A review of real-time railway and metro rescheduling models using learning algorithms

Author(s):
Jusup, Matej; Trivella, Alessio; Corman, Francesco

Publication date:
2021-09

Permanent link:
https://doi.org/10.3929/ethz-b-000504155

A Review of Real-time Railway and Metro Rescheduling Models using Learning Algorithms

Matej Jusup

Alessio Trivella

Francesco Corman

Institute for Transport Planning and Systems

September 2021

STRC

21th Swiss Transport Research Conference

Monte Verità / Ascona, September 12 – 14, 2021

Institute for Transport Planning and Systems

A Review of Real-time Railway and Metro Rescheduling Models using Learning Algorithms

Matej Jusup, Alessio Trivella, Francesco
Corman
Institute for Transport Planning and Systems
ETH Zurich
Stefano-Franscini-Platz 5, 8093 Zurich
phone: +41-44-633 26 52
fax: +41-44-633 26 52
{matej.jusup,alessio.trivella,francesco.corman}@ivt.baug.ethzch

September 2021

Abstract

Planning railway and metro systems includes the critical step of finding a schedule for the trains. Although buffer times and running supplements are added to the schedule to make operations resilient to minor disturbances, they do not protect against all possible events that may lead to conflicts during everyday operations. Thus, real-time train rescheduling models are needed to restore feasibility using actions such as retiming, reordering, rerouting, overtaking or cancelling of trains. Unfortunately, despite many rescheduling models that have been developed in the literature, only a few can learn actions from past, simulated, or ongoing events and cope with disturbances and disruptions' stochastic nature. However, the last decade's expansion of learning algorithms is gaining momentum in the train rescheduling literature by bringing promising novel ideas. This paper aims to review the state-of-the-art learning algorithms applied to the real-time railway and metro rescheduling, identifying challenges and opportunities while making a parallel with other areas where learning algorithms led to breakthroughs.

Keywords

reinforcement learning, approximiate dynamic programming, train rescheduling, delay propagation

1 Introduction

A rail system requires a schedule to operate, which is usually planned several months in advance. The schedule includes buffer times and running time supplements to be resilient to minor delays (Kroon *et al.,* 2008), but which is not enough to avoid all potential conflicts among trains (e.g. headway violations) arising due to unexpected events in daily operations. When disturbances lead to delays and conflicts, the dispatcher's job is to resolve them in real-time and bring services back to normal by performing rescheduling actions that include retiming, reordering, rerouting, overtaking or cancellation. While disturbances directly cause primary (or initial) delays making them unavoidable, secondary (or consecutive, or knock-on) delays result from the propagation of primary delays through the network and can be reduced or prevented by rescheduling actions. Thus, rescheduling aims to compute an updated conflict-free schedule, minimizing deviations from the original schedule, which requires solving an optimization problem. Train rescheduling (TR) is essential to provide services that adhere to the planned schedule, resulting in efficient infrastructure utilization and passenger satisfaction.

Due to its importance, TR has become a thoroughly researched topic in rail transport planning, and a variety of rescheduling models have been developed in the literature, including both deterministic and stochastic models. Deterministic models assume perfect information of the system's future evolution, which is a simplification of rail operations but often needed to achieve tractability (Törnquist, 2006, Cacchiani *et al.,* 2014). On the other hand, stochastic models describe unknown future conditions such as train delays, passenger demand, and energy consumption using probability distributions or stochastic processes, yielding more realistic models at the expense of additional computational complexity. Corman and Meng (2014) overview the online dynamic rescheduling literature, highlighting the stochastic nature of the problem and discussing how to include it in future research. With the recent development of learning algorithms and the increase of computational power, adaptive stochastic rescheduling models have started to become tractable. This review can hence be seen as a follow-up of Corman and Meng (2014) tracking the progress of adaptive models under uncertainty. Note that in the literature, terms like real-time, online, dynamic and adaptive models are often used interchangeably, but some authors make a subtle distinction between them. In this paper, we use real-time or online to emphasize a quick execution during operations, i.e. within 1–2 minutes, and reserve the terms adaptive, dynamic and learning for models that can adjust their behaviour over time based on a new stream of information.

The rescheduling models in the literature also differ by the type of disturbances, type of rail system considered, and rescheduling objective. Commonly, disturbances lead to minor delays caused, e.g. by a change in speed limits due to weather or extended passenger alighting, while

disruptions lead to major delays induced, e.g. by a line blockage, train accidents, or infrastructure damage. In most literature, disturbances cause delays up to 15–20 minutes, whereas disruptions those over 20 minutes, but such a boundary is not strict. Although some researchers have modelled rescheduling during disruption using stochastic programming and reinforcement learning (RL) (Meng and Zhou, 2011, Li *et al.,* 2014, Zhu and Goverde, 2020), this literature is still in its infancy, and most contributions deal with disturbances instead. It is also essential to distinguish between railway, metro, and light rail, which not always have equivalent objectives and actions. Although minimizing delays is the most common objective, especially for the railway, some metro and light rail models have focused on minimizing energy consumption or passenger delays. Regarding actions, overtaking is important in railway but is generally not supported by metro and light rail infrastructure, where the focus is instead on a more precise retiming. Despite these differences, considerable similarities in these systems suggest that solving one of them efficiently may lead to a solution of the others with minor adaptations to the objective function, state and action spaces. For instance, some authors have used a weighted sum of multiple criteria as objective, allowing seeing the three systems under a common lens. Thus, in this paper, we cover all of them and point out the differences when needed.

Optimizing rescheduling actions in a rail network is a complex task, and learning models are emerging as a promising direction to tackle it. This work reviews the literature on adaptive rescheduling in railway, metro, and light rail systems using learning algorithms. We identified 13 papers falling within this scope, with 7 being journal papers. Zou *et al.* (2006) was the first such work in the rescheduling domain and the only one on the light rail to this date. Almost a decade later, Yin *et al.* (2014) proposed a learning model for metro rescheduling. Šemrov *et al.* (2016) presented the first learning model for the railway, and together with Yin *et al.* (2016) can be considered as a seminal work. From 2018 onward, learning algorithms have received growing attention, and we expect this trend to continue. Having a cohesive overview across domains is important since discoveries in one area might motivate novel ideas in others.

It is worth mentioning that learning algorithms have also been applied to train and traffic-related problems other than rescheduling, including train marshalling (Hirashima, 2011), railway access negotiation (Wong and Ho, 2010), train trajectory optimization (Wang *et al.,* 2020), and traffic signal control (Abdulhai *et al.,* 2003, Shoufeng *et al.,* 2008, Arel *et al.,* 2010). We also want to emphasize that although adaptive robust formulations fit within our review scope, we have not come across them. Nonetheless, robust optimization is an active area of research for scheduling under uncertainty, where schedules can be obtained by setting uncertainty limits that encode the desired level of stability (Fischetti *et al.,* 2009, Meng and Zhou, 2011, Shafia *et al.,* 2011, Cacchiani and Toth, 2012, Corman *et al.,* 2014, Hassannayebi *et al.,* 2017, Zhu and Goverde, 2020).

The rest of the paper is structured as follows. In §2, we introduce the control setup. In §3, we classify and compare the literature based on the problem type, reward function, state space, action space, stochasticity, learning algorithm, and network size. We discuss our findings in §4 and conclude in §5.

2 Control Setup

The control setup is illustrated in Figure 1 and consists of an environment, agent, state space, action space, transition function, and reward function. The environment includes the infrastructure, rolling stock, uncertainty and other internal and external factors in TR. The infrastructure can be a line connecting two main stations with intermediary stations in between or a network made of multiple lines that meet at junction points. Based on the level of details considered, there exist macroscopic, microscopic and mesoscopic infrastructure representations. Lines can be uni- or bi-directional and consist of single- or multi-tracks. Uncertainty might consist of train running time and passenger demand distributions. The agent is the dispatcher who observes the environment's state and can control it by performing rescheduling actions. The state space is the environment's representation available to the agent and must include all information needed for making decisions. For instance, it might consist of the location and speed of trains and block section signalling colour. The state space might also include random components, e.g. line blockage scenarios, train delay distribution, or passenger demand distribution. The action space is a set of rescheduling operations available to the dispatcher to utilize the resources in the best possible way. For instance, the dispatcher may influence train order or timing via signalling or by changing train speed, as shown in Figure 2. Since the action space is subject to many physical constraints (e.g. station capacity, block section availability, speed limits, train capacity, headway, dwelling and running time), the task of finding rescheduling decisions is particularly challenging. The transition function describes how the state of the system changes when performing an action from the current state. Finally, the reward function informs the agent about the quality of her decisions. For instance, by setting a reward equal to the amount of delay, the dispatcher can understand whether a decision may lead to lower future delays. Other domain-specific examples of reward functions are energy utilization, passenger satisfaction, or, most often, linear or non-linear combination of the mentioned objectives.

Formally, the TR problem can be formulated as a finite-horizon discrete-time Markov decision process. We consider discrete stages representing the rescheduling horizon $t = 0, \ldots, T$, where $t = 0$ is the current time. For a given t, we denote states and actions by $S_t \in \mathcal{S}_t$ and $a_t \in \mathcal{A}_t$, respectively, where S_t may include a random component. The reward $C_t(S_t, a_t)$ is a real-valued

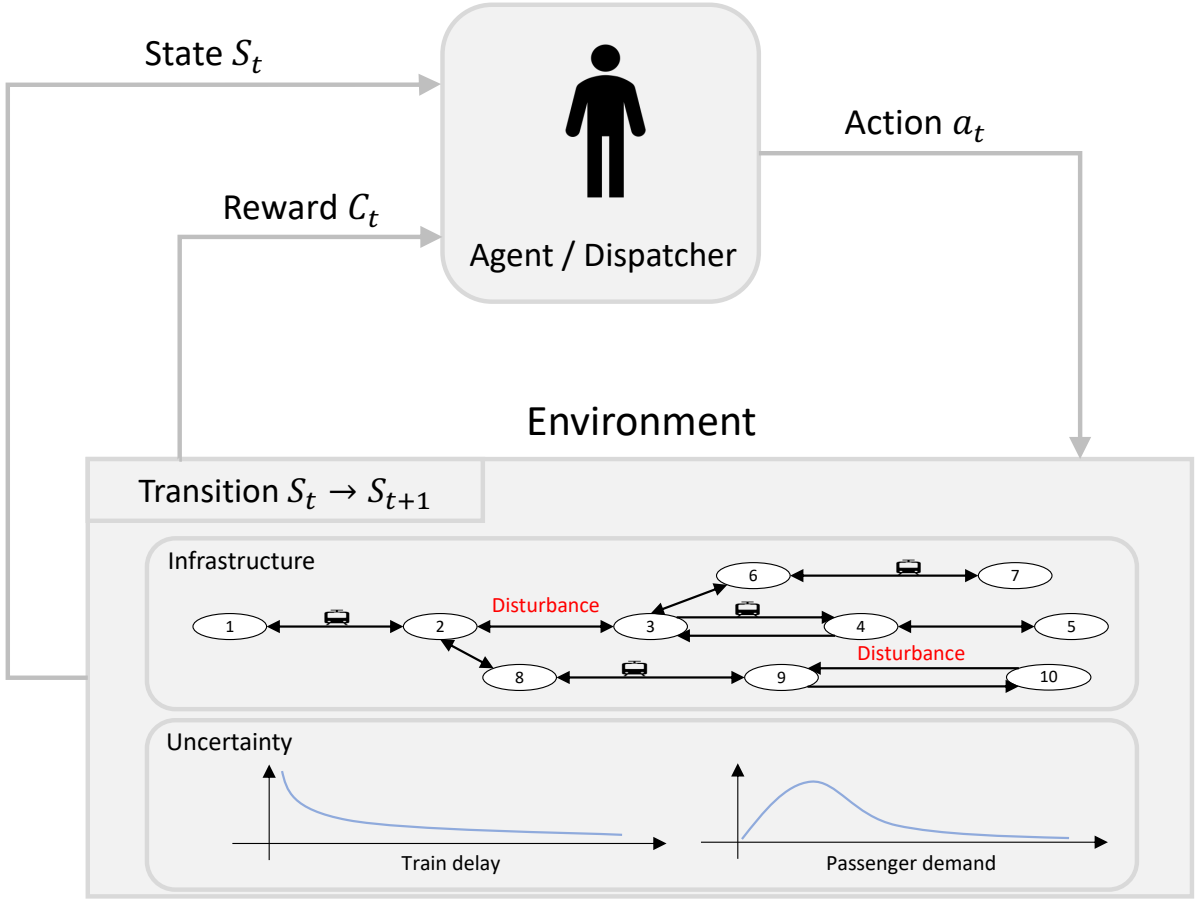

Figure 1: Control setup.

function of state and action, and the transition function is denoted by $S_{t+1} = f(S_t, a_t)$. We call a policy π a collection of decision rules $\{A_t^\pi(\cdot),\ t = 0, \ldots, T\}$ mapping states to actions at each stage t. Given the set of feasible policies Π, the goal of the agent is to find a policy $\pi \in \Pi$

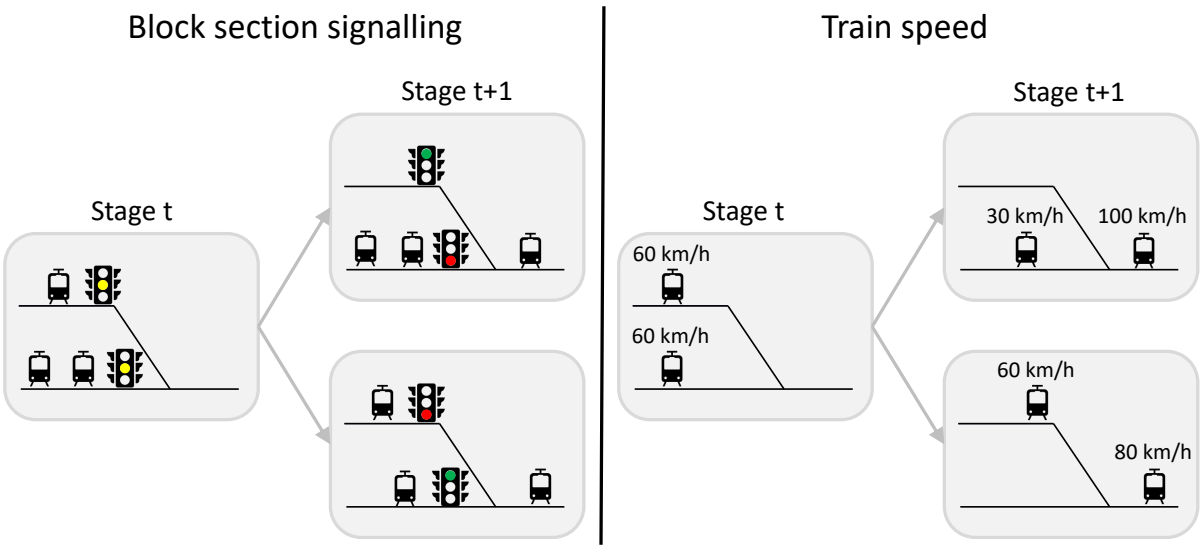

Figure 2: Sample rescheduling actions.

maximizing the expected cumulative discounted reward obtained during the horizon, conditional on the initial state S_0, i.e.

$$\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t C_t(S_t, A_t^{\pi}(S_t)) \,\middle|\, S_0 \right], \tag{1}$$

where γ is a discount factor. Model (1) is intractable even for the smallest problems, but it is known that the Bellman's equation allows its reformulation as a stochastic dynamic program by introducing a value function $V_t(S_t)$. Specifically, setting boundary conditions $V_{T+1}(S_{T+1}) = 0$, we can define for $t = T$ backward to $t = 0$ and each state $S_t \in \mathcal{S}_t$:

$$V_t(S_t) = \max_{a_t \in \mathcal{A}_t} \{C_t(S_t, a_t) + \gamma \mathbb{E}[V_{t+1}(f(S_t, a_t)) \,|\, S_t]\}. \tag{2}$$

Unfortunately, this formulation is also often intractable due to curses of dimensionality, such as a high-dimensional or continuous state or action space, which prevent Equation (2) from being solved directly by backward recursion (Powell, 2007). Approximate dynamic programming (ADP) comprises a broad set of methods to derive heuristic policies and overcome these curses of dimensionality, for instance, by computing a value function approximation $\hat{V}_t(S_t) \simeq V_t(S_t)$. To the best of our knowledge, all real-time models for solving stochastic adaptive TR problems within the scope of our review use or develop some ADP[1] technique.

3 Literature Classification

Learning algorithms are at the core of making Equation (2) tractable, so it is crucial to understand which algorithms have proven to perform well in which situation. Therefore, in this section, we classify the existing literature based on problem type (§3.1), state space (§3.2), action space (§3.3), reward function (§3.4), source of stochasticity (§3.5), network size (§3.6), and learning algorithm (§3.7). We remark that these categories should not be viewed as independent since they affect each other during the modelling phase. For instance, the state space representation may affect the actions and stochasticity design. Even though the transition function is an integral component of Equation (2), the authors tend not to report it explicitly while deducing it is a challenging task. Thus, we omit it not to introduce noise into the classification. We also neglect a classification of the infrastructure since the extant literature focuses on a single line. We believe this is due to models of entire rail networks remaining intractable at the moment, as we discuss later.

[1] We use in this paper ADP and RL interchangeably.

In Table 1, we provide a classification of the reviewed papers[2]. We report the notation used in this table in the supplementary Tables 2–8. The state space column displays various choices, which underscores the complexity and variety of representation ideas. We relegate to the appendix additional classification details and further information on infrastructure representation, test cases, computational resources and others. In the following, we elaborate on each of the classification criteria presented in Table 1 in connection with the related literature.

Reference	Problem	Reward	State space	Actions	Stochasticity	Algorithm	Size
Zou *et al.* (2006)	L	P, D	NTS, ATP, TDR	B	P	SARSA	M
Yin *et al.* (2014)	M	T, D, E	TL, RT	S	P	Q	S
Yin *et al.* (2015)	M	T, D, E	TL, RT, PD	S	P, T	Q	S
Šemrov *et al.* (2016)	R	T, D	TL, BA	B	T	Q	S
Yin *et al.* (2016)	M	P, D, E	DT, DD, AT, IVP, WP	S	P, T	Q	S
Khadilkar (2018)	R	T, D	PL, CL	B	T	Q	L
Schön and König (2018)	R	P, D	IVT, TD, FTD, CTD	S	T	H	M
Liu *et al.* (2018)	M	T, P, D	AD, HD	S	P, T	DDPG	S
Ghasempour and Heydecker (2019)	R	T, D	TP, BS	T	T	TD	S
Ghasempour *et al.* (2019)	R	T, D	CAT, RWA	B	T	TD	L*
Yang *et al.* (2019)	M	T, E	TD, DW, CNT, PNT, SNT	S, T	T	DDPG	L
Ning *et al.* (2019)	R	T, D	ATM, DTM	B	T	Q	S
Zhu *et al.* (2020)	R	T, D	ED, CL, TL	S	T	Q	S

Table 1: Literature classification

3.1 Problem Type

As previously mentioned, we distinguish between railway, metro, and light rail models. The main differences among those are in the objective function/reward, actions, and possible sources of randomness. Nonetheless, all models adhere to Equation (2) with a similar environment and agent. Thus, it is reasonable to assume that efficient solution frameworks for one problem type could be adapted to another type with limited effort. From Table 1, we observe that railway models emphasize minimizing train delays, whereas metro models focus on energy consumption and light rail models on passenger delays. However, some authors have considered a weighted sum of these objectives (Yin *et al.,* 2014, Liu *et al.,* 2018). Actions differ due to problem-specific details. While it is not usual to reorder trains in the metro and especially in light rail rescheduling due to shorter travel distance, it is often an important source of flexibility in the railway that originates from speed differences among trains. Sources of randomness differ due to external factors under which the rail operates and should be carefully analyzed, modelled and estimated. For instance, metro systems are less subject to weather uncertainty but more to passenger

[2]Authors not always mention all details we are interested in explicitly, in which case we deduce it from other available information.

boarding time. Each model then imposes a set of operational constraints that are, in most cases, based on dwell and headway times or speed limits.

3.2 State Space

The state space definition is critical as inadequate choices may lead to curses of dimensionality, causing intractability (Powell, 2007). Since the environment we want to model is highly complex and the idea of applying ADP for TR only recently came under research focus, the discussion about how to define a state space for a rail line is ongoing, whereas defining a tractable state space for a general network is still an open question. Researchers have experimented with various state space representations, and there is room for creativity to come up with realistic and tractable ones, i.e. there is no single correct answer. Even the seemingly simple choice between microscopic, macroscopic and mesoscopic network representation is hard. One of the core decisions is how to define a time variable, i.e. in how many time points/stages the agent can make rescheduling actions. Moreover, there are different choices at our disposal on how to represent a direction (in case of bi-directional lines), train location, number of in-vehicle passengers, passenger demand, (block) section availability, disturbance time and duration, arrival time, dwelling time, train speed and many others.

In general, even small problems with few trains or stations can quickly become intractable. The state space in the existing literature is predominantly represented by n-dimensional vectors. For instance, Yang *et al.* (2019) use a vector consisting of train number, its last dwelling time, the control strategy, and the current speed and position of the other running trains. Ning *et al.* (2019) are the only ones who tried an $N \times M$ matrix representation. Unfortunately, existing representations appear too computationally expensive for a microscopic network or a long horizon with many decision stages. This most likely explains why almost all work is concentrated on a critical line within the network during peak hours. To expand the spatial complexity of the model, Ghasempour *et al.* (2019) proposed making distributed (i.e. independent) decisions at important junctions of a network, which is similar in spirit to existing deterministic models (Corman *et al.,* 2010). To make a state-action space invariant to the size of the network, Khadilkar (2018) introduced a local block-section-oriented neighbourhood for each train. Yang *et al.* (2019) proposed a similar idea based on train-oriented neighbourhoods. One potential drawback of such representations is that accounting for rerouting actions becomes hardly possible as small neighbourhoods cannot capture entire train routes and network dynamics. Since this field of research is quite novel and rapidly expanding, major improvements are expected in the next few years. Especially, we expect that models capturing a whole network dynamics will emerge, thereby enabling a richer spectrum or rescheduling decisions.

3.3 Action Space

Actions should be designed such that the dispatcher can easily implement them during operations. Depending on the environment and state space design, they come under three categories: station-level, block-section-level, and train-level. Station-level actions include varying the dwelling or departure time and adjusting the running time until the next station (Schön and König, 2018, Liu *et al.,* 2018). Block-section-level actions consist of modifying the signalling availability of the next section (Zou *et al.,* 2006, Šemrov *et al.,* 2016). The only train-level action in the literature in our review scope is adjusting the cruising speed (Yang *et al.,* 2019). Allowing for multiple or complex actions may also lead to the curses of dimensionality typical of ADP. Discrete actions like signalling block section availability are useful in a multi-track network as they allow the agent to reorder trains, for instance. Nevertheless, the number of such actions grows exponentially with the number of trains and decision stages. Continuous actions such as adjustment of dwelling and running times have similar expressive power but also lead to additional computational issues.

3.4 Reward Function

The reward function encodes the optimization goal of the agent and guides the learning algorithm on how to take actions by getting feedback from the environment. In all cases we have encountered, the reward is a function of train delay, train running time, passenger delay, passenger travelling time, and energy utilization. The linear or non-linear (weighted) sum of these components is also common, while a set of rules for calculating the rewards is specified in rare cases (Khadilkar, 2018, Zhu *et al.,* 2020). An example of user-defined, rule-based reward is assigning a value of $+1$ if the sum of priority-weighted total delay is under a certain threshold and a value of -1 otherwise (Khadilkar, 2018). In railway rescheduling, the total or secondary train delays are used in all papers except for one in which passenger delays are chosen (Schön and König, 2018). As discussed, metro applications are instead usually more focused on energy consumption in combination with passenger or train delay. Liu *et al.* (2018) found it important to penalize late arrivals at the terminal station, keeping a regular headway. Zou *et al.* (2006) proposed an architecture composed of models with local rewards that cooperate via a global reinforcement signal.

3.5 Stochasticity

Capturing the stochasticity of rail operations is vital to improving real-time rescheduling decisions. Deterministic models cannot look beyond the observed current state unless we assume perfect information on future states. Improving over deterministic models is possible by robust optimization approaches that account for a certain amount of risk or by scenario-based techniques, which sample a discrete set of scenarios from probability distributions but are computationally practical only for a limited number of scenarios (Corman and Meng, 2014). When primary delays propagate through the network, it would be beneficial for a rescheduling model to account for the severity of the secondary delays when making decisions.

Randomness can be modelled by using either probability distributions (Yin *et al.,* 2014, Khadilkar, 2018, Ghasempour and Heydecker, 2019) or stochastic processes (Yin *et al.,* 2016, Liu *et al.,* 2018), which are more realistic but need expanding the state space with an exogenous component to be tracked. Thus, models of uncertainty dynamics have to be chosen carefully to balance a realistic description of operational uncertainty with a tractable state space. Moreover, such models require calibration based on data, which may be complicated (Trivella and Corman, 2019). As a consequence, most authors have so far used simple independent identically distributed probability distributions. The randomness studied in the existing literature includes train delays, dwelling and running times, and in a few cases, passenger demand or loading time. The most commonly used distributions are the Weibull, Poisson, and exponential distribution (Meester and Muns, 2007, Büker and Seybold, 2012).

3.6 Network Size

Due to the high computational complexity, all models up to this date were trained and tested on a single rail line and based on an environment simulating real-world lines. We have not come across any learning model tested during actual operations. Our classification uses a simple rule that a small line consists of less than 15 stations, a medium line of 15–25, and a large line of 25 or more. Accordingly, we identified eight small test cases, two medium-sized test cases and three large test cases. Note that such classification does not provide the complete picture because the number of trains, the time horizon, and the line representation also play a role in defining an instance and its size. For example, a microscopic representation often leads to an intractable problem, which motivates the choice of a small test case. Furthermore, there is no community-wide benchmark for comparing the results leading to authors usually trying to mimic the rail line of a city or a country by using commercial or their own simulators. A

need for solving specific problems or sources of funding might also motivate such choices. In combination with the lack of reported information in the numerical experiments, it makes it hard to make more detailed analyses.

Ghasempour *et al.* (2019) falls in the large test case category with a network consisting of 32 stations. However, this model makes independent rescheduling decisions on two portions of the network consisting of 4 stations each. Khadilkar (2018) and Yang *et al.* (2019) report 60 and 28 stations, respectively, which they could tackle only using the local-neighbourhood based state representation described in §3.2. On top of that, since this research area is in its infancy, we believe that most researchers try to achieve tractability on small instances with the potential for follow-up work on larger instances. Thus, most test cases are still very small due to the problem complexity, real-time execution requirement, novelty of the approach, and potentially limited computational resources. Unfortunately, even for small test cases, the training time is generally high and often not even reported in the papers.

3.7 Learning Algorithm

In the context of adaptive TR, four well-known learning algorithms have been reported. Q-learning (Watkins and Dayan, 1992) dominates the field by being used in more than half of the papers (Šemrov *et al.,* 2016, Zhu *et al.,* 2020). State–action–reward–state–action (SARSA) (Zou *et al.,* 2006), a temporal difference (TD) (Ghasempour *et al.,* 2019) and deep deterministic policy gradient (DDPG) (Yang *et al.,* 2019) are the algorithms used in the rest of the available literature, except for Schön and König (2018). They developed a heuristic method for solving a backward recursion using domain-specific knowledge. Details about these and other algorithms can be found, e.g. in Powell (2007). One important difference that we want to point out is that Q-learning operates under the assumption of discrete actions while DDPG handles continuous actions (Lillicrap *et al.,* 2015).

The choice of the learning algorithm can affect the quality of rescheduling decisions as well as the training time, online performance and adaptiveness. Because of the discussed high complexity of state space and actions, applying a suitable learning algorithm can make the difference between tractability and intractability. Only recently a substantial progress has been made in the domain of ADP/RL applications for efficient approximations of Equation (2); one of the major successes is AlphaZero, with an RL-based model beating the best human player in the board game Go two decades sooner than expected (Silver *et al.,* 2018). Despite significant differences in the control setup, such steep progress gives hope that also the adaptive stochastic TR problem for an entire network will be solved efficiently in the near future.

4 Discussion

The application of learning algorithms for real-time TR is still novel and exclusively focuses on ADP methods. Due to the problem complexity, existing control setups lead to intractability even for medium-sized lines, while models of entire networks have not been reported yet. If we compare it with the evolution of learning algorithms in other application areas, we can draw multiple conclusions.

Firstly, the control setup can always get better by developing new ideas that may result in higher quality and more computationally efficient solutions. One of the best examples is natural language processing, which has gone through multiple development stages that often involved architecture redesign (Hochreiter and Schmidhuber, 1997, Bahdanau *et al.,* 2014, Vaswani *et al.,* 2017).

Secondly, the successful application of learning algorithms to real-world problems in other fields has often required an enormous amount of computational resources. In most cases, a distributed network of graphical processing units (GPUs) (Brown *et al.,* 2020) or tensor processing units (TPUs) (Silver *et al.,* 2018). Unfortunately, TR models have not been tested on such a scale yet, but we believe this will be required soon to tackle larger instances and more complex networks.

Thirdly, there are other classes of learning frameworks or algorithms with the potential to be successfully applied. For instance, TR has a nice graph representation (D'ariano *et al.,* 2007) which might be a fruitful ground for graph neural networks (GNNs) (Wu *et al.,* 2020). GNNs offer the possibility of supervised, semi-supervised, and what might be of particular interest unsupervised learning (Zhou *et al.,* 2020). Link prediction might be used for making decisions under disturbances and node classification for conflict detection. Trying to acquaint and utilize novel developments in machine learning should be encouraged because it may lead to quicker progress.

Lastly, almost a decade ago, Corman and Meng (2014) raised the need for a global test case. Other areas like computer vision, natural language processing, board games and others have proven that having a benchmark increases competition and can lead to much faster progress. A promising step in that direction happened when the Flatland competition—TR under disruption—was organized on the AIcrowd platform during NeurIPS2020 (Laurent *et al.,* 2021). It was sponsored by Swiss Federal Railway Company (SBB), Deutsche Bahn and Société Nationale des Chemins de fer Français. SBB and AIcrowd also developed an open-source Python package (SBB and AIcrowd, 2019) to make it easier to develop and compare RL models for TR. Such

developments might bring other benefits like transfer learning (Taylor and Stone, 2009), which was already briefly discussed in Khadilkar (2018). Transfer learning often leads to reduced computational requirements since pre-trained models are specialized for specific instances of the problem. An excellent example of transfer learning is computer vision, where a convolutional neural network is pre-trained on a huge set of images and specialized from thereon.

5 Conclusions and future research

This paper reviews the current state of research for solving the real-time railway and metro rescheduling problem under uncertainty using learning algorithms. We envisage four promising lines for future advances: (i) further improving the existing ADP-based solutions, (ii) expanding the methodological focus by applying different classes of learning algorithms, (iii) exploiting larger computational power, especially during the training phase, and (iv) working on a community-wide benchmark and reporting the results achieved on it. Since learning algorithms in this field are at the outset, they have been employed sparsely in the literature. However, the growing number of publications in the last three years suggests that they will be given much more attention in the years to come.

In terms of learning methods, the paper's focus has been on ADP because the literature has exclusively considered this class. While the existing research sets a solid ground for further improvements of ADP-based approaches, different learning techniques such as GNNs seem promising and might be tried in the future.

One of the main challenges is scaling up the spatial complexity from small lines to larger instances, or ideally, whole networks, which unfortunately has not been reported yet. As a consequence, it is hard to imagine railway companies being confident in utilizing such solutions. Once a breakthrough from small lines to medium- or large-sized networks is made, we might see the first applications outside of simulated environments, as was the case for deterministic rescheduling models about a decade ago.

Acknowledgement

The authors acknowledge support by the Swiss National Science Foundation under the research project DADA/181210.

Symbol	Meaning
L	light rail
M	metro
R	railway

Table 2: Problem type notation.

Symbol	Meaning
T	train
P	passenger
D	delay
E	energy

Table 3: Reward input notation.

Symbol	Meaning
NTS	number of trains in the blocking section
ATP	average number of passengers (in all trains) in a section
TL	train location
NP	number of passengers in a train
TDR	train direction
RT	reserved trip time, i.e. total time remaining before reaching the final station
PD	passenger demand
BA	block section availability
DT	disturbance time
DD	disturbance duration
AT	arrival time
AD	deviation from scheduled arrival time
HD	deviation from scheduled headway
IVP	number of in-vehicle passengers
WP	number of waiting passengers
PL	priority levels for multiple blocks before and after current train position
CL	congestion level for multiple blocks before and after current train position
TD	train delay
FTD	feeder train delay
CTD	connecting train delay
CAT	number of trains in control area
RWA	sequence of right-of-way assignments
TN	train number
DT	last train dwelling time
CNT	control strategy of neighbouring trains
SNT	speed of neighbouring trains
PNT	position of neighbouring trains
TP	permutation of train orders in the control area
BS	block section signalling state
ED	event delay, i.e. a simulation of a departure or arrival of a train at a station
ATM	matrix of possible arrival times for each train and station
DTM	matrix of possible departure times for each train and station

Table 4: State space notation.

Symbol	Meaning
S	station level
B	block section level
T	train level

Table 5: Actions notation.

Symbol	Meaning
SARSA	state–action–reward–state–action
Q	Q-learning
DDPG	deep deterministic policy gradient
TD	temporal difference
H	heuristic

Table 6: Learning algorithm notation.

Symbol	Meaning
P	passenger loading/arrival time or demand
T	train delay

Table 7: Stochasticity notation.

Symbol	Meaning
S	small (less than 15 stations)
M	medium (between 15 and 25 stations)
L	large (25 or more stations)
*	small blocks within large network considered

Table 8: Network size notation.

6 References

Abdulhai, B., R. Pringle and G. J. Karakoulas (2003) Reinforcement learning for true adaptive traffic signal control, *Journal of Transportation Engineering*, **129** (3) 278–285.

Arel, I., C. Liu, T. Urbanik and A. G. Kohls (2010) Reinforcement learning-based multi-agent system for network traffic signal control, *IET Intelligent Transport Systems*, **4** (2) 128–135.

Bahdanau, D., K. Cho and Y. Bengio (2014) Neural machine translation by jointly learning to align and translate, *arXiv preprint arXiv:1409.0473*.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.* (2020) Language models are few-shot learners, *arXiv preprint arXiv:2005.14165*.

Büker, T. and B. Seybold (2012) Stochastic modelling of delay propagation in large networks, *Journal of Rail Transport Planning & Management*, **2** (1-2) 34–50.

Cacchiani, V., D. Huisman, M. Kidd, L. Kroon, P. Toth, L. Veelenturf and J. Wagenaar (2014) An overview of recovery models and algorithms for real-time railway rescheduling, *Transportation Research Part B: Methodological*, **63**, 15–37.

Cacchiani, V. and P. Toth (2012) Nominal and robust train timetabling problems, *European Journal of Operational Research*, **219** (3) 727–737.

Corman, F., A. D'Ariano and I. A. Hansen (2014) Evaluating disturbance robustness of railway schedules, *Journal of Intelligent Transportation Systems*, **18** (1) 106–120.

Corman, F., A. D'Ariano, D. Pacciarelli and M. Pranzo (2010) Centralized versus distributed systems to reschedule trains in two dispatching areas, *Public Transport*, **2** (3) 219–247.

Corman, F. and L. Meng (2014) A review of online dynamic models and algorithms for railway traffic management, *IEEE Transactions on Intelligent Transportation Systems*, **16** (3) 1274–1284.

D'ariano, A., D. Pacciarelli and M. Pranzo (2007) A branch and bound algorithm for scheduling trains in a railway network, *European journal of operational research*, **183** (2) 643–657.

Fischetti, M., D. Salvagnin and A. Zanette (2009) Fast approaches to improve the robustness of a railway timetable, *Transportation Science*, **43** (3) 321–335.

Ghasempour, T. and B. Heydecker (2019) Adaptive railway traffic control using approximate dynamic programming, *Transportation Research Procedia*, **38**, 201–221.

Ghasempour, T., G. L. Nicholson, D. Kirkwood, T. Fujiyama and B. Heydecker (2019) Distributed approximate dynamic control for traffic management of busy railway networks, *IEEE Transactions on Intelligent Transportation Systems*, **21** (9) 3788–3798.

Hassannayebi, E., S. H. Zegordi, M. R. Amin-Naseri and M. Yaghini (2017) Train timetabling at rapid rail transit lines: a robust multi-objective stochastic programming approach, *Operational Research*, **17** (2) 435–477.

Hirashima, Y. (2011) A new reinforcement learning system for train marshaling with selectable desired layout, *IFAC Proceedings Volumes*, **44** (1) 6976–6981.

Hochreiter, S. and J. Schmidhuber (1997) Long short-term memory, *Neural computation*, **9** (8) 1735–1780.

Khadilkar, H. (2018) A scalable reinforcement learning algorithm for scheduling railway lines, *IEEE Transactions on Intelligent Transportation Systems*, **20** (2) 727–736.

Kroon, L., G. Maróti, M. R. Helmrich, M. Vromans and R. Dekker (2008) Stochastic improvement of cyclic railway timetables, *Transportation Research Part B: Methodological*, **42** (6) 553–570.

Laurent, F., M. Schneider, C. Scheller, J. Watson, J. Li, Z. Chen, Y. Zheng, S.-H. Chan, K. Makhnev, O. Svidchenko *et al.* (2021) Flatland competition 2020: Mapf and marl for efficient train coordination on a grid world, *arXiv preprint arXiv:2103.16511*.

Li, X., B. Shou and D. Ralescu (2014) Train rescheduling with stochastic recovery time: A new track-backup approach, *IEEE Transactions on systems, man, and cybernetics: systems*, **44** (9) 1216–1233.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra (2015) Continuous control with deep reinforcement learning, *arXiv preprint arXiv:1509.02971*.

Liu, Y., T. Tang, L. Yue, J. Xun and H. Guo (2018) An intelligent train regulation algorithm for metro using deep reinforcement learning, paper presented at the *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 1208–1213.

Meester, L. E. and S. Muns (2007) Stochastic delay propagation in railway networks and phase-type distributions, *Transportation Research Part B: Methodological*, **41** (2) 218–230.

Meng, L. and X. Zhou (2011) Robust single-track train dispatching model under a dynamic and stochastic environment: A scenario-based rolling horizon solution approach, *Transportation Research Part B: Methodological*, **45** (7) 1080–1102.

Ning, L., Y. Li, M. Zhou, H. Song and H. Dong (2019) A deep reinforcement learning approach to high-speed train timetable rescheduling under disturbances, paper presented at the *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 3469–3474.

Powell, W. B. (2007) *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703, John Wiley & Sons.

SBB and AIcrowd (2019) Flatland documentation, `http://flatland-rl-docs.s3-website.eu-central-1.amazonaws.com/`. Accessed 18 June 2021 at `http://flatland-rl-docs.s3-website.eu-central-1.amazonaws.com/`.

Schön, C. and E. König (2018) A stochastic dynamic programming approach for delay management of a single train line, *European Journal of Operational Research*, **271** (2) 501–518.

Šemrov, D., R. Marsetič, M. Žura, L. Todorovski and A. Srdic (2016) Reinforcement learning approach for train rescheduling on a single-track railway, *Transportation Research Part B: Methodological*, **86**, 250–267.

Shafia, M. A., M. P. Aghaee, S. J. Sadjadi and A. Jamili (2011) Robust train timetabling problem: Mathematical model and branch and bound algorithm, *IEEE Transactions on Intelligent Transportation Systems*, **13** (1) 307–317.

Shoufeng, L., L. Ximin and D. Shiqiang (2008) Q-learning for adaptive traffic signal control based on delay minimization strategy, paper presented at the *2008 IEEE International Conference on Networking, Sensing and Control*, 687–691.

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

Taylor, M. E. and P. Stone (2009) Transfer learning for reinforcement learning domains: A survey., *Journal of Machine Learning Research*, **10** (7).

Törnquist, J. (2006) Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms, paper presented at the *5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'05)*.

Trivella, A. and F. Corman (2019) Modeling uncertainty dynamics in public transport optimization, paper presented at the *19th Swiss Transport Research Conference (STRC 2019)*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin (2017) Attention is all you need, *arXiv preprint arXiv:1706.03762*.

Wang, P., A. Trivella, R. M. Goverde and F. Corman (2020) Train trajectory optimization for improved on-time arrival under parametric uncertainty, *Transportation Research Part C: Emerging Technologies*, **119**, 102680.

Watkins, C. J. and P. Dayan (1992) Q-learning, *Machine learning*, **8** (3-4) 279–292.

Wong, S. and T. K. Ho (2010) Intelligent negotiation behaviour model for an open railway access market, *Expert Systems with Applications*, **37** (12) 8109–8118.

Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang and S. Y. Philip (2020) A comprehensive survey on graph neural networks, *IEEE transactions on neural networks and learning systems*.

Yang, G., F. Zhang, C. Gong and S. Zhang (2019) Application of a deep deterministic policy gradient algorithm for energy-aimed timetable rescheduling problem, *Energies*, **12** (18) 3461.

Yin, J., D. Chen, L. Yang, T. Tang and B. Ran (2015) Efficient real-time train operation algorithms with uncertain passenger demands, *IEEE transactions on intelligent transportation systems*, **17** (9) 2600–2612.

Yin, J., D. Chen, W. Zhao and L. Chen (2014) Online adjusting subway timetable by q-learning to save energy consumption in uncertain passenger demand, paper presented at the *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2743–2748.

Yin, J., T. Tang, L. Yang, Z. Gao and B. Ran (2016) Energy-efficient metro train rescheduling with uncertain time-variant passenger demands: An approximate dynamic programming approach, *Transportation Research Part B: Methodological*, **91**, 178–210.

Zhou, J., G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li and M. Sun (2020) Graph neural networks: A review of methods and applications, *AI Open*, **1**, 57–81.

Zhu, Y. and R. M. Goverde (2020) Dynamic and robust timetable rescheduling for uncertain railway disruptions, *Journal of Rail Transport Planning & Management*, **15**, 100196.

Zhu, Y., H. Wang and R. M. Goverde (2020) Reinforcement learning in railway timetable rescheduling, paper presented at the *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6.

Zou, L., J.-m. Xu and L.-x. Zhu (2006) Light rail intelligent dispatching system based on reinforcement learning, paper presented at the *2006 International Conference on Machine Learning and Cybernetics*, 2493–2496.

A Supplementary material: Detailed classification tables

Authors sometimes do not explicitly mention all the details relevant to our review. Such occurrences are mostly related to state space, actions and network details. In such cases, we make an effort to deduce it from other available information. We leave blank cells when the information is missing or when we are unable to deduce it confidently. According to such rules, we fill in Tables 9, 10 and 11 with the correct information to the best of our knowledge. In Tables 12–23, we explain the symbols used.

Reference	Publication	Problem
Zou *et al.* (2006)	C	L
Yin *et al.* (2014)	C	M
Yin *et al.* (2015)	J	M
Šemrov *et al.* (2016)	J	R
Yin *et al.* (2016)	J	M
Khadilkar (2018)	C	R
Schön and König (2018)	J	R
Liu *et al.* (2018)	C	M
Ghasempour and Heydecker (2019)	J	R
Ghasempour *et al.* (2019)	J	R
Yang *et al.* (2019)	J	M
Ning *et al.* (2019)	C	R
Zhu *et al.* (2020)	C	R

Table 9: Publication and problem type.

Reference	Reward	State space	Actions	Stochasticity	Algorithm
Zou *et al.* (2006)	PWT	NTS, ATP, TDR	LR	PL, PA	SARSA
Yin *et al.* (2014)	WS(TD, EC)	TL, RT	RT	PD	Q
Yin *et al.* (2015)	NWS(TD, EC)	RT, PD	DT, RT	PD, TD	Q
Šemrov *et al.* (2016)	TD	TL, BA	SC	TD	Q
Yin *et al.* (2016)	WS(PD, PTT, EC)	DT, DD, AT, IVP, WP	RI, DT, RT	PD, TD	Q
Khadilkar (2018)	TBF(PWD(TD))	PL, CL	LR	TD	Q
Schön and König (2018)	PD	IVT, TD, FTD, CTD	DT	TD	H
Liu *et al.* (2018)	WS(PTT, PA, HR)	AD, HD	DT, RT	PD, TD	DDPG
Ghasempour and Heydecker (2019)	TD	TP, BS	DS	TD	TD
Ghasempour *et al.* (2019)	RTC	CAT, RWA	DS	TD	TD
Yang *et al.* (2019)	RE, TE	TD, DW, CNT, PNT, SNT	CS, DT	TD	DDPG
Ning *et al.* (2019)	AD	ATM, DTM	DS	TD	Q
Zhu *et al.* (2020)	UDF(TD)	ED, CL, TL	EI	TD	Q

Table 10: Control setup.

Reference	Network size	Network type	Track type	Line direction	Infrastructure representation	RAM (GB)	CPU (GHz)
Zou *et al.* (2006)	M	ML	S	U	MI		
Yin *et al.* (2014)	S	L			MA		
Yin *et al.* (2015)	S	L	S		MA		
Šemrov *et al.* (2016)	S	L	S	B	MI	8	3.4
Yin *et al.* (2016)	S	L	S	B	MA		
Khadilkar (2018)	L	L	S, D	B	ME	4	
Schön and König (2018)	M	L	S	U	MA		3.3
Liu *et al.* (2018)	S	L	S		MA		
Ghasempour and Heydecker (2019)	S	L	M	U	MI	32	4
Ghasempour *et al.* (2019)	L*	ML	M	U	MI	32	4
Yang *et al.* (2019)	L	L	S	U	MA		
Ning *et al.* (2019)	S	L	S	U	MA	12	2.5
Zhu *et al.* (2020)	S	L	D	U	MI		

Table 11: Infrastructure details and computational resources.

Symbol	Meaning
C	conference/proceedings
J	journal

Table 12: Publisher notation.

Symbol	Meaning
L	light rail
M	metro
R	railway

Table 13: Problem type notation.

Symbol	Meaning
WS()	weighted sum
NWS()	non-linear weighted sum
PWS()	priority-weighted sum
TBF()	threshold-based function
UDF()	user-defined fuction
PWT	passenger waiting time
TD	train delay
EC	train energy consumption
PD	passenger delay
PTT	passenger travel time
PA	punctual arrival at the terminal station
AD	average delay
HR	headway regularity
RTC	running time of a train through the control area
RE	recovery energy
TE	traction energy

Table 14: Reward notation.

Symbol	Meaning
NTS	number of trains in the blocking section
ATP	average number of passengers (in all trains) in a section
TL	train location
NP	number of passengers in a train
TDR	train direction
RT	reserved trip time
PD	passenger demand
BA	block section availability
DT	disturbance time
DD	disturbance duration
AT	arrival time
AD	deviation from scheduled arrival time
HD	deviation from scheduled headway
IVP	number of in-vehicle passengers
WP	number of waiting passengers
PL	priority levels for multiple blocks before and after current train position
CL	congestion level for multiple blocks before and after current train position
TD	train delay
FTD	feeder train delay
CTD	connecting train delay
CAT	number of trains in control area
RWA	sequence of right-of-way assignments
TN	train number
DT	last train dwelling time
CNT	control strategy of neighbouring trains
SNT	speed of neighbouring trains
PNT	position of neighbouring trains
TP	permutation of train orders in the control area
BS	block section signalling state
ED	event delay—a simulation of a departure or arrival of a train at a station
ATM	matrix of possible arrival times for each train and station
DTM	matrix of possible departure times for each train and station

Table 15: State space notation.

Symbol	Meaning
LR	binary variable indicating if a train can leave current resource
DT	dwelling time in the station
RT	running time for the next resource - determined in the station
SC	signalling colour (stop/go)
RI	indicator signalling if the train should be rescheduled
DS	departure sequence of a trains from a current resource
CS	train cruising speed
EI	indicator signalling if an event is implemented

Table 16: Actions notation.

Symbol	Meaning
PL	passenger loading time
PA	passenger arrival time
PD	passenger demand
TD	train delay

Table 17: Stochasticity notation.

Symbol	Meaning
SARSA	state–action–reward–state–action
Q	Q-learning
DDPG	deep deterministic policy gradient
TD	temporal difference
H	heuristic

Table 18: Learning algorithm notation.

Symbol	Meaning
S	small (less than 15 stations)
M	medium (between 15 and 25 stations)
L	large (25 or more stations)
*	small blocks within large network considered

Table 19: Network size notation.

Symbol	Meaning
L	line connecting two major stations
ML	multiple lines within network

Table 20: Network type notation.

Symbol	Meaning
S	single track
D	double track
M	multi track

Table 21: Track type notation.

Symbol	Meaning
U	uni-directional
B	bi-directional

Table 22: Line direction notation.

Symbol	Meaning
MI	microscopic
MA	macroscopic
ME	mesoscopic

Table 23: Infrastructure representation notation.