak or average delay in the network, etc.

Since the majority of train (re)scheduling problems are NP-hard (Cai and Goh, 1994), inexact approaches such as heuristics are frequently employed to speed up the algorithm at the expense of solution quality. The robustness and transferability of heuristic approaches may also suffer. If solved by exact methods, scalability usually becomes the biggest obstacle when the problem size gets large (Khadilkar, 2019). No universally applicable remedy is available for this dilemma between exact and inexact methods in optimization. Reinforcement learning, a significant subfield of machine learning, offers a promising strategy for addressing train (re) scheduling problems by striking a balance between precise approaches and heuristic methods. The fundamental principle of reinforcement learning algorithms is the accumulation of knowledge through a trial-and-error process within an environment. This process can be time-consuming, particularly for problems with NP-hard characteristics. However, once an agent is adequately trained, reinforcement learning algorithms can make optimal or near-optimal decisions in new environments with short computational times. This efficiency is achieved without being significantly influenced by the problem size, which distinguishes reinforcement learning from exact methods that often struggle with scalability (Sutton and Barto, 2018).

This article focuses on the train rescheduling problem typically given 24–48 h beforehand, so-called Very Short Term Rescheduling (VSTR), and proposes a Q learning-based reinforcement learning approach to adjust arrival and departure times of existing services and/or give cancellation (including early termination at intermediate stations) decisions to specific service(s). This kind of problem is common when incidents/events can be known shortly in advance, e.g., adverse weather forecasted, planned engineering work, sports events, and so forth. This train rescheduling problem has a longer time window compared to real-time rescheduling, but still, the reschedule has to be produced as quickly as possible for the decision makers. Compared with existing reinforcement learning applications, the key contributions of this paper are.

- ***Tiered rewarding mechanism:*** The implementation of a two-tiered reward system aims to more accurately assess the quality and individual contribution of each train in both successful and unsuccessful scenarios. This approach allows for a clear differentiation between exclusive reward values and the selection of relevant decision history excerpts for each individual train agent, enabling 'more targeted' learning.
- ***Lightweight train identifier:*** Using train service direction as identifiers in state vectors, rather than individual train IDs, reduces the state space's dimensionality in the reinforcement learning algorithm and facilitates knowledge sharing among homogeneous trains.
- ***Knowledge reusability:*** Prior knowledge obtained from training an earlier VSTR instance is proved to be reusable to another (likely more difficult) instance on the same corridor, and significantly reduces learning efforts. This feature enables the reinforcement learning algorithm to draw from the accumulated knowledge of past problems, thus reducing the trial-and-error phase and ultimately expediting the learning process when confronted with new issues.
- ***Delay times are not given as input for the model:*** Instead, based on indirect information (e.g., adverse weather), our model will determine the delay time or termination for each train during the rescheduling process.

The rest of this paper is organized as follows. Section 2 reviews the state of the art of railway rescheduling research. Section 3 provides a common description of VSTR problem. In Section 4, we propose reinforcement learning approach to solve VSTR problem. Section 5 presents the case studies and associated computational experiments based on proposed method in real-world instances. Finally, Section 6 concludes the key observations of this paper and our future works.

2. Literature review

Train scheduling and rescheduling are conventionally formulated as mixed-integer programming (MIP) problems. Binary variables typically represent the order of trains entering sections, while continuous variables describe departure and arrival times (Jovanovic and Harker, 1991; Carey, 1994a; Higgins et al., 1996, 1997). Most problems addressed bidirectional railway systems (Carey, 1994a, 1994b) with constraints including minimum headways, speed limits, and specific departure/arrival times. Solutions vary from exact methods (Higgins et al., 1996) to heuristics, such as genetic algorithm (Wang et al., 2019; Liu et al., 2019), swarm intelligence (Eaton et al., 2017), etc.

Fixed block system-based scheduling models incorporate constraints ensuring no two trains occupy the same block simultaneously. Additional constraints, such as headways between train exits and entrances in block sections, are necessary (Kraay et al., 1991; Törnquist and Persson, 2007). These problems can also be modelled as job-shop problems, using alternative graph models for conflict-free scheduling (D'Ariano et al., 2007, 2008a, 2008b; Corman et al., 2010). Particularly for train rescheduling under disruptions, heuristic approaches like Particle Swarm Optimization aim to minimize secondary delays (Wang et al., 2019). Other MIP models focus on real-time rescheduling under severe disruptions, minimizing train-deviation costs (Zhan et al., 2022) and delays (Wang, 2019) Heuristic approaches for train (re)scheduling include binary variables for train stop decisions (Cai and Goh, 1994) and conflict resolution using Genetic Algorithm (Dündar and Şahin, 2013). Parallel algorithms improve real-time rescheduling efficiency (Josyula et al., 2018). The discrete-event approach simulates train movements, organizing overtaking and passing plans to avoid deadlocks and minimize delays, however, this method, although effective, struggles to enumerate all possible events for decision-making (Li et al., 2014).

However, both exact and heuristic algorithms have their limitations when addressing the complexities of train rescheduling problems. Exact algorithms encounter scalability issues; as the problem size grows (e.g., with more trains, tracks, and time intervals) the computational effort required increases exponentially, making these methods impractical for large-scale scenarios. Conversely, heuristic algorithms face constraints due to their design. They often focus on exploiting known good solutions and can become trapped in local optima, leading to suboptimal outcomes, especially in complex, large-scale settings. In recent years, reinforcement learning (RL) is gaining traction as an alternative to optimization methods for train (re)scheduling due to its ability to handle complex, dynamic environments where traditional methods struggle (Bešinović et al., 2021; Melnikov et al., 2024). RL algorithms learn optimal policies by interacting with the system and receiving feedback, enabling them to adapt to varying conditions and uncertainties. This adaptability makes RL suitable for real-time decision-making and dynamic problem-solving (Agasucci et al., 2023). Additionally, RL can improve performance over time by continuously learning from past experiences, providing scalable solutions that adjust to changes and complexities in train scheduling (Cui et al., 2016).

Obara et al. (2018) proposed an approach based on DQN and deep RL, where an agent adjusts running times and departure sequences to maximize passenger satisfaction and minimize total delays. This model, designed for double-track lines, simplifies traffic and conflict management compared to single-track lines. Šemrov et al. (2016) used Q-learning for rescheduling trains affected by single-track disruptions in Slovenia, considering both reordering and retiming with actions specified as signal controls and states represented by track capacity, train positions, and time. Ning et al. (2019) developed a deep RL method to minimize average delays along a rail line, incorporating detailed train operations in block sections and stations. The agent modifies running times, dwell periods, and departure orders, handling conflicts through train sequencing. This method was applied to the Beijing-Shanghai High-speed Railway, a double-track line.

Khadilkar (2019) presented a Q-table based RL method for scheduling single-track lines, specifying track assignments and arrival/departure times for all trains to minimize weighted delays. This method decouples the size of the state-action space from the

Table 1
Notations used in this article.

Symbol	Description
$\mathscr{E} = (G, \mathscr{M}, \mathscr{D}, \mathscr{C})$	An environment $\mathscr{E}$ representing the dynamics of a single-track corridor system
$G = (N, L)$	A path graph corresponding to a macroscopic level of a corridor single-line railway with stations and segments
N, n	Set, and index of nodes (stations), $n \in N$
L, l	Set, and index, of segments (directionless track sections between stations), $l \in L$. We also use them to denote links (directional sections between stations) when no ambiguity arises.
$M_k(t) \in \mathscr{M}$	Train movement property (including node/link available capacity $\boldsymbol{\alpha}_k(t)$ and dwell/travel time δ_k) information of a node/link k at time t, $k \in N$ or L. $\mathscr{M}$ contains all such information over all nodes/links and timestamps.
$\mathscr{T}, \tau$	Set, and index, of trains, $\tau \in \mathscr{T}$
$C_\tau(t)$	Train characteristic information of train τ at time stamp t
$\mathscr{C}$	Set of train characteristic information, $C_\tau \in \mathscr{C}$
δ_n	Dwelling time of a certain train at node n
δ_l	Travelling time of a certain train at link l
δ_l^D	Travelling time of a certain train at link l considering disturbance D
$\delta_n^{\tau,\tau'}$	Headway between train τ and train τ'
D_i	Individual disturbance information
$\mathscr{D}$	Set of disturbance information, $D_i \in \mathscr{D}$
$D_\tau(t)$	Disturbance information for train τ at time t. $D_\tau(t)$ can be a D_i or a null value.
$P_\tau(t)$	Overall property information for train τ at time t, including train direction, neighbouring node/link available capacity $\boldsymbol{\alpha}_\tau(t)$ and related disturbances $w(D_\tau(t))$.
$\mathscr{P}$	Set of all possible $P_\tau(t) \in \mathscr{P}$, which stores accumulated experienced states over the past episodes and thus defines the Q table used later.
$\boldsymbol{s}_\tau^n$	State vector of train τ at station n
S	Set of states, $\boldsymbol{s}_\tau^n \in S$
$a \in A$	Action and set of actions, where $A = \{a_1, a_2, a_3\}$. $a_1 = halt, a_2 = go, a_3 = terminate.$
e, E	Episodes and total number of episodes so far
H^e	Accumulated decision history in episode e
H_τ^e	Accumulated decision history of train τ in episode e
a_s	Decision (as a chosen action) made at state s
Q	Q table accumulated through reinforcement learning algorithm
$q_{(s,a)}$	Q value corresponding to state-action pair $(\boldsymbol{s}, a)$
ϵ	Threshold value of action decision-making policy
R_τ	Overall reward value for train τ
r_τ	Reward value to quantify train τ's individual performance
r_e	Reward value to quantify global performance of episode e
$t_a^Q(\tau,n,e), t_d^Q(\tau,n,e)$	Rescheduled arrival and departure times of train τ at station n from Q learning in episode e
$t_a(\tau,n), t_d(\tau,n)$	Arrival and departure time of train τ at station n in original timetable

problem size, ensuring efficient scalability. States are defined by resource consumption near each train, and actions determine whether a train should wait or proceed. The number of states can be large when many trains are involved, and backward propagation is approximated by averaging recent states. Zhu et al. (2020) proposed an RL-based method that learns to reschedule a timetable offline and applies it online to create optimized dispatching plans. Similar action and state representations to Khadilkar (2019) are used, assuming each train has a pre-defined tentative delay time and generating a new timetable to minimize overall delays.

Ying et al. (2020) developed an actor-critic method for metro train scheduling with restricted rolling stock, formulating the problem as an MDP governed by stochastic passenger demand. Using artificial neural networks to parameterize state and action spaces, the framework facilitates the assessment and search for optimum solutions. A deep deterministic policy gradient algorithm trains neural networks through simulated system transitions before implementing the actor-critic agent for live schedule control. The method was evaluated using the Victoria Line, London Underground. Ying et al. (2022) extended this research by adding a multi-agent feature for controlling various service lines in a metro network with passenger transfer. Each control agent is associated with a critic function for estimating future states and an actor function for deriving operational decisions. The framework was tested on the Bakerloo and Victoria Lines, London Underground. Liu and Liu (2023) designed an actor-critic reinforcement learning algorithm to address the train insertion problem in mixed-use rail networks. In this approach, the actor agent is trained to make new timetable decisions for newly inserted trains, while the critic agent ensures overall operational robustness. Yue et al. (2024) employed a policy-based reinforcement learning approach to tackle the real-time rescheduling problem in high-speed rail systems. They proposed an action sampling strategy to select actions, aiming to achieve more efficient and effective exploration, considering the NP-hard nature of the rescheduling problem.

In this paper, we propose a novel reinforcement learning based approach for the VSTR in a single-track corridor. The model can be easily modified into the double-track scenarios, which have a simpler structure than single-track. As far as the authors are aware, this is the first time that reinforcement learning is applied to solve train rescheduling problem based on short-noticed information on disturbances/disruptions themselves (e.g., adverse weather, engineering work, etc.).

3. Problem statement

In this section, we describe the VSTR problem and formulate it with a mixed integer programming (MIP) approach. The scope of the VSTR issue is to address the problem of rescheduling trains within 24–48 h before the timetable is executed, taking into account known hazards in the network that could impact the implementation of the existing timetable. In VSTR problem, the key considered hazards are deterministic disturbance or/and disruptions (such as forecasted bad weather, ad-hoc engineering activities and maintenance of infrastructure, etc.), and thus, its impacts to the existing timetable and its operation is deterministic which can be fully anticipated and considered in the decision making, and thus, the target of the solutions for VSTR is to adjust the existing timetable to satisfy forecasted demand as much as possible. Table 1 summarizes the notations used in this article.

3.1. *Modelling single-track railway environment and disturbance*

In this study, railway system is modelled at a macro level, excluding features such as signalling system, train length, and train acceleration/deceleration. Corridor-based single-track railway is represented by a path graph $G = (N, L)$ with nodes N and links L. l_n is a directionless segment of rail connecting two stations, n and $n+1$. Trains travelling in the same or opposite directions are not permitted to share l_n. When travel direction is concerned, segment l_n is implicitly associated with two directional links: the outgoing connection c_n connects station n to station $n+1$, and the inbound link $\overline{c}_n$ connects station $n+1$ to station n. Outbound train i travels in the opposite direction of inbound train j. Under a context without confusion, we use the terms "segment" (directionless) and "link" (directional) interchangeably both represented by $l \in L$. We use t to index the times. Note that the time is discretized in a resolution of minutes. Fig. 1 illustrates an example of the representation.

Based on the path graph $G = (N, L)$, additional property features are needed to model the movements of the trains and infrastructure occupancy in the line. The property information on link l (or node n) at time t is denoted as $M_l(t) = [\alpha_l, \delta]$ (or $M_n(t) = [\alpha_n, \delta]$ respectively), which includes available capacity α, a dynamic property of the number of resources (node/link space) available, and travel or dwell time δ needed on the node or link (a travel/dwell time dynamic property). As a result, train movement activities within path graph G can be identified by values in $M_k(t)$. States of nodes and links are defined as their available capacity $\boldsymbol{\alpha}_k(t)$. The last component included into the environment is for disturbances in the rail network.

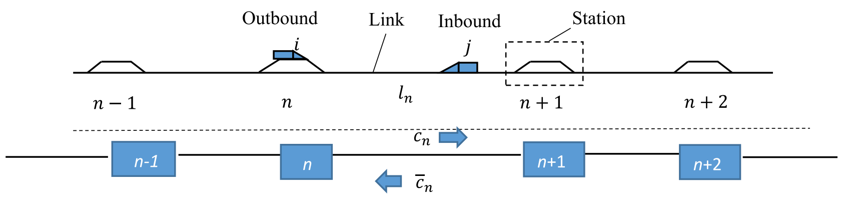


Fig. 1. A schematic of single-track corridor railway (top) and its mapped path graph (bottom) with stations as nodes and segments as links. The stations are consecutively numbered from 1 to $|N|$.

Remarks. Without causing ambiguity, we use disturbances hereafter for both disruptions and disturbances considered in the time scope of VSTR problem, and any other incident that will affect the operations of trains by imposing a reduced speed limit, extra dwelling, partially/full cancellation of the train service, etc.

The disturbances within VSTR problem usually include ad-hoc engineering work, short term maintenance work, adverse weather, etc. which place temporary speed restrictions or closure on specific areas. Such temporary speed-limits could be imposed on different parts and during different time periods, depending on where and when the impact of the disturbances is detected. Let $\mathscr{D} = \{D_1, D_2, \dots\}$ be the set of all such disturbances, where $D_i =$ [affected link ID, disturbance start-time, disturbance end-time, adjusted speed as a percentage of the original speed]. The disturbances information for train τ at time t, $D_\tau(t)$, can be determine by mapping the disturbance events onto the scheduled train path: if the train path runs through a disturbance event, the train's running speed through that part of the track/network would follow the reduced speed limit.

Finally, trains form a component of the environment. The trains are characterised by their IDs, direction of travel indicated as a binary attribute d_τ, their route ρ_τ (including stopping patterns), their original and rescheduled arrival and departure times at each station, and the associated station dwell time and link travel time. These information are represented in set $C_\tau \in \mathscr{C}$ for train τ, where $C_\tau = \left[ID, d_\tau, \rho_\tau, t_a(\tau,n), t_d(\tau,n), t_a^Q(\tau,n), t_d^Q(\tau,n), \delta_n, \forall n, \delta_l, \forall l \in L\right]$. The inclusion of $\mathscr{C}$ completes the construction of the environment $\mathscr{E} = (G, \mathscr{M}, \mathscr{D}, \mathscr{C})$ ready for the Q-learning process.

3.2. Mathematical formulation of VSTR

The VSTR problem is defined as, considering disturbance $D, D \in \mathscr{D}$ to generate a feasible timetable for all the services of the day, $\left\{t_a^Q(\tau,n), t_d^Q(\tau,n)\right\}, \tau \in \mathscr{T}, n \in N$ to optimize and replace the existing timetable $\{t_a(\tau,n), t_d(\tau,n)\}, \tau \in \mathscr{T}, n \in N$ to satisfy the specific quality of the services. In this study, the target of making a train rescheduling decision is to minimize the impact of delay in every dwelling station across the trains' journey and these dwelling station should have same priority in the control target, so the optimization objective is set as to minimize the deviation of the rescheduled timetable to original one, and we set the objective function of the optimization as total accumulated departure delays (TAD) of all the trains (see equation (1)). It should be noted that the mathematical formulation is not essential for organizing the reinforcement learning algorithm. The purpose of this MILP formulation is to demonstrate the common objective and the constraints imposed by infrastructure and operations.

$$TAD = \sum_{n \in N} \sum_{\tau \in \mathscr{T}} \left(t_d^Q(\tau,n) - t_d(\tau,n)\right) \tag{1}$$

The objective function is optimized with respect to the following constraints:

$$t_a^Q(\tau,n) \ge t_a(\tau,n), \forall\, \tau \in \mathscr{T}, n \in N \tag{2}$$

$$t_d^Q(\tau,n) \ge t_d(\tau,n), \forall\, \tau \in \mathscr{T}, n \in N \tag{3}$$

$$t_a^Q(\tau,n+1) \ge t_d^Q(\tau,n) + \delta_{l(n,n+1)}^D, \forall\, \tau \in \mathscr{T}, n \in N \tag{4}$$

$$t_d^Q(\tau,n) \ge t_a^Q(\tau,n) + \delta_n^D, \forall\, \tau \in \mathscr{T}, n \in N \tag{5}$$

$$t_d^Q(\tau',n) + M \times \left(1 - i_n^{\tau,\tau'}\right) \ge t_d^Q(\tau,n) + \delta_n^{\tau,\tau'}, \forall d_\tau = d_{\tau'}, \tau \in \mathscr{T}, n \in N \tag{6}$$

$$t_d^Q(\tau,n) + M \times i_n^{\tau,\tau'} \ge t_d^Q(\tau',n) + \delta_n^{\tau,\tau'}, \forall d_\tau = d_{\tau'}, \tau \in \mathscr{T}, n \in N \tag{7}$$

$$t_a^Q(\tau',n) + M \times \left(1 - i_n^{\tau,\tau'}\right) \ge t_a^Q(\tau,n) + \delta_n^{\tau,\tau'}, \forall d_\tau = d_{\tau'}, \tau \in \mathscr{T}, n \in N \tag{8}$$

$$t_a^Q(\tau,n) + M \times i_n^{\tau,\tau'} \ge t_a^Q(\tau',n) + \delta_n^{\tau,\tau'}, \forall d_\tau = d_{\tau'}, \tau \in \mathscr{T}, n \in N \tag{9}$$

$$t_d^Q(\tau',n) + M \times \left(1 - i_n^{\tau,\tau'}\right) \ge t_a^Q(\tau,n) + \delta_n^{\tau,\tau'}, \forall d_\tau \ne d_{\tau'}, \tau \in \mathscr{T}, n \in N \tag{10}$$

$$t_d^Q(\tau,n) + M \times i_n^{\tau,\tau'} \ge t_a^Q(\tau',n) + \delta_n^{\tau,\tau'}, \forall d_\tau \ne d_{\tau'}, \tau \in \mathscr{T}, n \in N \tag{11}$$

$$t_a^Q(\tau',n) + M \times \left(1 - i_n^{\tau,\tau'}\right) \ge t_d^Q(\tau,n) + \delta_n^{\tau,\tau'}, \forall d_\tau \ne d_{\tau'}, \tau \in \mathscr{T}, n \in N \tag{12}$$

$$t_a^Q(\tau,n) + M \times i_n^{\tau,\tau'} \ge t_d^Q(\tau',n) + \delta_n^{\tau,\tau'}, \forall d_\tau \ne d_{\tau'}, \tau \in \mathscr{T}, n \in N \tag{13}$$

$$\delta_n^{\tau,\tau+1} = \begin{cases} \delta_1 \text{ min, if } d_\tau = d_{\tau+1} \\ \delta_2 \text{ min, otherwise} \end{cases} \tag{14}$$

$$\alpha_l(t) = \{0, 1\} \tag{15}$$

$$\alpha_n(t) = \{0, 1, \ldots, |A_n|\} \tag{16}$$

$$i_n^{\tau,\tau'} = \{0, 1\} \tag{17}$$

Equations (2) and (3) impose constraints on the arrival and departure times of the rescheduled train at station n. Specifically, these equations ensure that the rescheduled train arrival and departure times at station n are not earlier than the originally planned arrival and departure times, respectively. Equation (4) describes the train travelling constraints from station n to station $n+1$, considering a certain regional disturbance D, i.e., the travelling time between n and $n+1$ should not smaller than the estimated travelling time $\delta^D_{l(n,n+1)}$. Equation (5) is the dwelling constraints which indicates train τ at station n shall satisfy the dwelling time of δ^D_n if the station is covered by disturbance D. Equations (6)–(9) represent the headway constraints for trains travelling in the same direction at stations, where the headway is fixed at 2 min, i.e., $\delta_1 = 2$. To relax these constraints, the big-M method is used. The binary indicator $i_n^{\tau,\tau'} = 1$ indicates that the arrival or departure event of train τ occurs before train τ' at station n, while $i_n^{\tau,\tau'} = 0$ indicates the opposite. M is a large positive value. Equations (10)–(13) establish the headway constraints at station n for trains travelling in opposite directions. Specifically, the entering and leaving events of trains in opposite directions must have a minimum headway of 3 min between them due to dispatching limitations, i.e., $\delta_2 = 3$. Furthermore, Equation (14)–(17) defines the range of the values in this formulation. To generate a solution for a certain VSTR problem, the variables $t_d^Q(\tau, n)$ and indicators $i_n^{\tau,\tau'}$ will be carefully adjusted to minimize the objective function in Equation (1) with respect to the constraints described in Equations (2)–(13).

4. Reinforcement learning algorithm for train rescheduling problem

We introduce here the key elements of our proposed reinforcement learning method to solve train rescheduling problem: the modelling of railway environment and its operation, agent-based modelling approach to deploy and execute decision making process (i.e., the rescheduling solutions), policy function and learning process of the reinforcement learning. For the original algorithm structure of reinforcement learning, readers can refer to Sutton and Barto (2018). In this paper, Q-learning approach is chosen to organize the reinforcement learning algorithm for solving the single-track VSTR.

4.1. *Q-learning agent*

A train-oriented agent modelling approach is used in this study, i.e., the agents are the individual trains whose schedule/movements through the network, in terms of their departure and arrival times at stations, are the subject for learning from the network state information and their own characteristics. Train movements across the whole single-track corridor are modelled as a series of state-action pairs, which provides trains associated decisions at nodes in the corridor.

4.1.1. *Representation of state and state vector*

In this study, information from physical railway network, railway traffic management centre, and other environmental information (e.g., weather forecast) is considered. An overall train property $P_\tau(t)$ with movement information, train characteristics, and disturbance information of the rail section that the train is moving into, is defined as $P_\tau(t) = [C_\tau, M_\tau(t), D_\tau(t)]$. Based on $P_\tau(t)$, we further define the state vector of train τ at time t as a vector $\boldsymbol{s}_\tau(t) = (d_\tau, \boldsymbol{\alpha}_\tau(t), w(D_\tau(t)))$. We introduce how the three terms of the state vector $d_\tau, \boldsymbol{\alpha}_\tau(t), w(D_\tau(t))$ are constructed below.

Train characteristics $C_\tau(t) \in \mathscr{C}$ contains multiple attributes and to form the state vector of a train at a timestamp, only its direction d_τ is included.

Node/link available capacity Local available capacity around a train represented by vector $\boldsymbol{\alpha}_\tau(t)$ with a length of $(b+1+f)$ is used, which includes the available capacity conditions of b spaces behind the train, f spaces ahead of the train, and the space the train is currently occupying. As specified in our assumptions, the two terminal stations have a 'huge' capacity and we indicate it with a value of M. A dummy capacity of NaN is assigned to elements of a state vector that are beyond terminal stations. The availability information that is forwarded to the train agent has a significant impact on the train's movement decision. Additionally, considering the current

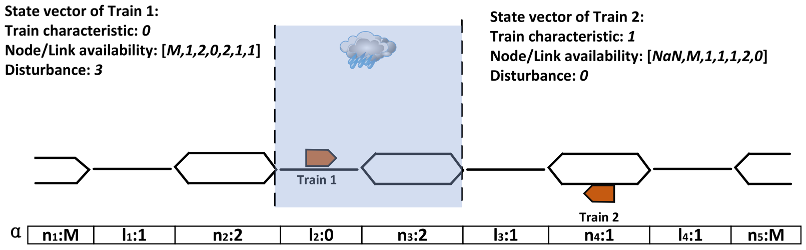


Fig. 2. Example of state vector representation.

section being occupied helps the train agent determine whether there is another train running parallel to the one being considered. The reason for backwardly using capability information with the train agent is that the reinforcement learning algorithm can identify the number of trains waiting behind a specific train at any given time. This feature allows the algorithm to prioritize a particular train if the line behind it is overcrowded. An example of the representation of two running trains on a corridor line between nodes n_1 and n_5, connected by links l_1 to l_4, is shown in Fig. 2. With $b = 3, f = 3$, the state vector based on train 1 currently on link l_2 is represented by the available capacity of node n_4, link l_3 and node n_3 in front of it, the capacity of the resource it currently occupies (link l_2), and capacities of node n_2, link l_1 and node n_1 behind it. Similarly, the state vector of train 2 is marked in Fig. 2, where the third element behind it is beyond the terminal station n_5 and thus has a capacity of *NaN*. Assume the current timestamp is t. Then Train 1 (represented by τ_1) has a state vector $\mathbf{s}_{\tau_1}(t) = (0, [M,1,2,0,2,1,1], 3)$, and Train 2 ($\tau_2$) has a state vector $\mathbf{s}_{\tau_2} = (1, [NaN,M,1,1,1,2,0], 0)$.

Disturbances Disturbance information will be also considered in the state vector which is used to indicate the operational restrictions due to the impacts of disparate conditions of disturbances to train operation. As mentioned in Section 3.1, the disturbance $D_\tau(t)$ is gathered from the node/link the considered train is about to enter, and to be used to support the decision of train on whether to access the next rail section when all the operational constraints at current rail section are satisfied. Remember that $D_\tau(t) = D_i =$ [affected link, disturbance start time, disturbance end time, adjusted speed percentage], if a disturbance is affecting this train, or simply a null value implying not disturbance is observed for train τ at time t. To incorporate it in the state vector, a transformation function w is designed to convert $D_\tau(t)$ to a pre-defined non-negative integer representing the corresponding $D_\tau(t)$, such that $w(D_\tau(t) \in \mathbb{N}$. The different values of $w(\cdot)_i$ correspond to various disturbance scenarios, ranging from no disturbance ('0') to different types of disturbances, with positive values determined by the specifics of $D_\tau(t)$. This function classifies the similarity of all the disturbance information, $\mathscr{D}$, based on its consequential impacts on railway operations and assigns it a universal set of integer numbers to simplify subsequent computations. For example, in Fig. 2, assume at time t inside an interval $[t_1, t_2]$, station n_3 is experiencing a disturbance $D_1 = [l_3, t_1, t_2, v'] = D_{\tau 1}(t)$ and link l_3 has no disturbance $D_2 = null = D_{\tau_2}(t)$. If the reduced speed v' corresponds to value '3', then $w(D_{\tau 1}(t)) = 3$ and $w(D_{\tau_2}(t)) = 0$. Train 1 (τ_1) and train 2 (τ_2) will thus have disturbances information of n_3 ('3', with heavy rain) and l_3 ('0', with no disturbance), respectively, in their state vectors.

4.1.2. *Action and policy function*

In a certain state, a train agent shall choose one action as its dispatching decision through its policy function, $\pi(\cdot|s)$. In this article, the state's actions are characterized by a trifold decision set, i.e., a_1, a_2 and a_3 standing for holding the train (halt), dispatching the train (go) and terminate the service at current operational node, respectively. For each specific operational decision, there is an associated Q value, $\left\{q_{(s,a)} \in Q \middle| q : S \times A \rightarrow \mathbb{R}\right\}$, attached to each action which quantifies the confidence of achieving high quality train rescheduling solution with a certain action a at state s. All possible Q values $q_{(s,a)}$ in $S \times A$ are initialised by a unified value $q^{initial}$ at the beginning of the learning process by Equation (17). They are then subject to be updated during the learning process by Equation (18), i.e.,

$$q^0_{(s,a)} = q^{initial}, \forall (s,a) \in S \times A \tag{17}$$

$$q^{K'_{(s,a),e}+k}_{(s,a)} = q^{K'_{(s,a),e}+k-1}_{(s,a)} + \beta \cdot \mu^k_{(s,a)}(H^e), \tag{18}$$

$$k = 1, \ldots, K_{(s,a)}(H^e);\ e = 1, \ldots, E.$$

Equation (18) is interpreted as: the Q-value of an (s,a) pair may be updated iteratively in $K_{(s,a)}(H^e)$ steps after the completion of episode $e = 1,\ldots,E$. Within each update in step k, $q^{K^{e-1}_{(s,a)}+k}_{(s,a)}$ is the new Q value, $q^{K^{e-1}_{(s,a)}+k-1}_{(s,a)}$ is the old Q value, β is a pre-set learning rate, and $\mu^k_{(s,a)}(\cdot)$ is a metric to adjust the desirability of (s,a) at step k based on accumulated history H^e experienced by (s,a) in episode e. Note that in the update iterations (Equation (18)), not all (s,a) pairs need to be updated based H^e—only those relevant (observed) pairs will. Thus, $K_{(s,a)}(H^e)$ is defined as the total number of such observations associated with (s,a) from H^e ($K_{(s,a)}(H^0) := 0$) and $K'_{(s,a),e} = \sum_{i=1}^{e-1} K_{(s,a)}(H^i)$ is the total number of accumulated steps experienced by (s,a) until episode $e-1$. A positive value of $\mu^k_{(s,a)}(H^e)$ indicates the focused pair is more desirable while a negative value implies otherwise. The specific way in updating Q values in our own Q-learning approach will be discussed in Section 4.1.4.

Q values are stored in a Q table which is frequently updated based on the accumulated decision history. With the execution of a series of dispatching decisions on state-action pairs in episodes, the learning process of reinforcement learning algorithm will appreciate or depreciate Q values associated with every state-action pair with respect to their contributions of achieving a successful episode.

All the state-action pairs will be given a default Q value of $q^{initial} = 0$ at the beginning of our reinforcement learning algorithm. To make a decision to a state-action pair, the choice of decision for a state-action pair is driven by a designed policy. The policy will switch between an *exploration* mode and a *loyalty* mode via a threshold parameter ϵ. In the exploration mode, decision action corresponding to a state vector is chosen randomly which will provide the decision-making process with sufficient diversity to evaluate and find optimized train scheduling solutions regardless cumulated knowledge in the Q table, as shown in the first part of Equation (19). Once the system is switched into the loyalty mode, the Q table should be confident enough to make a decision using the action with the higher Q value determined by $\text{argmax}_{a \in A} q(s,a)$ as shown in the second part of Equation (19).

Once a decision is required for a train at an operational node, two Q values for two actions will be extracted from the Q table with respect to the state of the train, and result with their absolute difference. If the difference value is lower than ϵ, the action will be chosen through the exploration mode. Otherwise, the action with the higher Q value will be selected. Detailed mechanism of the action selection policy is shown in Equation (19), where rand_choose($\{a_1, a_2, a_3\}, 0.33$) means to randomly choose $a_1, a_2, a_3 \in A$ each with a 33.33% chance (exploration mode),

$$\pi(\cdot|s) = a_s = \begin{cases} \text{rand\_choose}(\{a_1, a_2, a_3\}, 0.33), \text{if } abs\left(q_{(s,a_1)} - q_{(s,a_2)}\right) < \epsilon \\ \underset{a \in A}{\text{argmax}} q(s,a), \text{otherwise} \end{cases} \tag{19}$$

Within the decision-making process, the threshold parameter ϵ plays as a 'mentor' across the whole learning process, which provides the algorithm with a benchmark on 'how much experience/knowledge will be enough to make a decision independently'. So, the higher value of ϵ means more knowledge should be accumulated in the reinforcement learning algorithm. Moreover, the design of the switching mechanism between the exploration mode and the loyalty mode differs slightly from the traditional ϵ-greedy approach. We define H_τ^e as the history of decision-made actions $a_s \in A = \{go, halt, terminiate\}$ associated with state s for train τ within episode e, organised sequentially by stations $n \in N$, as

$$H_\tau^e = \left\{\left(s_\tau^n, a_s^n\right)\right\}_{n \in N} = \left\{\left(s_\tau^1, a_s^1\right), \left(s_\tau^2, a_s^2\right), \ldots, \left(s_\tau^n, a_s^n\right), \ldots, \left(s_\tau^{|N|}, a_s^{|N|}\right)\right\} \tag{20}$$

Values collected from H_τ^e will be used to update the Q table after episode e is finished.

4.1.3. Optimization objective and tiered rewarding

Within the reinforcement learning algorithm, the train rescheduling solutions generated by the policy function interact with a virtual railway simulator to support 'real-time' train dispatching decisions during an episode of simulation. The simulator operates based on a series of state-action decisions until either all trains have reached their termination stations or a deadlock is detected. The reward for a train agent is determined by its current state-action pair (s, a), quantified as $\mathscr{R}(s, a)$, where $\mathscr{R}(\cdot)$ is the reward function. The total reward for a train τ is the cumulative reward over its trajectory H_τ^e, calculated by Equation (21).

$$R_\tau = \sum_{n=1}^{|N|} \gamma \cdot \mathscr{R}\left(s_\tau^n, a_s^n\right) \tag{21}$$

In this study, the expected cumulated reward for train τ from its state s to the end of its journey over the policy function $\pi(\cdot)$ is quantified by a state value function $V_\pi(s)$ which is shown in Equation (22)

$$V_\pi(s) = \mathbf{E}_\pi\left(\sum_{n=1}^{|N|} \gamma \cdot \mathscr{R}\left(s_\tau^n, a_s^n\right) \middle| s = s_\tau^n\right) \tag{22}$$

The proposed reinforcement learning algorithm aims to discover the optimal policy $\pi^*(\cdot)$ by repeatedly attempting and adjusting its learning process in order to maximize the total reward received by each individual agent, so the optimal value function can be formulated as follows.

$$V_{\pi^*}(s) = \max_\pi \mathbf{E}_\pi\left(R_\tau \middle| s = s_\tau^n\right) \tag{23}$$

In order to determine the total reward earned by each train agent in a given episode, an objective function is created that maps the optimization target to a particular value. In the context of this study, the optimization objective is to minimize train departure delays in every station across its journey, and consequently, improve the quality of services across the whole corridor. So, total accumulated departure delays (TAD) of all simulated trains in episode e is designed as the optimization objective. In an episode that all the trains have arrived their termination stations, the departure delay of a certain train can be derived from its rescheduled departure times and departure times in its original timetable at all passed stations, so the TAD_e can be calculated through Equation (5) which is shown in Equation (24)

$$TAD_e = \sum_{n \in N} \sum_{\tau \in \mathscr{T}} \left(t_d^Q(\tau, n, e) - t_d(\tau, n)\right) \tag{24}$$

Otherwise, if an episode is terminated by a deadlock, some of the trains' services maybe interrupted by the deadlock, and thus, their rescheduled timetables are incomplete or even empty, so the TAD_e of an episode terminated by deadlock is given as positive infinity, i.e., $TAD_e = +\infty$.

At the end of an episode, if the TAD of this episode is smaller than the currently best (smallest) total accumulated delay (denoted by $TAD' = \min_{i=1,\ldots,e} TAD_i$), this episode is a *successful* episode. Otherwise, the episode is *unsuccessful* (either by deadlock or by poor episode TAD). Within every episode e, the TAD for each individual train is also calculated, i.e.,

$$TAD_{\tau,e} = \sum_{n \in N} \left(t_d^Q(\tau, n, e) - t_d(\tau, n)\right), \text{if train } \tau \text{ is not early teminted} \tag{25a}$$

Equation (25a) only considers situations where an individual train is not early terminated. However, calculating the TAD using Equation (25a) for a train terminated by action a_3 poses a challenge because it considers a limited number of stations, potentially resulting in a small value for $TAD_{\tau,e}$. This value may not be comparable to a fully serviced train, possibly leading the algorithm to draw misleading conclusions such as the knowledge that 'terminating a train early is good'. To enhance the evaluation of the performance of early terminated trains, dummy delay values are employed to fill the undelivered stops of the train service. Adhering to British operational principles, a train is typically cancelled if its delay is projected to exceed 25 min. In this paper, we adopt this threshold value to fill the skipped stations. As a result, the $TAD_{\tau,e}$ for early terminated train is developed in Equation (25b):

$$TAD_{\tau,e} = \sum_{n' \in N} \left(t_d^Q(\tau, n', e) - t_d(\tau, n') \right) + (|N| - n') \times 25, \quad \text{if train } \tau \text{ is early terminated} \tag{25b}$$

where n' is the index of stop delivered by the train τ before it is terminated.

In a successful episode, the train-specific TADs for all trains are calculated by (25). In an unsuccessful episode.

(1) If a train has arrived its destination (regardless of deadlock), calculate its $TAD_{\tau,e}$ by Equation (25a);
(2) If a deadlock causes the failure, for a train that never arrives at its destination,
    a. If a train is directly responsible for the deadlock, set $TAD_{\tau,e} = -1$.
    b. If the train is not directly responsible, set $TAD_{\tau,e} = +\infty$.
(3) If a train is terminated by a_3, its $TAD_{\tau,e}$ is highly likely to be small, given the reduced stations considered, so the value of $TAD_{\tau,e}$ is indicative of the necessity of termination in the considered state.
    a. If there is no impending hazard within the considered train's service time frame, or if the time frame of the hazard does not overlap with the train's timetable in the specific area, the termination operation is deemed unnecessary. In such cases, set $TAD_{\tau,e} = -1$.
    b. Otherwise, the terminating operation is identified as necessary, in that case, the TAD value is derived by Equation (25b).

The best TAD values (the smallest positive value) for both individual trains and an episode across all past episodes are recorded as TAD'_τ and TAD', respectively, which will be updated at the end of each episode e. i.e.,

$$TAD'_\tau = \begin{cases} TAD_{\tau,e}, \text{if } 0 < TAD_{\tau,e} < TAD'_\tau \\ TAD'_\tau, \text{otherwise} \end{cases} \tag{26}$$

$$TAD' = \begin{cases} TAD_e, \text{if } 0 < TAD_e < TAD' \\ TAD', \text{otherwise} \end{cases} \tag{27}$$

For an individual train τ, if $0 < TAD_{\tau,e} < TAD'_\tau$, the service of the train is called *better* than the incumbent; if $TAD_{\tau,e} > TAD'_\tau$, the service of the train is called *worse* than the incumbent, and if $TAD_{\tau,e} = -1$, which can only happen with a deadlock, the train's service is called *responsible* (for the deadlock) or the train is terminated without necessity.

Because the optimization objective of this study, TAD, is a globalized evaluation benchmark which can only be fully calculated at the end of simulation, it is hard to quantify the performance of a single decision through the intermediate states of the simulation. The reward value shall be decided at the end of each episode e of the simulation to qualify a completed episode and to be applied in Q learning algorithm. In this study, the overall reward given to train τ after episode e, denoted by $R_{\tau,e}$, is consisted with two elements: (i) $r_{\tau,e}$, the reward to evaluate an individual train τ's service, and (ii) r_e, the reward to quantify the entire episode e.

$$R_{\tau,e} = r_{\tau,e} + r_e \tag{28}$$

The reward for episode, r_e, is decided by the final performance of the considered episode.

$$r_e = \begin{cases} 0, \text{if } TAD_e > TAD' \\ +1, \text{otherwise} \end{cases} \tag{29}$$

Reward r_τ is decided based on the completeness and effectiveness of the service of the train.

Table 2
Tiered reward values under different episode conditions.

Condition of episode		Value of reward		
Train's performance	Episode result	$r_{\tau,e}$	r_e	$R_{\tau,e}$
Better	Unsuccessful	+1	0	+1
Worse	Unsuccessful	0	0	0
Responsible	Unsuccessful	−1	0	−1
Better	Successful	+1	+1	+2
Worse	Successful	0	+1	+1

$$r_{\tau,e} = \begin{cases} 0, \text{if } TAD_{\tau,e} > TAD_{\tau}^{'} \\ +1, \text{if } 0 < TAD_{\tau,e} \leq TAD_{\tau}^{'} \\ -1, \text{if } TAD_{\tau,e} = -1 \end{cases} \tag{30}$$

The detailed rewarding mechanism is listed in Table 2.

4.2. Advantages of tiered rewarding

In this paper, the principle under the *tiered reward/punish mechanism* aims to classify contributions of each train within an episode and maps the optimization objective into tiered objectives, i.e., minimizing the TAD value of the whole episode and minimizing the TAD of each train. In a successful episode, all the decisions will get a positive reward and Q value attached with these decisions will be appreciated (positively rewarding). In unsuccessful episodes which are interrupted by a deadlock, the episode consists with three types of train services, i.e., trains that successful delivered their services, trains whose services were interrupted by the deadlock and trains that are directly responsible for the deadlock. In such a condition, the trains leading to deadlock are the 'culprits', so the last decisions of the trains that are directive responsible to the deadlock and should be punished (negatively rewarding). The trains that are irresponsible for the deadlock are relatively 'innocent' in an unsuccessful episode, so every decision made by these trains is not considered (nullified rewarding). The tiered reward/punish mechanism helps the Q learning algorithm breakdown the decision-making process to every state-action pair with its own contribution to the whole episode and filter the 'essence' and 'dross' of past experience, through which the learning process can be thus more computational efficient.

4.2.1. Q learning process

At the end of each episode, all the Q values of past decisions will be updated using Q learning algorithm through the decision history of each train. The Q learning algorithm is based on the Bellman equation (Equations (17) and (18)) with an initial value of zero assigned to all Q values at the beginning of the learning process. Then in the end of each episode it conducts the recursion (18) over all state-decision pairs a subset of history $H_{\tau}{'}$ (see later) of train τ, and then over all trains $\tau \in \mathscr{T}$. The metric $\mu^{k}_{(s,a)}(\cdot)$ in Equation (18) is defined following the principle of backward propagation as in a typical Q-learning process, i.e.,

$$q^{new}_{(s^n_\tau,a)} = q^{old}_{(s^n_\tau,a)} + \beta \cdot \left(R_{\tau,e} + \gamma \cdot \max_{a \in A} q^{old}_{(s^{n+1}_\tau,a)} - q^{old}_{(s^n_\tau,a)} \right),$$

$$\forall (s^n_\tau, a) \in \overline{H}^e_\tau, \forall \tau \in \mathscr{T}, e = 1, 2, \ldots, E \tag{31}$$

where β and γ are factors between (0,1] and stands for the learning rate and discount factor, respectively. s^n_τ is the state experienced by train τ at station n; s^{n+1}_τ stands for the next observed state of τ after the execution of decision at state s^n_τ. $\overline{H}^e_\tau$ is a subset of the current H^e_τ for train τ based on its reward value in episode e, $R_{\tau,e}$, i.e.,

$$\overline{H}^e_\tau = \begin{cases} \{(s^1_\tau, d_s), (s^2_\tau, d_s), \ldots, (s^{|N|}_\tau, d_s)\}, R_{\tau,e} > 0 \\ \{(s^{|N|}_\tau, d_s)\}, R_{\tau,e} = -1 \\ \varnothing, R_{\tau,e} = 0 \end{cases} \tag{32}$$

Note that after episode e, looping over state-action pairs is realised by first looking up the elements in the history of an individual train τ, i.e. $\overline{H}^e_\tau$, from the first station to the last (if $R_{\tau,e} > 0$), or only at the last (if $R_{\tau,e} = -1$), or doing nothing (if $R_{\tau,e} = 0$). If the current station n matches with a pair (s^n_τ, a), an update between an 'old' and 'new' Q-values of (s^n_τ, a) will take place as per (31). This recursion will then continue as the process loops over all the trains $\tau \in \mathscr{T}$ each time a matching between station n and pair (s^n_τ, a) happens. This two-layer looping is illustrated in Algorithm 1. Using backward propagation, the Q learning algorithm will recursively update all eligible Q values of state-action pairs that are found in $\overline{H}^e_\tau$. If neighbouring same states in $\overline{H}^e_\tau$ are detected, only the Q value of the last state-action pair will be updated. For the Q value associated with $s^{|N|}_\tau$, there will be no subsequently observed state, so the part $\gamma \cdot \max_{a \in A} q(s^{n+1}_\tau, a) - q(s^n_\tau, a)$ in (31) will be assigned to 0. The detailed Q learning algorithm is shown in Algorithm 1:

Algorithm 1. Pseudo code of Q-learning algorithm (Execute at the end of episode e)

```
1  Decide the overall reward value, $R_\tau$, for each train (see Eq. (28))
2  for train $\tau \in \mathscr{T}$
3    if $R_{\tau,e} > 0$
4      $\overline{H}^e_\tau := \{(s^1_\tau, a_s), (s^2_\tau, a_s), \ldots, (s^n_\tau, a_s), \ldots, (s^{|N|}_\tau, a_s)\}$
5    elseif $R_{\tau,e} < 0$
6    $\overline{H}^{2e}_\tau := \{(s^{|N|}_\tau, d_s)\}$
7    else
```

(continued on next page)

(*continued*)

```
8        $\overline{H}_{\tau}^{e} := \varnothing$
9      end
10     for state s from $s_{\tau}^{|N|}$ to $s_{\tau}^{1}$
11       if $s == s_{\tau}^{|N|}$
12       $q_{(s_{\tau}^{n},a)}^{new} \leftarrow q_{(s_{\tau}^{n},a)} + \beta \cdot R_{\tau,e}$
13     elseif $s \neq s_{\tau}^{|N|}$ & $s == s_{\tau}^{n+1}$
14       $q_{(s_{\tau}^{n},a)}^{new} \leftarrow q_{(s_{\tau}^{n},a)}$
15     else
16     $q_{(s_{\tau}^{n},a)}^{new} \leftarrow q_{(s_{\tau}^{n},a)}^{old} + \beta \cdot \left(R_{\tau,e} + \gamma \cdot \max_{a \in A} q_{(s_{\tau}^{n+1},a)}^{old} - q_{(s_{\tau}^{n},a)}^{old}\right)$
17       end
18     end
19   end
```

4.2.2. *Significance of the proposed algorithm*

A similar reinforcement learning algorithm as proposed in Khadilkar (2019), labelled as K-QL, is set as a benchmark to be compared with the newly proposed reinforcement learning algorithm in this paper (marked as New-QL) in section 5. It is worth noting that K-QL lacks the capability to make early termination decisions within its algorithm. The key differences between the two algorithms are indicated in Table 3.

The key rationale of solely using directions (up and down) to distinguish trains in state vectors is to take the advantage of train homogeneity such that accumulated knowledge from early dispatched trains of the same type can be reused by later trains more quickly and more effectively. In this sense, using train ID will actually slow down the learning process as the knowledge learnt from the same type of trains cannot be shared with other homogeneous trains. In many cases, trains (of the same priority and direction) can be regarded homogeneous and thus their decision knowledge should be shared and transferred to other instances with similar structures. Our direction-based labelling can be easily extended to the cases with train priorities by adding another attribute 'priority' such that individual IDs are still not needed.

Apart from that, using train direction to label them significantly reduces the dimension of state space. The state vector is formed as $s_{\tau}(t) = \left(\boldsymbol{\alpha}_{\tau}(t), w\left(D_{\tau}(t), C_{\tau}^{id}\right)\right)$. The maximum size of state space (or equivalently, of the Q table) is $|S| = dim[\boldsymbol{\alpha}] \cdot dim[w(\mathscr{D})] \cdot dim[\mathscr{C}_{id}]$, where we use $dim[\cdot]$ to denote the dimension needed for each component of the state vector and $\mathscr{C}_{id}$ stands for the train identifier (ID or direction). The disturbance entry only takes one space ($dim[w(\mathscr{D})] = 1$) in our case. About spaces for the train identifier, by using train ID, we have $dim[\mathscr{C}_{label}] = |\mathscr{T}|$, and if direction is used, it is reduced to $dim[\mathscr{C}_{id}] = 2$. In a practical railway network normally $|\mathscr{T}| \gg 2$, and thus, representing train characteristics with train service direction is computationally efficient and will reduce the size of state space and the Q table by a factor of $|\mathscr{T}|/2$, whose benefits will be significantly enlarged with the increase of train volume.

Using lightweight train identifier by direction also encourages reusability where the Q value learnt from one instance can be used for another instance with similar structure or with a different set of trains. Indeed, by restricting learning only within an individual train, knowledge learnt in one train cannot be transferred to another train in another case. If trains are only labelled by their directions, knowledge (Q values) learnt from one instance can be propagated in trains of the same direction in another smoothly.

In our simulation approach, unlike the approach in Khadilkar (2019) which has deadlock avoidance mechanism, we intentionally allow the algorithm to experience deadlocks. This strategy helps agents learn from incorrect or inappropriate actions, improving their ability to estimate the value of such actions and reducing the likelihood of making similar mistakes in the future (Sutton and Barto, 2018). Experiencing deadlocks is beneficial for training agents to handle these situations and is crucial for enabling them to avoid deadlock decisions in future episodes.

4.3. *Simulation approach and algorithm integration*

A time-based simulator is used in this study to represent train operations in a single-track environment $\mathscr{E} = (N, L, \mathscr{D}, \mathscr{C})$. A full run of one day's train operations is called an episode, operating in discrete time intervals of 1 min, the minimum time resolution of the British railway timetable. During each episode, trains are dispatched from their origin at the scheduled departure time, unless their

Table 3
Key differences between K-QL and New-QL.

Properties of algorithm	K-QL	New-QL
Representation of train characteristic	Individual train ID	Train direction (up and down)
Rewarding mechanism	Rewarding based on performance of episode (success rate)	Tiered rewarding based on performance of both individual train services and an entire episode
Punish mechanism	Punish all the state-action pairs in a failed episode	Punish only the relevant state-action pairs causing deadlock in a failed episode

front link is occupied, and are returned to inventory upon reaching their destination. Time and location are recorded for each train. The train agent updates track availability information (i.e., $\alpha_k, k \in N \cup L$) at each time step and makes decisions as needed. When running on a link, trains continue until the next station. If a train's dwell time exceeds its schedule, a decision to go or wait is made by the agent using Q-learning. Deadlocks are detected by negative capacity values and can occur at multiple locations simultaneously. The simulation ends when all trains reach their destinations or a deadlock is detected. Disturbance information $D_\tau(t)$ for a specific train τ at time t is determined based on time t, the train's current location (from route in characteristic C_τ), and its departure decision a_s.

The reinforcement learning algorithm will interact with the simulator through every episode information of train dispatching decisions and accumulate knowledge from all past episodes will be passed from one to the other. The learning process will be terminated once the algorithm cannot improve the TAD value consecutively in 200 episodes. Here, the termination approach draws inspiration from heuristic algorithms to enhance the scalability of the algorithm. The interaction flow between the simulation and reinforcement learning algorithm is shown in Fig. 3.

5. Case study

Cambrian Line, a British single-track rail corridor, is used to test the proposed reinforcement learning algorithm. The corridor spans 81.5 miles from Aberystwyth station to Shrewsbury station and includes eight intermediate stations. Each pair of adjacent stations is linked by a single-track line that allows only one train to operate at a time. All stations have two platforms, enabling two trains to dwell simultaneously. A schematic map of the Cambrian Line is provided in Fig. 4. In this study, three different service volumes were used to evaluate the performance of the proposed Q-learning algorithm (New-QL).

- **Low Traffic Volume**: 11 trains over a period between 12:30 and 21:30
- **Medium Traffic Volume**: 21 trains over the same period
- **High Traffic Volume**: 30 trains over the same period

The algorithm is implemented in Matlab R2020 and scenarios tested on a Windows PC with AMD Ryzen 7 4800U processor and 24 GB RAM. Detailed benchmarks for evaluating the algorithm's performance are discussed in Appendix A, which also covers the parameter settings and the rationale behind selecting these parameters.

5.1. Performance in solving train rescheduling problem

The capability on tackling the VSTR problem of the proposed algorithm is tested in synthetic scenarios considering several realistic disturbance possibilities in railway operations. These disturbances include weather impacts and planned engineering activities, leading to speed reductions and extra dwell times. Four types of scenarios are considered.

- **Scenario 1**: full-day weather impact to a single section of the corridor – speed reduction spans from 10% to 40%, applicable to all the trains passing the section. This means there is a single disturbance D in a link l throughout the day, or $D = [l, day\_start, day\_end, v']$, where $v' = 90\%, 80\%, 70\%, 60\%$. In practice, rail operations during bad weather will follow either the First Come First Served (FCFS) rule, the Timetable Order Enforced (TTOE) rule, or both.

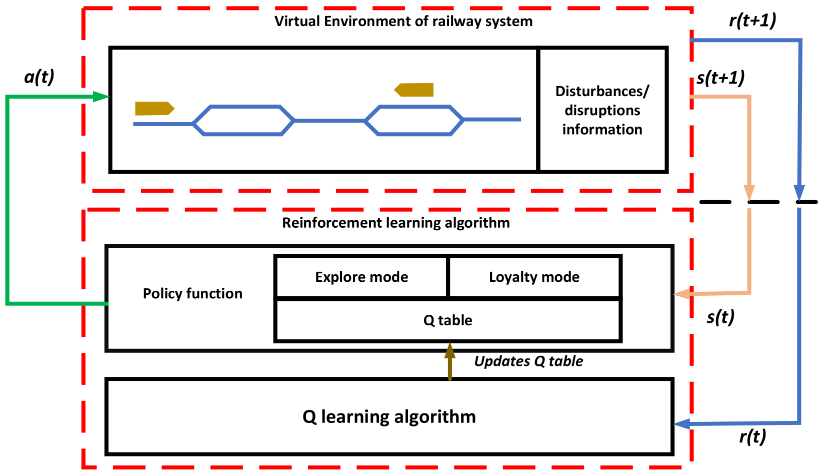

Virtual Environment of railway system
Disturbances/ disruptions information
r(t+1)
s(t+1)
a(t)
Reinforcement learning algorithm
Policy function
Explore mode
Loyalty mode
Q table
s(t)
Updates Q table
Q learning algorithm
r(t)

Fig. 3. The interaction flow between simulation and reinforcement learning algorithm.

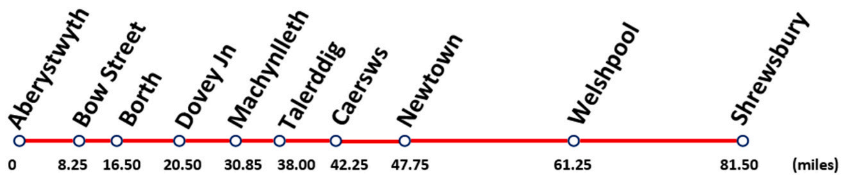


Fig. 4. Schematic map of Cambrian Line single track corridor (with mileage from Aberystwyth).

- **Scenario 2**: rolling weather impact moving in both temporally and spatially, creating a sequence of k locally affected incidents – speed reduction of 34% to all the trains passing the considered rail section in specific time period. This means $D = [D_1, D_2, \ldots, D_k]$, where $D_i = [l_i, t_{i1}, t_{i2}, 66\%]$. In practice, FCFS and/or TTOE will be used.
- **Scenario 3**: engineering activity in a specific station n – extra dwell time Δt (spans from 5 min to 30 min) is applied at a certain timestamp t_0. This means $D = [n, t_0, \Delta t]$. In practical rail operation, we usually prioritize station throughput, so FCFS is the default operation rule.
- **Scenario 4**: pre-planed engineering activity in a specific section - temporary closure of section l in both directions during a specified time period, $D = [l, day\_start, day\_end, 0]$. Engineering activities usually extend over a long period in practice, during which operators may consider cancelling their services if necessary.

Samples of rescheduled timetable of Scenario 1, 2, 3 and 4 using New-QL are shown in Fig. B-1, B-2, B-3 and B-4, respectively in Appendix B. The dotted lines correspond to the original timetabled trains while the solid lines give the rescheduled trains. Comparative

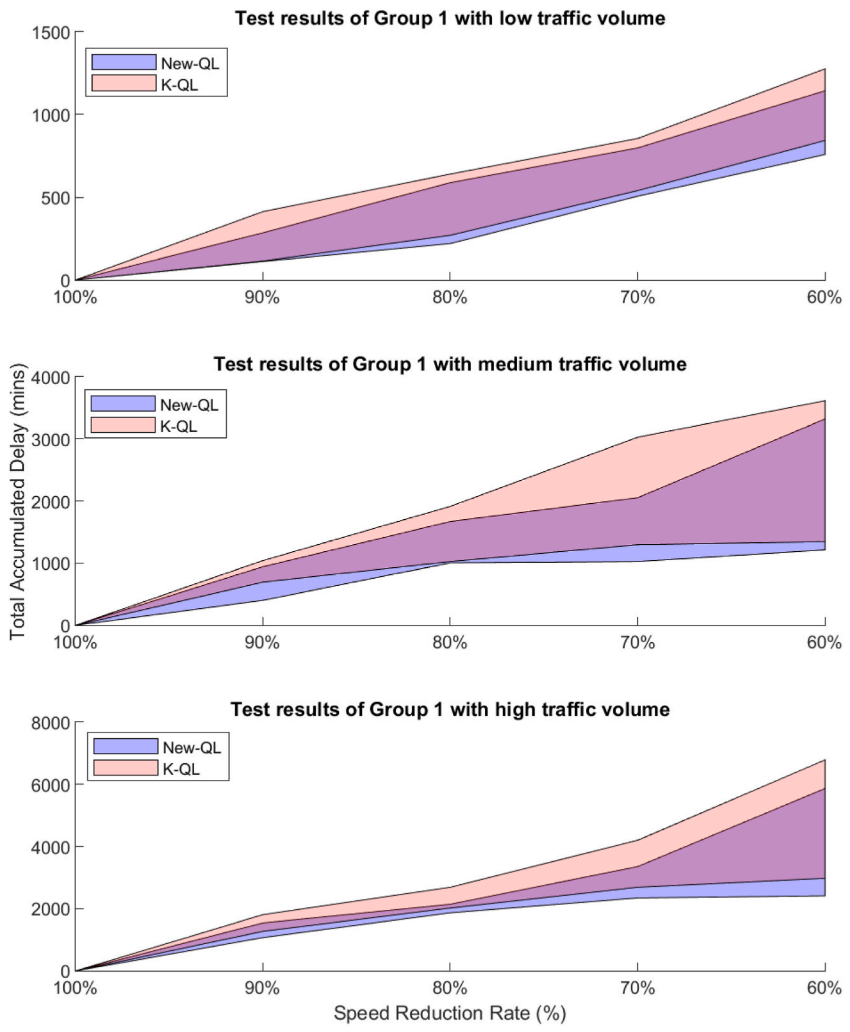


Fig. 5. Test results of Scenario 1.

results of the tested scenarios between K-QL and New-QL are presented in Fig. 5, Table 4 and Fig. 6, respectively. In Fig. 5, the ribbon areas represent the range of objective function values obtained in the designed test case trials in Scenario 1. The ribbon is designed to visualize all potential delay values in Scenario 1. Each ribbon, whether for New-QL or K-QL, is bounded by the maximum and minimum delay values under the designed speed deduction rate, providing a comprehensive comparison between the performance of the New-QL and K-QL algorithms.

New-QL performs better than K-QL in all the cases tested in the Scenario 1 to 3. In Scenario1, New-QL can reduce TAD spans from 3 min to 920 min with a mean value of 279.8 min; in Scenario 2, the improvement on TAD spans from 17 min to 495 min with a mean value of 174.2 min; and in group 3, the improvement of New-QL is from 5 min to 276 min with a mean value of 125.8 min. Additionally, in Fig. B-4 for Scenario 4, due to the prolonged blockage of the section between Newtown and Welshpool, New-QL opts for the early termination of 4 trains rather than rescheduling them. This strategic sacrifice of terminated services significantly enhances the efficiency and convenience of subsequent rescheduling decisions. Across all the scenarios, in average, the New-QL can reduce the delay time by 206.8 min compared to K-QL, thanks to the tiered rewarding mechanism of New-QL that considers performance both at the episode and individual train levels, the lightweight train characteristic labelling that enables knowledge share among different trains of the same directions. Finally, the representation approach of terminal station by a big number M also improves the quality of final solution generated by New-QL. Because the train-oriented agent modelling approach will only extract state vector from the surrounding environment and in K-QL the occupancy information of terminal station is identified by platform available capacity, K-QL will be 'myopic' to the trains which have not accessed the simulation yet, i.e., the dispatching decisions made in K-QL will ignore a train that has not been dispatched at initial station and launch a train that is heading to its terminal station (same station as initial station of the train not been dispatched) if its next single-track element links terminal station is empty. However, such a decision might block a train that cannot depart from its initial station due to its next single-track line is occupied by other train and results in huge delay across its service. By using a virtual big number for the initial/terminal stations, New-QL is able to avoid such train conflicts as it distinguishes the initial and terminal stations from the intermediate ones.

5.2. Computational efficiency comparison of reinforcement learning and exact algorithm

In this section, we mainly discuss the computational time of the proposed reinforcement learning algorithm. Based on the findings presented in this section, it can be inferred that the proposed reinforcement learning algorithm exhibits better computational efficiency as the complexity of the problem grows, attributed to an augmented volume of trains. Moreover, the algorithm demonstrates a notable capability to yield solutions of equivalent quality, experiencing little to no degradation when compared to exact algorithms.

Although the running efficiency may be different due to variable CPU rates of the computational unit, the computational time is given here to benchmark typical time consumption of resolving a train rescheduling problem in the single track rail corridor in this study. To evaluate the whole computational process of the proposed reinforcement learning algorithm including its three typical phases, the criteria in Appendix A, the episode ID of E1, E2 and E3, are also used in this section. The same train rescheduling problem was also tested on the same windows PC using exact algorithm. The same train rescheduling problems listed above is solved by Gurobi (v9.2.6) interfaced with Matlab 2020b via Yalmip (R20210331). The computational times for New-QL in different phases are summarized in Table 5, and the comparation of computational time between New-QL and exact algorithm is shown in Fig. 7.

In the 11-train and 21-trains cases, the exact algorithm has faster computational time. However, the reinforcement learning approach significantly outperforms the exact method in the 30-trains case, where the exact algorithm requires longer time. This indicates that the proposed reinforcement learning algorithm is more efficient in handling high-volume scenarios and offers better scalability compared to exact methods in this train rescheduling problem.

The reason for this 'reversed result' on the 30-train case is that the computational complexity of reinforcement learning and an exact algorithm increases in heterogeneous manners. In New-QL, computational complexity depends on the number of states of all the trains shall face in a single episode, making the computational complexity is proportional to the state space of the entire train rescheduling problem. Thus, the computational complexity increases linearly with the increase of train volume. Conversely, within exact algorithms, being NP-hard, have complexity dependent on the number of variables and constraints, growing exponentially with problem size. Therefore, reinforcement learning is more efficient and scalable in practice for this problem.

5.3. Knowledge reusability: learning based on prior knowledge

Reusability of prior knowledge is another feature of the proposed reinforcement learning algorithm, i.e., the Q table obtained from previously solved train rescheduling problems (basic knowledge) can be reused and further enhanced when solving a new instance with shorter computation time and competitive solution quality compared to solving the new instance from scratch. This is because the

Table 4
Test results of Scenario 2.

Affected area and time	TAD value of K-QL			TAD value of New-QL		
	11 trains	21 trains	30 trains	11 trains	21 trains	30 trains
Dovey Jn to Machynlleth, 7.30–9.30	87	250	723	55	123	597
Dovey Jn to Talerddig, 7.30–11.30	164	451	1524	147	385	1029
Dovey Jn to Caersws, 7.30–13.30	326	536	3158	174	432	2728

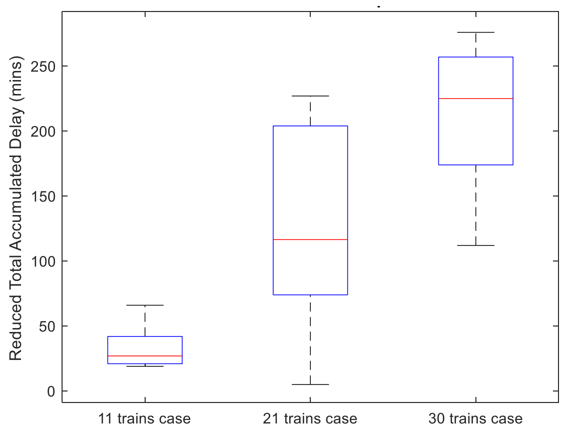


Fig. 6. Improvements (reduced TAD) of New-QL against K-QL (scenario 3).

Table 5
Computational time of proposed reinforcement learning algorithm.

Test case	E1 (end of Phase 1)	E2 (end of Phase 2)	E3 (end of Phase 3)
11 trains case	20.4553s	22.9889s	61.5521s
21 trains case	37.6497s	43.2274s	115.0406s
30 trains case	68.6602s	76.0839s	205.8008s

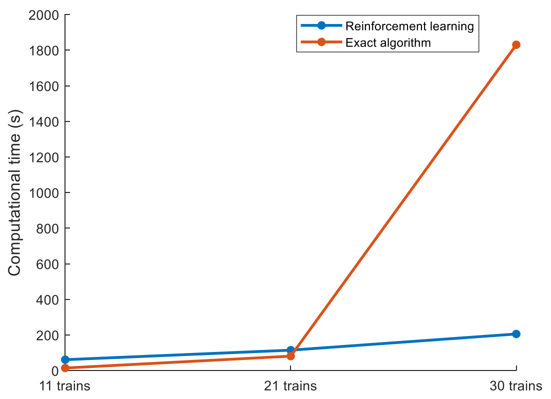


Fig. 7. Computational time of reinforcement learning and exact algorithm.

direction-based train labelling allows the reuse of the Q values learnt previously over all trains of the same direction in a new instance, rather than only on an individual train as in the case of K-QL.

Two disturbance scenarios, *R1* and *R2*, are considered in the tests, identifying different speed reduction rates. The reusability tests estimated the impact by comparing the final output solution and convergence KPIs (defined in Table A-1) of the proposed algorithm

Table 6
Results of experiments on knowledge reusability.

Scenario	Number of trains	Disturbance	Prior knowledge	E1	E2	E3
KT_1	21	R1	None	65	78	289
KT_2	21	R1	11 trains + R1	22	39	241
KT_3	30	R1	None	418	495	705
KT_4	30	R1	21 trains + R1	47	135	311
KT_5	30	R1 & R2	None	545	798	1051
KT_6	30	R1 & R2	30 trains + R1	81	95	296

with/without prior knowledge for solving the same problem. Table 6 shows the results of reusability tests in 6 scenarios with different number of trains, disturbance (R1: medium weather hazard between Bow Street and Borth, with 10% speed reduction, R2: severe weather hazard between Bow Street and Borth, with 20% speed reduction), if prior knowledge is used (if yes, based on what previously learnt Q table).

The results indicate that using knowledge from prior instances significantly shortens the learning process, particularly in the Phase 1, where the total number of episodes for accumulating knowledge to avoid deadlocks is reduced. This is because the solutions to these scenarios are inherently similar. For example, in scenario KT_1 to KT_4, the Q table gained from the instances with a lower train volume (21 trains) is used to solve the instance with a higher volume (30 trains) under the same disturbance R1. The prior Q table cannot fully present all state conditions when the traffic volume has increased. So, the episodes in Phase 1 of KT_2 and KT_4 are used to populate extra state-action pairs to cover the new states from extra traffic volume, saving significant time compared to learning the Q table from scratch. Similarly, in scenario KT_5 and KT_6, the prior Q table only contains knowledge under disturbance *R1*. However, as the related disturbances will only impact train operations in one section rather than the whole corridor, the prior Q table can still be reused to make decisions for trains not affected by *R2,* avoiding exploring all essential state-action pairs, and thus reducing the time and computational effort of the learning process.

Furthermore, it is important to note that for the proposed reinforcement learning to be applicable in terms of knowledge reusability, the infrastructure and/or the service volume should be extended in a homogenous manner. This expansion should not alter the underlying logic of created state-action pairs within the reinforcement learning algorithm, and thus, the existing knowledge can be reapplied to newly established scenarios. For example, let the single-track corridor extend with more stops and covers long distance, since the physical extension of infrastructure will not change the representation of the state vector in reinforcement learning (i.e., the length of track is not included in state vector), there is no alteration in the logical connection between the state and action results from the extension, moreover, the operational rules remain the same with the single track corridor, and thus, the knowledge populated in earlier cases can be fully reused without much change.

Finally, the knowledge reusability feature of reinforcement learning is highly significant and is believed to have extensive applications throughout the development lifecycle of data-driven algorithms. Here, we outline the conditions under which this knowledge reusability property is particularly suitable.

1) **Same and/or similar infrastructure topology**: When dealing with rail networks that have the same or similar infrastructure topology, such as another corridor single-track line with a few additional or fewer nodes and links, the reinforcement learning approach can effectively handle problems with structural similarities. This allows the knowledge base to be reusable, as the learned policies and strategies can be applied to rail infrastructure with similar topological features.
2) **Same and/or similar operational rules**: When the rail networks operate under the same or similar operational rules, such as headway regulations or segment occupation constraints, reinforcement learning can leverage the pre-learned knowledge. This enables the reusability of the trained models, as they can adapt to operational environments that follow similar rules and constraints.
3) **Largely homogenous train properties**: The knowledge base is suitable for reuse when the properties of the trains are largely homogenous, such as: same service type (i.e., local, regional and high speed services), similar services under different operators (i.e., competitive services over same infrastructure), etc.

6. Conclusions

This research presents a novel reinforcement learning-based approach for VSTR in a single-track corridor. Q-learning is used due to its suitability for our VSTR problem, which features simple state and action representations and prioritizes efficiency and interpretability. Our proposed algorithm offers several advantages over existing reinforcement learning approaches.

1) Trains are distinguished only by their direction (up and down) in the state vector, enabling knowledge sharing of Q values among all trains in the same direction. This significantly reduces the search space and accelerates the algorithm.
2) A tiered reward/punishment mechanism makes the search process more purposeful and enables more meaningful backward propagation based on episode history, improving the quality of the final solution.
3) The knowledge gained from prior instances is reusable in new instances, as the Q-learning algorithm can identify similarities and differences, focusing computational resources on the differences.
4) The knowledge reusability feature is further explored to identify potential areas in railway industry where it can shorten the algorithm development lifecycle through effective reuse of previously acquired knowledge.

Experiments on the Cambria Line, a single-track corridor in the UK, used real-world and artificial timetables of varying sizes from 11 to 30 trains. We identified three phases during the learning process and defined three KPIs to measure convergence performance. The proposed algorithm outperformed an existing algorithm in all KPIs, with an average total delay reduction of 206.8 min compared to the benchmark reinforcement learning algorithm. Computational experiment shows that both the tiered rewarding mechanism and direction-based train characteristic labelling contributed to the improved performance, with the latter also enabling efficient and reliable knowledge reusability in new instances with higher traffic volumes or different disturbance scenarios.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported by the Assisted Very Short Term Planning (aVSTP) Project (Grant No.: RSSB/494204565) funded by UK Rail Safety and Standards Board (RSSB). The first author also acknowledges the support of the Michael Beverley Innovation Fellowship at the University of Leeds.

Appendix A. KPIs for algorithm testing and parameter tuning

In this section, we mainly discuss the impact of the parameters in the proposed reinforcement learning algorithm including the train characteristic property C_{τ}^{id} in the state vector and the learning rate β. A common approach to evaluate the searching capability and convergence rate of an algorithm is analyzing its convergence curve of the entire solution searching process. A typical convergence curve of the proposed reinforcement learning algorithm is shown in Fig. A-1. The whole learning process can be divided into three phases.

- **Phase 1**: At the early stage of the algorithm, almost all episodes will fail, and the Q table will first learn rules to avoid deadlock in the corridor until it is capable of making a series of correct decisions to achieve a first feasible episode. Most of the decisions in Phase 1 are made randomly in the exploration mode (see Equation (19)). The criteria to declare the end of Phase 1 is that the Q table is able to make a feasible solution.
- **Phase 2**: The Q table will accumulate knowledge on making a series of decisions to optimize dispatching solution and try to minimize the TAD value in each episode of Phase 2. The decisions will be chosen in both exploration mode and loyalty mode, and the TAD of every episode of Phase 2 fluctuates frequently, until it converges to the best solution. The criteria of end of Phase 2 is that the final optimized solution is detected, which is defined as the schedule found in the episode with the smallest TAD throughout the entire learning process
- **Phase 3**: The algorithm has obtained high-quality Q values for decision-making and finally converges to optimized solutions in a stable state. With the optimized solution being continuous detected across episodes, the decisions in an episode will be fully made via the loyalty mode of policy function (see Equation (19)) and start making the same or similar dispatching solutions in later episodes. The algorithm will thus be terminated when no improvement has been made for 200 episodes.

In this study, efficiency of the reinforcement learning algorithm means the algorithm can fast-converge to its final optimized solution. Because the computational time of an algorithm may be different due to factors such as CPU rate, RAM capacity, etc., the associated episode ID where the algorithm terminates (denoted as E3) is chosen as the criterion to quantify algorithm efficiency. Additionally, to better evaluate its searching capability in different phases of the learning process, the episode IDs at the end of Phase 1 (marked as E1) and Phase 2 (marked as E2) are also considered. Furthermore, because the decisions are chosen randomly in the exploration mode, randomness of the exploration mode of the action selection policy may significantly influent the quality of the optimized solution. As a result, reproducibility tests of the algorithm are applied to benchmark its searching capability in Phase 1 and Phase 2 where the percentage of episodes with the same optimized solution is recorded. These KPIs are listed in Table A-1.

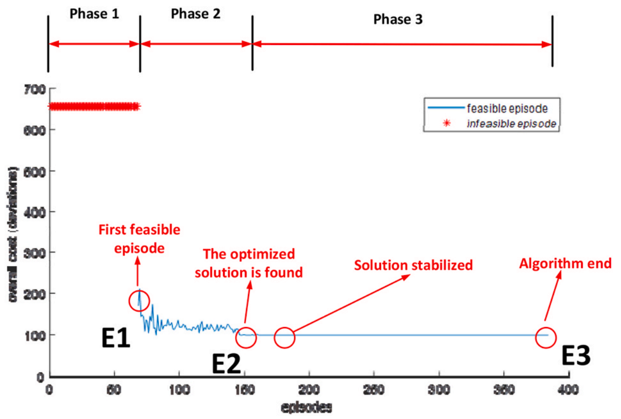


Fig. A-1. Typical convergence curve of reinforcement learning algorithm

Table A-1
KPIs to evaluate Q learning algorithm

KPIs	Description	Expected value
E1	Episode ID where the first feasible solution is found	As small as possible
E2	Episode ID where the final optimized solution is first found	As small as possible
E3	Episode ID that the algorithm ends	As small as possible
Reproducibility (%)	The percentage of episodes where the algorithm can converge to the same optimized solution	As high as possible, not lower than 95%

The tests are recurrently executed for 150 times using same scenario i.e., 21 trains representing medium train volume, 10% speed reduction in the section between Welshpool and Newtown. We compare two strategies in characterising trains in state vector, i.e. train ID based and train direction based. The remaining parameters were determined based on computational experiments and our empirical knowledge, i.e., $f = b = 3$, $\gamma = 0.95$, $\epsilon = 0.5$, $\beta = 0.01, 0.02, \ldots, 0.06$.

The statistical results of the two groups of tests against proposed KPIs are summarized in Table A-2. The test clusters are numbered in the format of "*train characteristic property_learning rate*", and train service direction and train ID based strategies are identified by "TD" and "ID", respectively. e.g., TD_0.01 and ID_0.03 stand for train direction based with $\beta = 0.01$ and ID based with $\beta = 0.03$, respectively.

Table A-2
Statistic results of test clusters in group 1 and 2

Test Cluster	E1 (Avg.)	E2 (Avg.)	E3 (Avg.)	Reproducibility (%)
Direction based				
TD_0.01	109.8	123.4	330.4	96.67%
TD_0.02	56.7	65.1	278.1	93.33%
TD_0.03	39.8	46.9	295.3	79.3%
TD_0.04	32.6	37.9	282.4	72.0%
TD_0.05	27.2	31.8	276.8	68.6%
TD_0.06	23.3	27.2	268.5	65.3%
ID-based				
ID_0.01	583.3	699.1	899.5	91.3%
ID_0.02	316.2	366.8	584.2	79.3%
ID_0.03	203.2	232.8	501.1	58.0%
ID_0.04	154.6	174.3	458.2	43.3%
ID_0.05	134.0	150.1	445.4	29.3%
ID_0.06	120.9	135.4	436.0	17.3%

The above results suggest that, under the same learning rate, test clusters in the train direction based group perform consistently better than the ID based group in both convergence and reproducibility. The required number of episodes across all three learning phases and in particular Phase 1 in the direction based group is much fewer, showing the significant advantages of our lightweight direction based train characteristic labelling in knowledge sharing and state space reduction.

The learning rate β controls the speed that the algorithm switches between the exploration mode and loyalty mode measured by ϵ. A higher β means faster convergence and consequently smaller numbers of episodes in Phase 1 and Phase 2. However, shortened Phase 1 and Phase 2 tend to significantly degrade the quality of optimized solution because the number of evaluated solutions is limited, that is, the algorithm cannot fully explore the searching space, and thus reduces the reproducibility of the algorithm.

Among the tests conducted above, cluster TD_0.01 has the best performance. Hereafter, we adopt the same settings (direction based and learning rate $\beta = 0.01$) in the remainder of the case study experiments.

Appendix B. Samples of results in Section 5.1

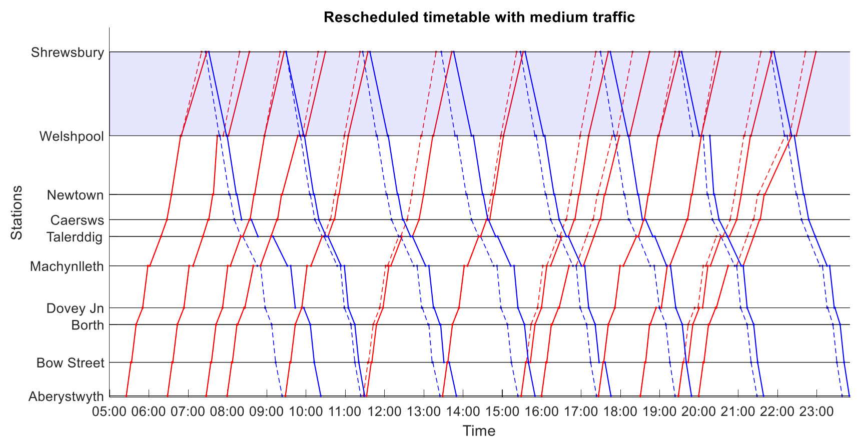


Fig. B-1. Sample of rescheduled timetable with medium train volume in Scenario1

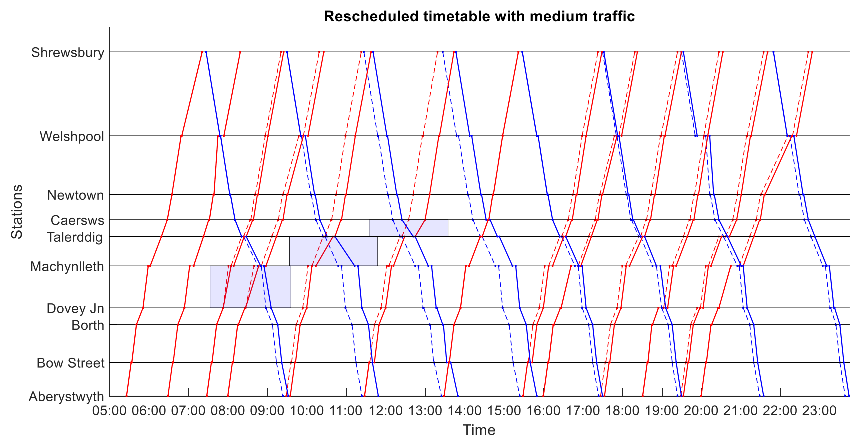


Fig. B-2. Sample of rescheduled timetable with medium train volume in Scenario 2

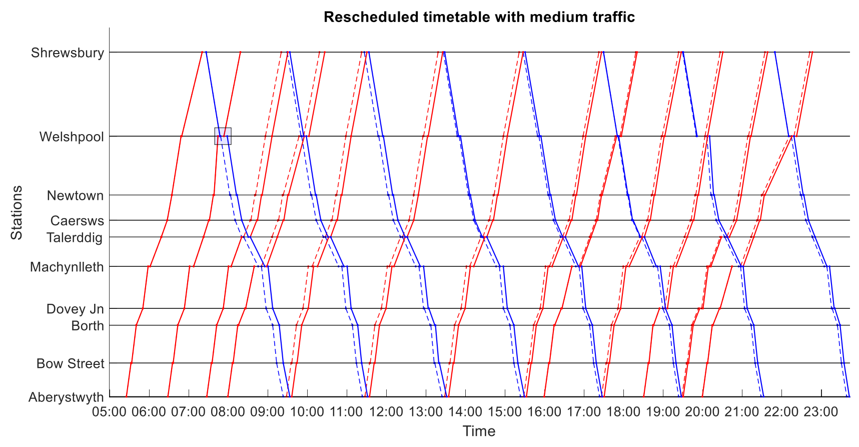


Fig. B-3. Sample of rescheduled timetable with medium train volume in Scenario 3

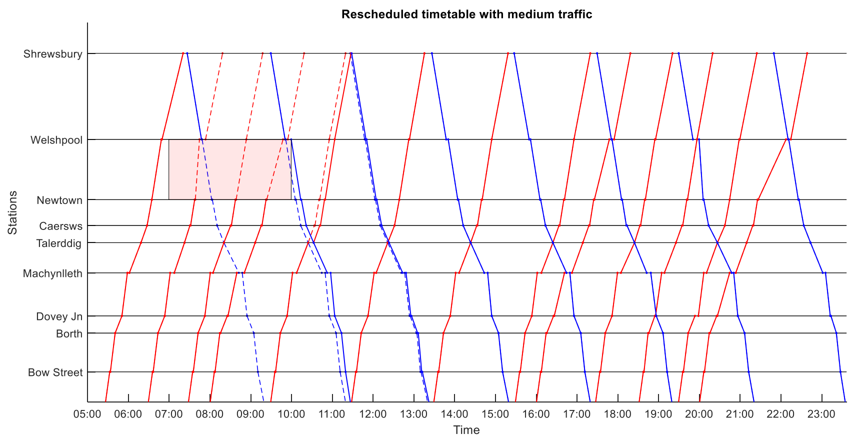


Fig. B-4. Sample of rescheduled timetable with medium train volume in Scenario 4

References

Agasucci, V., Grani, G., Lamorgese, L., 2023. Solving the train dispatching problem via deep reinforcement learning. J. Rail Transp. Plan. Manag. 26, 100394. https://doi.org/10.1016/j.jrtpm.2023.100394.

Bešinović, N., Donato, L. De, Flammini, F., Goverde, R.M.P., Lin, Z., Liu, R., Marrone, S., Nardone, R., Tang, T., Vittorini, V., 2021. Artificial intelligence in railway transport: taxonomy, regulations and applications. IEEE Trans. Intell. Transport. Syst. 1–14. https://doi.org/10.1109/TITS.2021.3131637.

Cai, X., Goh, C.J., 1994. A fast heuristic for the train scheduling problem. Comput. Oper. Res. 21, 499–510. https://doi.org/10.1016/0305-0548(94)90099-X.

Carey, M., 1994a. Extending a train pathing model from one-way to two-way track. Transport. Res. Part B 28, 395–400. https://doi.org/10.1016/0191-2615(94)90038-8.

Carey, M., 1994b. A model and strategy for train pathing with choice of lines, platforms, and routes. Transport. Res. Part B 28, 333–353. https://doi.org/10.1016/0191-2615(94)90033-7.

Carey, M., Lockwood, D., 1995. A model, algorithms and strategy for train pathing. J. Oper. Res. Soc. 46, 988–1005. https://doi.org/10.2307/3009909.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2010. A tabu search algorithm for rerouting trains during rail operations. Transp. Res. Part B Methodol. 44, 175–192. https://doi.org/10.1016/J.TRB.2009.05.004.

Cui, Y., Martin, U., Zhao, W., 2016. Calibration of disturbance parameters in railway operational simulation based on reinforcement learning. J. Rail Transp. Plan. Manag. 6, 1–12. https://doi.org/10.1016/j.jrtpm.2016.03.001.

D'Ariano, A., Pacciarelli, D., Pranzo, M., 2007. A branch and bound algorithm for scheduling trains in a railway network. Eur. J. Oper. Res. 183, 643–657. https://doi.org/10.1016/J.EJOR.2006.10.034.

Dong, H., Tian, Z., Spencer, J.W., Fletcher, D., Hajiabady, S., 2024. Bi-Level optimization of sizing and control strategy of hybrid energy storage system in urban rail transit considering substation operation stability. IEEE Trans. Transp. Electrif. 1. https://doi.org/10.1109/TTE.2024.3385821.

Dong, H., Tian, Z., Spencer, J.W., Fletcher, D., Hajiabady, S., 2023. Coordinated control strategy of railway multisource traction system with energy storage and renewable energy. IEEE Trans. Intell. Transport. Syst. 24, 15702–15713. https://doi.org/10.1109/TITS.2023.3271464.

Dündar, S., Şahin, I., 2013. Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways. Transport. Res. C Emerg. Technol. 27, 1–15. https://doi.org/10.1016/J.TRC.2012.11.001.

Eaton, J., Yang, S., Gongora, M., 2017. Ant colony optimization for simulated dynamic multi-objective railway junction rescheduling. IEEE Trans. Intell. Transport. Syst. 18, 2980–2992. https://doi.org/10.1109/TITS.2017.2665042.

Higgins, A., Kozan, E., Ferreira, L., 1997. Heuristic techniques for single line train scheduling. J. Heuristics 3, 43–62. https://doi.org/10.1023/A:1009672832658.

Higgins, A., Kozan, E., Ferreira, L., 1996. Optimal scheduling of trains on a single line track. Transp. Res. Part B Methodol. 30, 147–161. https://doi.org/10.1016/0191-2615(95)00022-4.

Hong, X., Meng, L., D'Ariano, A., Veelenturf, L.P., Long, S., Corman, F., 2021. Integrated optimization of capacitated train rescheduling and passenger reassignment under disruptions. Transport. Res. C Emerg. Technol. 125, 103025. https://doi.org/10.1016/j.trc.2021.103025.

Josyula, S.P., Törnquist Krasemann, J., Lundberg, L., 2018. A parallel algorithm for train rescheduling. Transport. Res. C Emerg. Technol. 95, 545–569. https://doi.org/10.1016/J.TRC.2018.07.003.

Jovanovic, D., Harker, P., 1991. Tactical scheduling of rail operations: the SCAN I system. Transport. Sci. 25, 46–64.

Khadilkar, H., 2019. A scalable reinforcement learning algorithm for scheduling railway lines. IEEE Trans. Intell. Transport. Syst. 20, 727–736. https://doi.org/10.1109/TITS.2018.2829165.

Kraay, D., Harker, P.T., Chen, B., 1991. Optimal pacing of trains in freight railroads: model formulation and solution. Oper. Res. 39, 82–99.

Li, F., Sheu, J.B., Gao, Z.Y., 2014. Deadlock analysis, prevention and train optimal travel mechanism in single-track railway system. Transp. Res. Part B Methodol. 68, 385–414. https://doi.org/10.1016/J.TRB.2014.06.014.

Lin, Z., Kwan, R.S.K., 2016. A branch-and-price approach for solving the train unit scheduling problem. Transp. Res. Part B Methodol. 94, 97–120. https://doi.org/10.1016/j.trb.2016.09.007.

Liu, J., Chen, L., Roberts, C., Nicholson, G., Ai, B., 2019. Algorithm and peer-to-peer negotiation strategies for train dispatching problems in railway bottleneck sections. IET Intell. Transp. Syst. 13, 1717–1725. https://doi.org/10.1049/iet-its.2019.0020.

Liu, J., Liu, R., 2023. Schedule extra train(s) into existing timetable using actor-critic reinforcement learning. In: 26th IEEE International Conference on Intelligent Transportation Systems. IEEE, Bilbao, Bizkaia, Spain.

Melnikov, L., Michele, G., Raphaelle, C., Michel, L., Nicola, S., Philippe, L., 2024. The journey toward AI-enabled railway companies. https://doi.org/10.13140/RG.2.2.15261.32484.

Ning, L., Li, Y., Zhou, M., Song, H., Dong, H., 2019. A deep reinforcement learning approach to high-speed train timetable rescheduling under disturbances. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 3469–3474. https://doi.org/10.1109/ITSC.2019.8917180.

Obara, M., Kashiyama, T., Sekimoto, Y., 2018. Deep reinforcement learning approach for train rescheduling utilizing graph theory. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 4525–4533. https://doi.org/10.1109/BigData.2018.8622214.

RSSB, 2022. Assisted VSTP [WWW Document]. Br. Rail Saf. Stand. Board. URL. https://www.rssb.co.uk/research-catalogue/CatalogueItem/I01-CLR-06.

Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., Srdic, A., 2016. Reinforcement learning approach for train rescheduling on a single-track railway. Transp. Res. Part B Methodol. 86, 250–267. https://doi.org/10.1016/j.trb.2016.01.004.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: an Introduction. MIT press.

Törnquist, J., Persson, J.A., 2007. N-tracked railway traffic re-scheduling during disturbances. Transp. Res. Part B Methodol. 41, 342–362. https://doi.org/10.1016/J.TRB.2006.06.001.

Veelenturf, L.P., Potthoff, D., Huisman, D., Kroon, L.G., Maróti, G., Wagelmans, A.P.M., 2014. A quasi-robust optimization approach for crew rescheduling. Transport. Sci. 50, 204–215. https://doi.org/10.1287/trsc.2014.0545.

Wang, M., Wang, L., Xu, X., Qin, Y., Qin, L., 2019. Genetic algorithm-based Particle swarm optimization approach to reschedule high-speed railway timetables: a case study in China. J. Adv. Transport. 2019. https://doi.org/10.1155/2019/6090742.

Wang, Y., 2019. Incorporating Weather Impact in Railway Traffic Control. The University of Leeds.

Ying, C., Chow, A.H.F., Nguyen, H.T.M., Chin, K.-S., 2022. Multi-agent deep reinforcement learning for adaptive coordinated metro service operations with flexible train composition. Transp. Res. Part B Methodol. 161, 36–59. https://doi.org/10.1016/J.TRB.2022.05.001.

Ying, C. shuo, Chow, A.H.F., Chin, K.S., 2020. An actor-critic deep reinforcement learning approach for metro train scheduling with rolling stock circulation under stochastic demand. Transp. Res. Part B Methodol. 140, 210–235. https://doi.org/10.1016/J.TRB.2020.08.005.

Yue, P., Jin, Y., Dai, X., Feng, Z., Cui, D., 2024. Reinforcement learning for online dispatching policy in real-time train timetable rescheduling. IEEE Trans. Intell. Transport. Syst. 25, 478–490. https://doi.org/10.1109/TITS.2023.3305074.

Zhan, S., Wong, S.C., Shang, P., Lo, S.M., 2022. Train rescheduling in a major disruption on a high-speed railway network with seat reservation. Transp. A Transp. Sci. 18, 532–567. https://doi.org/10.1080/23249935.2021.1877369.

Zhu, Y., Wang, H., Goverde, R.M.P., 2020. Reinforcement learning in railway timetable rescheduling. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6. https://doi.org/10.1109/ITSC45102.2020.9294188.