Proceedings Paper:
Liu, J. orcid.org/0000-0002-3808-5957 and Liu, R. orcid.org/0000-0003-0627-3184 (2024) Schedule Extra Train(s) into Existing Timetable Using Actor-Critic Reinforcement Learning. In: 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), 24-28 Sep 2023, Bilbao, Spain. IEEE , pp. 1166-1171. ISBN 979-8-3503-9947-9

https://doi.org/10.1109/itsc57777.2023.10422338

Schedule Extra Train(s) into Existing Timetable Using Actor-Critic Reinforcement Learning

Jin Liu*, Ronghui Liu
Institute for Transport Studies
University of Leeds
Leeds, LS2 9JT,United Kingdom
Email: J.Liu12@leeds.ac.uk

***Abstract*— Train scheduling is a crucial part of railway operations, where trains are allocated to particular routes and times to ensure a sustainable utilization of the railway network. Decisions of train scheduling are usually made in well advanced after considering various factors such as expected passenger and freight demand, infrastructure availability, and operational constraints. Nonetheless, scheduling extra train services, being it for passengers or freight, to accommodate unexpected demand in the railway system remains a persistent challenge. This is due to the fact that the existing timetable is typically designed with predetermined robustness, and the introduction of additional trains can adversely impact the robustness of the timetable. To address this issue, this paper presents an approach using an actor-critic reinforcement learning algorithm that takes into account the aspect of timetable robustness in solving the problem of scheduling additional trains. A case study of this method demonstrates that the proposed algorithm can take into account the importance of robustness in the decision-making process, resulting in well-informed decisions and a more reliable timetable.**

Keywords—reinforcement learning, actor-critic reinforcement learning, train scheduling, railway traffic management

I. INTRODUCTION

Scheduling extra train services is a crucial process in railway transportation that involves adding additional trains to an existing timetable to meet ad-hoc demand or respond to unexpected disruptions. The goal of scheduling extra train services is to create an effective and efficient timetable that meets the needs of passengers and the railway system while minimizing costs and maximizing revenue.

In practice, the scheduling process for adding one or more extra trains usually includes three steps (shown in Fig.1): (1) the first step is to identify the optimal timing for the new trains. This step involves analyzing the existing timetable and identifying the best periods to meet the demand or congestion. The new trains must be scheduled at a time that minimizes disruptions to the existing schedule while meeting the demand of passengers. The timing of the new trains must also consider the allocation of infrastructure resources such as track elements, routes, platform, sidings, etc. which ensures the extra services will not lead unfeasibility to existing timetable. (2) the second step is the allocation of rolling stock and human resources. This step involves assigning appropriate resources to the new trains, such as locomotives, coaches, and crew members. The allocation of these resources must be done in such a way that the new service(s) do not interfere with the existing schedule on rolling stock and crew members. The resources must be allocated efficiently to ensure that they are used to their full capacity. (3) the third step is to ensure that the insertion does not adversely affect the overall quality of service of the railway system. The new trains must be scheduled in such a way that they do not cause delays or cancellations to existing trains.

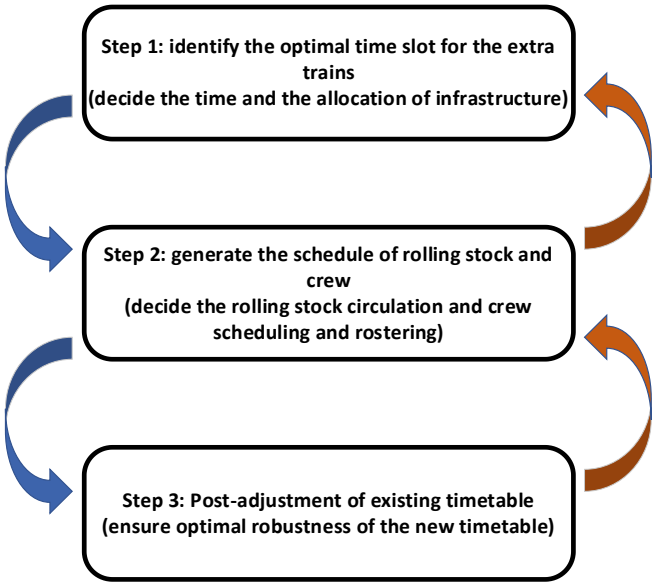

Fig. 1 the scheduling process for adding extra services

The challenge of scheduling additional train services on an existing railway timetable is considered difficult due to the fixed robustness aspect of train timetables, because adding extra trains into an existing timetable can lead to significant degradation of network resilience and makes both the existing and extra services are sensitive to hazards during real-time operations. During the resolution process, constructing a railway timetable in one attempt is also challenging, since it requires negotiation among numerous railway operators, and conflicts may arise in their operation objectives. The final decision may not always be optimal, as the negotiation is limited by the operational scope of each operator and the limited time of this process. Moreover, this scheduling problem is not a typical scheduling or rescheduling problem in terms of railway operation and usually cannot be solved by existing optimization technique for train scheduling; as in practice, post-adjustments to existing train timetables are usually necessary for better timetable robustness once the time slots of the inserted trains are decided, which makes the operational constraints for both existing and additional trains inherently flexible for this whole optimization process, and would need several rounds of optimizations to split, relax, and narrow-down detailed constraints to make the final decision. Finally, timetable robustness is post-measurement, namely, the robustness of newly generated timetable can be assessed only after the timetable is executed, and the timetable resolved

by empirical knowledge is usually not reliable due to the high uncertainties of rail operation circumstance.

The main contribution of this paper is the application of actor-critic reinforcement learning algorithm in solving the train insertion problem. We use the spatiotemporal lateness distribution data from the current timetable to train the critic agent, creating a refined version of the critic agent that undergoes iterative updates during the learning process. As a result, the critical agent can proficiently steer the actor agent in making well-informed decisions pertaining to the introduced train. This process factors in the resilience of the timetable, consequently greatly shorten the decision-making procedure for incorporating additional trains into the timetable.

II. Related works

Extensive efforts have been dedicated to solving the railway scheduling and rescheduling problems. In the subsequent section, we outline previous research that is particularly pertinent to this study. If the readers are interested, the surveys [1][2][3] can provide a comprehensive scope on objective identification and formulation, mathematical modelling and resolution algorithm of the railway scheduling and/or rescheduling problem, and the review paper [4] summarizes taxonomy, regulation, and further application of A.I. in railway

Liu et al. describes a multi-agent framework for solving railway train rescheduling problem, the framework provides a comprehensive architecture for a multi-agent system which can tackle the train rescheduling problem in decentralize way [5]. The study was extended for a railway bottleneck section in and compared with centralized resolution system to show its computational efficiency [6].

Burdett and Kozan proposed novel approach to solve train inserting problem using a hybrid job-shop formulation [7]. The problem in this study considers the competition for railway infrastructure between the inserted train(s) and existing trains in different operators and solve this problem with a meta-heuristic scheduling technique. The time constraints in this study are optional viewed as 'soft' or 'hard' constraints. If the time constraints are viewed as 'soft', these constraints are primarily enforced by penalizing the total time window violations in the objective, otherwise (i.e., the time constrains are 'hard'), timing conditions should be strictly enforced by defining an operation as fixed and then disallowing any time window violation associated with fixed operations to occur.

Cacchiani et al. proposed a novel approach to maximize the utilization of railway infrastructure by scheduling freight trains on passenger lines [8]. The proposed approach involves modeling the problem as an integer linear programming and is solved by using a Lagrangian heuristic based algorithm. The algorithm first identifies the optimal time window for freight trains and then optimizes the timetable by inserting as many freight trains as possible within that window. This approach has the potential to significantly improve the efficiency of railway systems and enhance the overall transportation network.

Jiang et al. presented a novel approach to address the issue of increasing passenger demand by scheduling additional trains [9]. The goal of this approach is to dispatch these trains as quickly as possible to congested areas, where they can alleviate the passenger volume. To achieve this, the authors propose a skip-stop heuristic rule that shortens the journey of these trains by skipping stops where dwelling is not necessary. This rule is employed to solve the problem of scheduling these additional trains efficiently.

Ljunggren et al. developed a framework to tackle path searching problem for train inserting problem [10]. Within the framework, railway network is represented with a graph formulation and the shortest path is searched by a variant of Dijkstra's algorithm. In the test experiments, maximizing timetable robustness is set as the objective function and pre-processing is applied to omit train driver's maximum allowed workload.

III. Problem Description

A general railway network is usually represented by a mixed multi-graph $G = (N, E \cup A)$ in which E and A are the edge and the arc set. Each node $n \in N$ represents a station in the network, each edge $e_{n,n'} \in E$ represents a bi-directional track between station n and n' where trains can travel in both directions, and each arc $a_{n,n'} \in A$ stands for a mono-direction track between station n and n', $n, n' \in N$. Moreover, $k \in E \cup A$.

For the trains considered in this train insertion problem, given that $T = T_{ex} \cup T_{in}$, where T_{ex} is the set of train in existing timetable, and T_{in} is the set of trains to be inserted. For each train $\tau \in T_{ex}$, its operational timetable $\mathbb{t}_\tau$ is usually decided in planning stage and organised with corresponding arrival and departure times in stations, i.e., $\mathbb{t}_\tau = \{t^a_{\tau,n}, t^d_{\tau,n} \dots t^a_{\tau,n'}, t^d_{\tau,n'}\}, \tau \in T_{ex}, n, n' \in N$, where $t^a_{\tau,n}$ and $t^d_{\tau,n}$ stand for the arrival and departure times for train τ at station n, respectively. For the train to be inserted, i.e., $\tau' \in T_{in}$, its timetable shall be generated as , $\mathbb{t}_{\tau'} = \{t^a_{\tau',n}, t^d_{\tau',n} \dots t^a_{\tau',n'}, t^d_{\tau',n'}\}, \tau' \in T_{in},\ n, n' \in N$.

Within this study, the proposed reinforcement learning algorithm shall generate the timetables for inserted trains, $\mathbb{t}_{\tau_j}$, into the existing timetable, $\mathbb{t}_{\tau_i}$. As discussed in section I, the new schedule including all the existing trains and inserted trains shall consider the robustness aspect in practice operational, so the timetable of existing trains might require further adjustment when consider the extra trains. Let the adjustment for timetable of train τ_i is $\mathbb{e}_\tau = \{\epsilon^a_{\tau,n}, \epsilon^d_{\tau,n}, \dots, \epsilon^a_{\tau,n'}, \epsilon^d_{\tau,n'}\},\ \tau \in T_{ex},\ n, n' \in N$, where ϵ is the adjustment value of a single time point. The adjusted timetable for train τ_i in final decision is then derived as $\mathbb{t}_\tau{}' = \mathbb{t}_\tau + \mathbb{e}_\tau$, and thus, the final decided timetable including the inserted trains is $\mathbb{t}' = \mathbb{t}_\tau{}' \cup \mathbb{t}_{\tau'}$.

The train insertion problem requires careful evaluation of the robustness of a given timetable, which encompasses all trains $\mathbb{t}'$. However, as timetable robustness is typically a post-measurement, to overcome this challenge, we will employ a data-driven approach in this study, leveraging historical delay distribution data from the same network to predict the robustness of a newly developed timetable. Here we quantify the robustness of a certain timetable with an indicator between 0 and 1, i.e., $\mathcal{R}(\mathbb{t}',\ H) \in [0,1]$, where H is the historical spatiotemporal delay distribution data for the considered network, the value of $\mathcal{R}(\cdot)$ with respect to the robustness of the considered timetable spans for 0 to 1 where 0 stands for unfeasible timetable and 1 stands for the optimal robustness. Then the train insertion problem can be formulated as follows:

Objectives:

$$\textbf{Max.}\ O_1 = \mathcal{R}(\mathbb{t}',\ H) \tag{1}$$

$$\textbf{Min.}\ O_2 = \sum_{\tau \in T_{ex}} |\mathbb{e}_\tau| \tag{2}$$

Constraints:

$$t^a_{\tau,n'} \geq t^d_{\tau,n} + t_\tau(k_{n,n'}) \tag{3}$$

$$t^d_{\tau,n} \geq t^a_{\tau,n} + t^p_{\tau,n} \tag{4}$$

$$t^a_{\tau',n} + M \cdot (1 - i^n_{\tau,\tau'}) \geq t^a_{\tau,n} + \delta^n(\tau,\tau') \tag{5}$$

$$t^a_{\tau,n} + M \cdot i^n_{\tau,\tau'} \geq t^a_{\tau',n} + \delta^n(\tau,\tau') \tag{6}$$

$$t^d_{\tau',n} + M \cdot (1 - i^n_{\tau,\tau'}) \geq t^d_{\tau,n} + \delta^n(\tau,\tau') \tag{7}$$

$$t^d_{\tau,n} + M \cdot i^n_{\tau,\tau'} \geq t^d_{\tau',n} + \delta^n(\tau,\tau') \tag{8}$$

$$i^n_{\tau,\tau'} = \begin{cases} 1, & \text{if } \tau \text{ prior to } \tau' \text{ at station } n \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Within this study, the objective function O_1 (Equation (1)) shall maximise the robustness of the final developed timetable and the objective function O_2 (Equation (2)) shall minimize the post adjustment of timetable of existing trains. So the algorithm shall trade-off between the two objective functions in the optimisation process. The constraints of the optimisation is discussed in Equation (3) to (9). Equation (3) indicates the operational constraints for train τ between station n and n', i.e., the travelling for train τ from station n to n' should not be small than the estimated constant traveling time $t_\tau(k_{n,n'})$, where $k_{n,n'} \in E \cup A$. Equation (4) is the dwelling constraint for train τ at station n, where $t^p_{\tau,n}$ is the minimum dwelling time for train τ at station n. Equation (5)-(8) are the headway constraints for two neighbouring train τ and τ'at station n, these constraints are organised using big-M approach and $i^n_{\tau,\tau'}$ is a binary indicator which indicates the passing sequence of arrival or departure actions of train τ and τ'. $\delta^n(\tau,\tau')$indicates the headway between train τ and τ'at station n.

IV. Actor-Critic Reinforment Learning Algorithm

Actor-critic is a new branch reinforcement learning algorithm that combines elements of both policy-based and value-based methods. In actor-critic, there are two components: the actor agent and the critic agent.

- The actor agent is responsible for selecting actions to take in the environment based on the current state. It learns a policy that maps states to actions. This is done using a stochastic policy gradient method, such as Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO) algorithm, etc.
- The critic agent, on the other hand, is responsible for estimating the value function of the current state, which represents the expected cumulative reward that can be obtained from that state. The critic uses the temporal difference (TD) learning method to estimate the value function, by computing the difference between the predicted value of the next state and the actual observed value.

The actor and critic agents are usually approximated by neural networks and trained synchronously, with the critic providing feedback to the actor about the quality of its actions. This feedback is used to update the policy function of agent in a way that improves the expected cumulative reward. Fig. 2 provides an overview of the architecture of the proposed actor critic reinforcement learning.

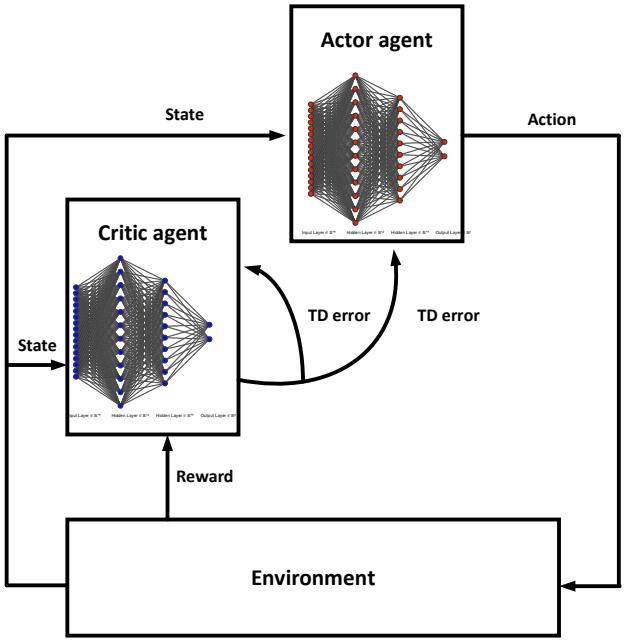

Fig. 2 The framework of Actor-critic reinforcement learning algorithm and the data flow within it

A. Actor agent

In this study, we propose an actor-agent that is trained to create a timetable for a train by learning how to map a promising action to the observed state of the train through its policy function, $\pi: s \to \alpha$. The algorithm takes into account the extra trains and first determines the timetables of the inserted trains before adjusting the schedules of the existing trains. The state of a train is abstracted at every station where the inserted train departs, dwells, and terminates. Let the train to be inserted is train τ_*The state of a train is represented as a tuple consisting of (1) current time, which indicating the peak or off-peak hour that the inserted train will be operated. (2) the arrival/departure time of neighbouring trains at the current time (including those that arrive/depart prior to and after the current time) which indicating the potential time slot for the inserted train. (3) minimum dwelling period of the inserted train, i.e., $s = \{t, t^a_{\tau,n}, t^d_{\tau,n}, t^a_{\tau',n}, t^d_{\tau',n}, t^p_*\}, s \in S$, where s stands for the state vector and S is the set of states. The action of the actor agent includes the candidate arrival and departure times of the train to be inserted at the considered station, i.e., $\alpha_n = \{t^a_{\tau_*,n}, t^d_{\tau_*,n}\}, \alpha_n \in \mathcal{A}$ where α_n is the action at station n and $\mathcal{A}$ is the set of actions.

The existing train schedule, which includes the arrival and departure times at all stations under consideration, is regarded as the environment in which the reinforcement learning algorithm operates. The algorithm interacts with the

environment through the actor agent to determine the timetable for an additional train, from its starting station to its destination. Once the actor agent has made decisions for all necessary extra trains, the current episode will be concluded, and the trajectory of the actor agent is formulated as $\varphi = \{(s_{n_1}, \alpha_{n_1}), (s_{n_2}, \alpha_{n_2}), \dots (s_N, \alpha_N)\}$.

In this study, we use policy gradient approach to update the neural network within actor agent, which is identified by parameters θ. The reward is defined to quantify the performance of the action, α_n, made by actor agent at station n with respect to the objective function O_1, and $r(n)$ (i.e., the value of reward at the station n) is decided as the same value derived by the objective function O_1. The total rewards through φ can be derived as:

$$R(\varphi) = \sum_{\varphi} \gamma \cdot r(n) \tag{10}$$

where $\gamma \in (0,1]$ is a discount factor. Because the reward for the actor agent is supported by the critic agent rather than the 'true' reward identified as, $r(n)$, finally evaluated by the algorithm (here, we call the reward provided by critic agent as baseline reward, identified as $b(n)$), so the distinction between the $R(\varphi)$ and $B(\varphi)$ will be viewed as temporal difference error which will be used to update the neural network in actor agent, then the objective function can be approximated as:

$$\begin{aligned} O_1'(\theta) &= \boldsymbol{E}_{\varphi\sim\pi} R(\varphi) \\ &= \sum_{\varphi} P(\varphi;\theta) \\ &\cdot \sum_{n}^{N} \big(r(n) - b(n)\big) \end{aligned} \tag{11}$$

and then the gradient of the expected rewards can be derived as:

$$\begin{aligned} \nabla_\theta O_1'(\theta) &= \sum_{\varphi} \nabla_\theta P(\varphi;\theta) R(\varphi) \\ &= \boldsymbol{E}_{\varphi\sim\pi} \nabla_\theta \log P(\varphi;\theta) \\ &\cdot \sum_{n}^{N} (r(n) - b(n)) \end{aligned} \tag{12}$$

At the end of an episode, the parameters θ will be updated as:

$$\theta \leftarrow \theta + l \cdot \nabla_\theta O_1'(\theta) \tag{13}$$

where l is the learning rate, i.e., $l \in (0,1]$.

B. Critic agent

In actor-critic reinforcement learning algorithms, the critic agent is responsible for estimating the value function for a given policy. The value function represents the expected return (i.e., cumulative reward) that an agent would receive starting from a particular state, and following a particular policy thereafter. In this study, the critic agent is organized to benchmark and quantify the robustness of a timetable. Usually, evaluating a railway timetable is typically a challenging task due to the fact that its robustness can only be quantified post-execution through the accumulation of operational data covering various hazards that may occur in the railway network.

The critic agent is approximated by a neural network parameterized by ω, which aims to determine the robustness of a newly inserted timetable against the existing ones at a single station. The neural network takes several inputs, including (1) the ID of the station being considered, n, (2) the timetable of the inserted train generated by the actor agent, $\{t^a_{\tau_*,n}, t^d_{\tau_*,n}\}$, and the timetables of the trains that run prior to and after the inserted train at the same station $\{t^a_{\tau_i,n}, t^d_{\tau_i,n}, t^a_{\tau_j,n}, t^d_{\tau_j,n}\}$. Here, τ_i and τ_j refer to the trains that pass before and after the train τ_*, respectively, at station n. Moreover, the output is the quantified robustness (i.e., a value between 0 and 1) for the considered timetable $\{t^a_{\tau_*,n}, t^d_{\tau_*,n}\}$.

The training of the critic agent is structured into two phases. The first phase involves off-line training using previously accumulated data in the network, while the second phase involves on-line training through iterative interactions within reinforcement learning. During the first phase, the critic agent acquires a basic understanding of the system's robustness based on historical data, but it is important to note that relying solely on past knowledge may not be sufficient to support decision-making processes for new situations. Therefore, in phase 2, the critic agent further develops its knowledge to accumulate information that can help it benchmark the robustness for new timetables across the whole network.

Fig.3 Train delay distribution over spatiotemporal information of the network

During phase 1, the previously collected data is organized into mock data in the same way and fed into the neural network. If the considered train is the first scheduled train, the timetable of the train prior is considered as $\{0,0\}$. In this study, the robustness value, which represents the expected output of the neural network, is evaluated using a percentage value that is quantified by the probability of neighboring trains being affected by the historical delay distribution of the considered train. Fig. 3 provides an example of train delay distribution over spatiotemporal information of rail network. Considering two neighboring train, τ_i and τ_j, passing the same station n_i, and their delay distributions at the station over time are $P_{\tau_i}(t)$ and $P_{\tau_j}(t)$ which are approximated by (reversed) exponential distribution, respectively. Let the considered time t' is the target time stamp we are trying to insert the train, then the robustness for t' is quantified by the minimum value of the possibility value from $t^d_{\tau_i,n}$ to t' and t' to $t^a_{\tau_j,n}$ which is

identified in the lower part of equation (14). Furthermore, if the time t' equals either $t^d_{\tau_i,n_i}$ or $t^a_{\tau_j,n_i}$. The time should be infeasible, and thus, the value of robustness equals to zero.

$$\mathcal{R}([t^d_{\tau_i,n}, t', t^a_{\tau_j,n}], H) = \begin{cases} \min_{t'}(P_{\tau_i}(t), P_{\tau_j}(t)), & t^d_{\tau_i,n} < t' < t^a_{\tau_j,n} \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

In Phase 2, the critic agent will try to further develop this knowledge on benchmarking robustness on a new timetable, and quantifies a temporal difference error to update the actor agent. At the end of each episode, the parameters of critic neural network will be updated as

$$\omega \leftarrow \omega + l \cdot \sum_n^N (r(n) - b(n)) \quad (15)$$

V. CASE STUDY

The proposed algorithm is tested using a British rail corridor covering from Newcastle to London Kings Cross. The rail corridor covers 7 main stations from northern England to central London area. Fig. 4 provides an overview of the corridor.

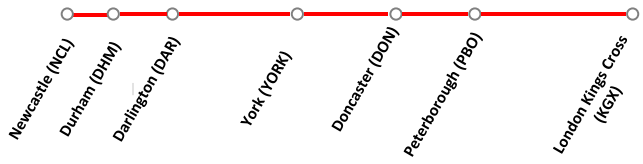

Fig. 4 Overview of network

The original timetable of this case study considered the schedule from 9:00 to 16:00, and one extra freight trains are expected to be inserted into the schedule. Both the actor and critic agents are approximate using two fully connected neural networks which are setup with the following parameters. In proposed actor-critic reinforcement learning algorithm is implemented Python (v3.10) on a windows PC with AMD Ryzen 7 4800U processor and 24GB RAM, TensorFlow is called to organize the neural networks in the reinforcement learning algorithm.

Table I. structure of actor and critic neural network

The Nth Layer	**No. of nodes**	**Activation function**
Critic Neural Network		
1st Normalized Input Layer	6	ReLU (rectified linear unit)
2nd Dense layer dims	256	ReLU
3rd Dense layer dims	256	ReLU
4th Dense layer dims	128	ReLU
5th Dense layer dims	64	ReLU
6th Dense layer dims	32	ReLU
7th Dense layer dims	1	Linear
Actor Neural Network		
1st Normalized Input layer	6	ReLU
2nd Dense layer dims	128	ReLU
3rd Dense layer dims	128	ReLU
4th Dense layer dims	64	ReLU
5th Dense layer dims	32	ReLU
6th Dense layer dims	1	tanh

Table II. Parameter settings of actor-critic reinforcement learning

Parameters	**Value**
Replay buffer size	50
Batch size	64
Delay episode for training	512
Update step	64
Discount factor	0.9
Actor learning rate	0.003
Critic learning rate	0.001

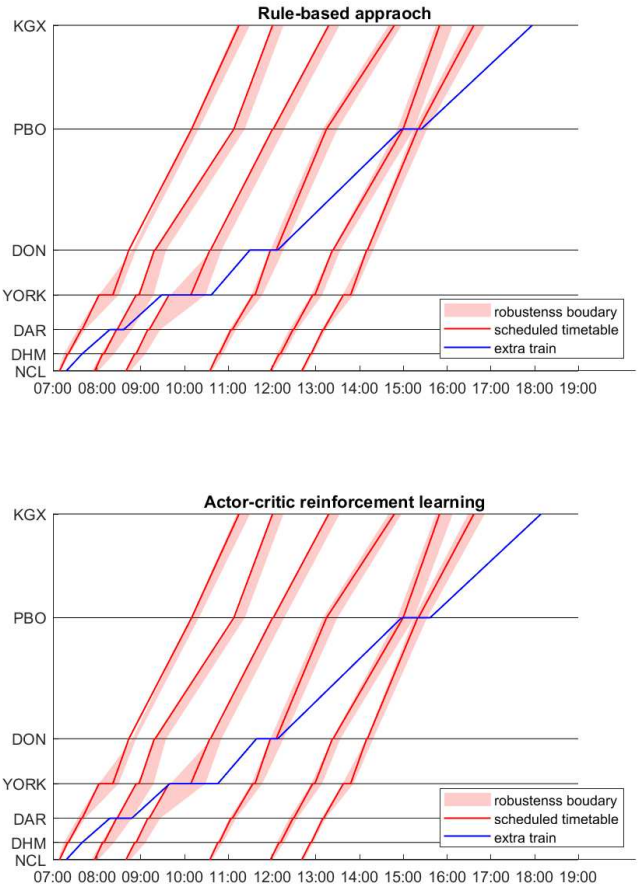

Fig. 5. The resultant timetable for the extra train

In this study, we use longest headway dispatching rule, a rule-based approach, as the benchmark. The rule-based approach will schedule the extra train only with the support of existing timetables, and the arrival and departure times of the inserted train will be selected ensuring equally longest time interval between the inserted train and its neighbouring trains

Table III Timetable for the extra train generated by rule-based algorithm and the reinforcement learning algorithm

Stations	NCL	DHM	DAR	YORK	DON	PBO	KGX
Rule-based algorithm							
Arrival	7:18	7:40	8:18	9:29	11:30	14:58	17:57
Departure	7:18	7:40	8:37	10:37	12:08	15:25	17:57
Actor-critic reinforcement learning algorithm							
Arrival	7:18	7:40	8:18	9:40	11:39	14:58	18:09
Departure	7:18	7:40	8:48	10:46	12:08	15:37	18:09

at stations. Fig. 5 provides the scheduling results generated by the benchmark approach and the proposed actor-critic reinforcement learning algorithm, and Table III illustrates the detailed timetable generated by both algorithms.

In Fig. 5, the red lines represent the timetables of existing trains, while the red polygons describe the robustness boundaries of each train within the station area. These robustness boundaries are generated based on historical lateness data of the trains. It is worth noting that there is significant lateness of trains at Darlington and York stations during peak hours (7:00 - 9:30 a.m.). Therefore, all trains passing through York station around peak hours shall incorporate an extra time interval in their headway to ensure the robustness of the existing trains' timetables. However, when observing the results generated by the rule-based approach (upper part of Fig. 5), it is evident that the timetable of the extra train has a slight overlap with the robustness boundary of the second train at Darlington station. This overlap has the potential to compromise the timetable robustness of the second train and increase the likelihood of readjusting its timetable during real-time operation. In contrast, the timetable generated by the reinforcement learning algorithm for the extra train (lower part of Fig. 5) takes into account the fluctuation of the departure time of the second train. As a result, the algorithm makes a decision to slightly postpone the departure time of the inserted train at Darlington station. Similar decisions are also made for the departure times at York station and Peterborough station. By considering the potential conflicts with existing train timetables and making appropriate adjustments, the reinforcement learning algorithm demonstrates its ability to improve robustness and minimize the need for real-time readjustments in train operations.

VI. Conclusions

This paper presents a novel actor-critic reinforcement learning algorithm to address the challenge of scheduling additional trains into existing timetables. The main difficulty in this train insertion problem lies in the unknown robustness aspect, which is a post-measurement during railway operations and challenging to incorporate during the planning stage. Within the actor-critic reinforcement learning algorithm, the critic agent uses historical lateness data to measure timetable robustness for the considered network. This information is then used to provide the reward to the actor agent, enabling better decision-making and ensuring improved robustness. We illustrate the effectiveness of our approach through a case study, showcasing a successful pilot test of the proposed reinforcement learning algorithm and its promising capabilities in solving practical train insertion problems in railway corridors.

References

[1] V. Cacchiani *et al.*, "An overview of recovery models and algorithms for real-time railway rescheduling," *Transportation Research Part B: Methodological*, vol. 63. Elsevier Ltd, pp. 15–37, May 01, 2014. doi: 10.1016/j.trb.2014.01.009.

[2] F. Corman and L. Meng, "A review of online dynamic models and algorithms for railway traffic management," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1274–1284, 2015, doi: 10.1109/TITS.2014.2358392.

[3] W. Fang, S. Yang, and X. Yao, "A Survey on Problem Models and Solution Approaches to Rescheduling in Railway Networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 2997–3016, 2015, doi: 10.1109/TITS.2015.2446985.

[4] N. Bešinović *et al.*, "Artificial Intelligence in Railway Transport: Taxonomy, Regulations and Applications," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–14, 2021, doi: 10.1109/TITS.2021.3131637.

[5] J. Liu, L. Chen, C. Roberts, Z. Li, and T. Wen, "A Multi-agent Based Approach for Railway Traffic Management Problems," in *2018 International Conference on Intelligent Rail Transportation, ICIRT 2018*, 2019, pp. 1–5. doi: 10.1109/ICIRT.2018.8641621.

[6] J. Liu, L. Chen, C. Roberts, G. Nicholson, and B. Ai, "Algorithm and peer-to-peer negotiation strategies for train dispatching problems in railway bottleneck sections," *IET Intell. Transp. Syst.*, vol. 13, no. 11, pp. 1717–1725, 2019, doi: 10.1049/iet-its.2019.0020.

[7] R. L. Burdett and E. Kozan, "Techniques for inserting additional trains into existing timetables," *Transp. Res. Part B Methodol.*, vol. 43, no. 8–9, pp. 821–836, Sep. 2009, doi: 10.1016/j.trb.2009.02.005.

[8] V. Cacchiani, A. Caprara, and P. Toth, "Scheduling extra freight trains on railway networks," *Transp. Res. Part B Methodol.*, vol. 44, no. 2, pp. 215–231, 2010, doi: 10.1016/j.trb.2009.07.007.

[9] F. Jiang, V. Cacchiani, and P. Toth, "Train timetabling by skip-stop planning in highly congested lines," *Transp. Res. Part B Methodol.*, vol. 104, pp. 149–174, Oct. 2017, doi: 10.1016/j.trb.2017.06.018.

[10] F. Ljunggren, K. Persson, A. Peterson, and C. Schmidt, "Railway timetabling: a maximum bottleneck path algorithm for finding an additional train path," *Public Transp.*, vol. 13, no. 3, pp. 597–623, 2021, doi: 10.1007/s12469-020-00253-x.