2021 IEEE Intelligent Transportation Systems Conference (ITSC)
Indianapolis, USA. September 19-21, 2021

A Policy-based Reinforcement Learning Approach for High-speed Railway Timetable Rescheduling

Yin Wang, Yisheng Lv†, Jianying Zhou†, Zhiming Yuan, Qi Zhang, Min Zhou

Abstract— **In the daily management of high-speed railway systems, the train timetable rescheduling problem with unpredictable disturbances is a challenging task. The large number of stations and trains leads to a long-time consumption to solve the rescheduling problem, making it difficult to meet the real-time requirements in real-world railway networks. This paper proposes a policy-based reinforcement learning approach to address the high-speed railway timetable rescheduling problem, in which the agent minimizes the total delay by adjusting the departure sequence of all trains along the railway line. A two-stage Markov Decision Process model is established to model the environment where states, actions, and reward functions are designed. The proposed method contains an offline learning process and an online application process, which can give the optimal rescheduling schedule based on the current state immediately. Numerical experiments are performed over two different delay scenarios on the Beijing-Shanghai high-speed railway line. The simulation results show that our approach can find a high-quality rescheduling strategy within one second, which is superior to the First-Come-First-Served (FCFS) and First-Scheduled-First-Served (FSFS) methods.**

I. Introduction

High-speed railways play an important role in the Chinese comprehensive transportation systems. As of June 2019, the total length of high-speed railways has reached $31,000$ kilometers, accounting for more than $2/3$ of the total mileage in the world and covering major cities in China.

During the daily railway operation, the train usually runs according to the prescribed timetable that schedules the train's arrival and departure time along the railway line [1]. However, due to unexpected disturbances such as infrastructure failures and natural disasters, severe railway traffic delays can happen, and the train timetable rescheduling (TTR) is needed. When rescheduling is required, the train dispatcher needs to make a decision quickly based on the specific situation. Instead of checking all feasible solutions, they can only adjust schedules depending on experience [2]. Four strategies are often used to address TTR problems in a short period: rerouting, retiming, reordering, and canceling trains.

As an NP-hard problem, the TTR problem has been extensively studied in recent years [3], [4]. Corman et al. [5] incorporated rescheduling algorithms and rerouting strategies in a tabu search scheme to calculate train schedules, which improved the computation speed and solution accuracy. In [6], Dündar and Şahin introduced genetic algorithms for conflict resolutions, which could find the optimal solutions for small-sized problems. D'Ariano et al. [7] modeled the TTR problem with a graph formulation and designed a branch and bound algorithm to find an optimal or near-optimal timetable within a short time. Ning [8] presented the parallel rail transportation systems, which led to a new paradigm of intelligent management, operation, and services of rail transportation systems.

Inspired by the successful applications in Atari games [9], deep reinforcement learning was introduced to the TTR problem. Šemrov et al. [10] adopted the Q-learning algorithm to train rescheduling on a single-track line. In [11], an approach for scheduling bidirectional railway lines was proposed based on Q-learning to minimize the total priority-weighted delay. Based on it, Zhu et al. [12] showed that the rescheduling solution was affected by the state representation of the railway environment. Obara et al. [13] simulated the rescheduling problem with graph theory and used the DQN method to handle small-scale networks. In [14], the DQN approach that changed the departure sequence of all trains was applied to the TTR problem, which modeled the state as the arrival and departure time matrix.

In this paper, a novel policy-based reinforcement learning approach is proposed to solve the TTR problem under unexpected disturbances. The main contributions of this paper are the following:

- We propose a two-stage Markov Decision Process (MDP) model for the TTR problem, based on which we establish the environment and elements of reinforcement learning.
- Considering the various constraints and conflicts in the TTR problem, we design a policy-based reinforcement learning algorithm, and develop an offline learning process and an online application process to solve the problem. Numerical experiments show that the proposed

This work was supported partially by the National Natural Science Foundation of China under Grants 61790573 and 61790575; the Center of National Railway Intelligent Transportation System Engineering and Technology (Contract No.RITS2019KF03), China Academy of Railway Sciences Corporation Limited; China Railway Project N2019G020.

Yin Wang and Yisheng Lv are with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China. They are also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, 100049, China. Yin Wang is also with the Center of National Railway Intelligent Transportation System Engineering and Technology, China Academy of Railway Sciences Corporation Limited 100081, China.

Jianying Zhou is with the School of Information Management, Shanghai Lixin University of Accounting and Finance, Shanghai 201620, China.

Zhiming Yuan and Qi Zhang are with the Center of National Railway Intelligent Transportation System Engineering and Technology, China Academy of Railway Sciences Corporation Limited 100081, China.

Min Zhou is with the State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, 100044, China.

†Corresponding Author. E-mail: yisheng.lv@ia.ac.cn, jianyingzhou@126.com

2021 IEEE International Intelligent Transportation Systems Conference (ITSC) | 978-1-7281-9142-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITSC48978.2021.9564980

approach can quickly find a high-quality solution compared to the First-Come-First-Served (FCFS) and First-Scheduled-First-Served (FSFS) methods.

The remainder of this paper is organized as follows. Section II introduces the high-speed railway model, time constraints, and conflicts. In Section III, the two-stage Markov Decision Process (MDP) model is presented. In Section IV, an algorithm for the TTR problem is introduced, and numerical experiments based on that are analyzed in Section V. Section VI summarizes this paper and discusses the future work.

II. Preliminary Of A High-speed Railway Model

Given one direction of a high-speed railway line, which consists of M stations, $M-1$ sections, and N trains, as displayed in Fig. 1, the original schedule provides the arrival and departure time of trains at each station. There are two main forms of delays: the departure delay at the current station and the arrival delay at the next station. The conflict caused by the delay may occur in any station or section.

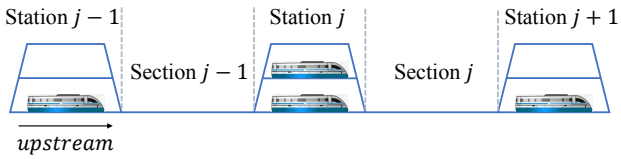

Fig. 1: Description of the high-speed railway line.

After giving an initial delay, the TTR problem is to adjust the arrival and departure timetables of all subsequent trains and generate an alternative schedule. In order to ensure operation safety and service quality, it is necessary to subject to some time constraints during operation. In this paper, our goal is to reduce the total delay of all trains along the railway line and return to the scheduled timetable.

A. Time Constraints in TTR

- When modifying the timetable, the arrival and departure time of the train at the station cannot be earlier than the scheduled time.

$$\begin{cases} A_{ij} \geq A_{ij}^{p}, 1 \leq i \leq N, 1 \leq j \leq M \\ D_{ij} \geq D_{ij}^{p}, 1 \leq i \leq N, 1 \leq j \leq M \end{cases} \tag{1}$$

- Considering the operation safety, adjacent trains need to meet the minimum arrival and departure interval time at the same station.

$$\begin{cases} A_{ij} - A_{i-1j} \geq I_a, 2 \leq i \leq N, 1 \leq j \leq M \\ D_{ij} - D_{i-1j} \geq I_d, 2 \leq i \leq N, 1 \leq j \leq M \end{cases} \tag{2}$$

- The dwell time of the train must satisfy the minimum waiting time to ensure the necessary service at the station.

$$D_{ij} - A_{ij} \geq I_s, 1 \leq i \leq N, 1 \leq j \leq M \tag{3}$$

TABLE I: Definition of symbols

Notation	Description
N	Number of trains
M	Number of stations
A_{ij}	The actual arrival time of the train i at the station j
A_{ij}^{p}	The planned arrival time of the train i at the station j
D_{ij}	The actual departure time of the train i at the station j
D_{ij}^{p}	The planned departure time of the train i at the station j
I_a	The minimum arrival time interval at the same station
I_d	The minimum departure time interval at the same station
I_s	The minimum dwelling time for train service at the station
I_t	The minimum tracking interval time for train at the section
I_q	The reserved buffer time for delay
C^j	Number of platform tracks at the station j

- Taking into account the safety at the section, successive trains are not allowed to cross and must keep the minimum tracking time interval.

$$if\ D_{ij} - D_{i-1j} \geq I_t\ then\ A_{ij+1} - A_{i-1j+1} \geq I_t$$
$$2 \leq i \leq N, 1 \leq j \leq M-1 \tag{4}$$

- The number of tracks at the station is different, which means that the current station receiving and starting trains cannot exceed the station capacity C^j.

For a better understanding of the paper, the parameters are introduced in Table I.

B. Conflict Detection and Solution

Fig. 2 shows the detection and solution process of conflicts caused by delays, where the blue and red lines express adjacent trains in operation. The red dashed line represents the original train plan, the circle represents the time constraints conflicts, and the red solid line means the resolution strategy of conflicts. In this sense, the conflict at sections and stations should be detected and solved firstly before adjusting a feasible schedule.

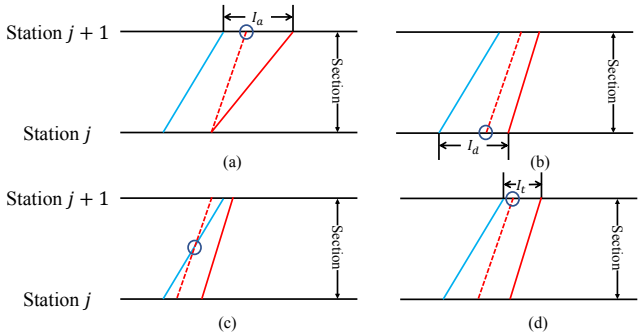

Fig. 2: Solution to the conflicts. (a) and (b) do not meet the arrival and departure time constraints. (c) is the overtaking conflict at the section. (d) does not meet the minimum tracking time interval.

III. Two-stage Markov Decision Process Model For Train Timetable Rescheduling

This section introduces the two-stage Markov Decision Process (MDP) model for the TTR problem and defines the elements of reinforcement learning.

A. Two-stage Markov Decision Process Model

Reinforcement learning is based on the Markov Decision Process (MDP), which can be abstracted as a four-element tuple $\langle S, A, P, R\rangle$, where S is state set, A is action set, P is transition probability, and R is reward function. The process of timetable rescheduling can be regarded as an MDP. For a railway operating plan, the MDP can be intuitively defined: At the current station j, the agent generates the departure sequence for all trains according to the current state s_j which is defined as the arrival time of all trains passing by the current station, and the environment transfers to the next station $j+1$ after executing the action a_j, then acquiring the next state s_{j+1}.

The action $a_j \in \mathbb{R}^n$ is an N-dimension (N-D) variable whose dimension depends on the number of trains at the station j. Inspired by the continuous action discretization of reinforcement learning [15], we propose a two-stage Markov Decision Process model to solve high-dimensional constrained action space.

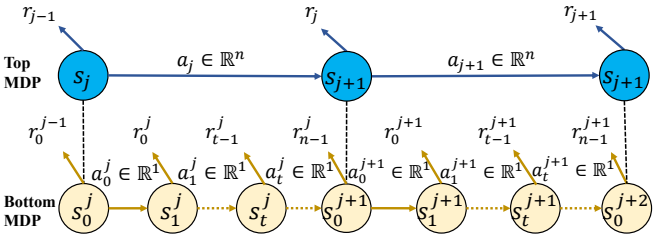

Fig. 3: Two-stage Markov Decision Process (MDP) model.

As shown in the Fig. 3, we construct a two-stage MDP model. The agent takes the N-D action $a_j \in \mathbb{R}^n$ under the state s_j in the top MDP. We introduce a new sequential decision model usually used in the recommendation system [16] in the bottom MDP, which shares the same environment with the top MDP. We decompose the original MDP model with N-D action into a series of 1-D actions. Among them, s_t^j represents the environment state in the t-th step at the current station j, a_t^j is the 1-D action made by the agent, and r_t^j is the immediate reward. The bottom MDP model is described in detail as follows.

B. States in the Bottom MDP

State is a representation of the observation which describes the environment. Taking the current station j as an example, each train x_i^j that needs to be adjusted is abstracted into a tuple consisting of original planned arrival time, original planned departure time, and actual arrival time, expressed as $x_i^j = \left[A_{ij}^p, D_{ij}^p, A_{ij}\right]$. The state s_t^j in the t-th step can be defined as:

$$s_t^j = \left\{x_0^j, x_1^j, \cdots, x_i^j, \cdots\right\} \tag{5}$$

where s_t^j is a set of the remaining trains for adjustment in the t-th step at the current station j. It is intuitive that for $t = \{0, 1, \cdots, n-1\}$, when $t = 0$, $s_0^j = \left\{x_0^j, x_1^j, \cdots, x_i^j, \cdots, x_{n-1}^j\right\}$ means the sets of all trains passing through the station j and when $t = n$, $s_n^j = \phi$ means that all trains have been adjusted at the station j.

C. Actions in the Bottom MDP

In each step of the bottom MDP model, the agent selects an action from all possible actions to interact with the environment. When at the current station j, action a_t^j means that the agent chooses a train that departs at the t-th position according to the current strategy π. Limited by the number of platform tracks, the discrete set of possible actions denoted as A^j, where $A^j = \left[0, C^j - 1\right]$. We define the strategy π as a $Softmax$ strategy, and the probability of selecting the action a_t^j is calculated as:

$$\pi_\theta\left(a_t^j | s_t^j\right) = \frac{\exp\left\{\mu_\theta\left(x_{a_t^j}^j\right)\right\}}{\sum_{a^j \in A^j} \exp\left\{\mu_\theta\left(x_{a^j}^j\right)\right\}}, a_t^j \in A^j, x_{a_t^j}^j \in s_t^j \tag{6}$$

where $\mu_\theta(\cdot)$ represents the strategy network and $x_{a_t^j}^j$ describes the tuple of train previously defined corresponding to action a_t^j.

D. Transition in the Bottom MDP

The transition function $T : S \times A \to S$ means that the agent transfers from the current state s_t^j to the next state s_{t+1}^j after selecting an action a_t^j. In the bottom MDP model, the action a_t^j indicates that the agent decides the train which departs at the t-th position, so the transition function is defined as removing the train $x_{a_t^j}^j$ from the current state s_t^j.

$$\begin{aligned} s_{t+1}^j &= T\left(s_t^j, a_t^j\right) \\ &= \left\{x_0^j, x_1^j, \cdots, x_i^j, \cdots\right\} \setminus x_{a_t^j}^j \end{aligned} \tag{7}$$

E. Reward Function in the Bottom MDP

The reward function plays a critical role in reinforcement learning. It determines the update direction of the agent. In this article, we design two immediate reward functions as follows:

$$\begin{aligned} r_t^j &= R\left(s_t^j, a_t^j\right) \\ &= -\left(A_{ij+1} - A_{ij+1}^P\right) \end{aligned} \tag{8}$$

$$\begin{aligned} r_t^j &= R\left(s_t^j, a_t^j\right) \\ &= -\left[\left(A_{ij} - A_{ij}^p\right) + \left(D_{ij} - D_{ij}^p\right)\right] \end{aligned} \tag{9}$$

There are two options for the objective function: the negative arrival delay time at the next station in (8) or the negative arrival and departure delay time at the current station in (9). Obviously, the larger the delay of trains indicates smaller reward. Note that in our two-stage MDP model, the sum of rewards in the bottom MDP is equal to the reward in the top MDP, which ensures the consistency of the model.

Algorithm 1 Train rescheduling in the bottom MDP model

Input: The strategy network with parameter θ, original state $s_0^j = \left\{x_0^j, x_1^j, \cdots, x_i^j, \cdots, x_{n-1}^j\right\}$, replay buffer B.
output: The scheduling trajectory B.

$T \leftarrow \left|s_0^j\right|$
for $t = 0$ **to** $T-1$ **do**
  for all feasible $a^j \in A^j$ **do**
    Calculate $\pi_\theta\left(a^j|s_t^j\right)$ in (6).
  end for
  Sample an action a_t^j according to $\pi_\theta\left(a^j|s_t^j\right)$.
  Conflict detection and resolution.
  $r_t^j \leftarrow R\left(s_t^j, a_t^j\right)$ in (8) or (9).
  Store tuple $\left(s_t^j, a_t^j, r_t^j\right)$ in B.
  $s_{t+1}^j \leftarrow P\left(s_t^j, a_t^j\right)$ in (7).
end for
return The scheduling trajectory B

IV. Algorithms for TTR Problem

This section presents the algorithm for the TTR problem and the training method based on reinforcement learning.

A. Parameters Learning with Policy Gradient

We take the Monte-Carlo sampling method to get the complete episode. Algorithm 1 shows the sequential decision process in the bottom MDP model at an arbitrary station.

By repeating the train rescheduling process until the terminal station, we obtain a scheduling trajectory based on the two-stage MDP model. We use the policy gradient algorithm in reinforcement learning to update the strategy network parameters. Among them, the high-speed train dispatching trajectory τ can be expressed as:

$$\tau = \left(s_0^0, a_0^0, r_0^0, \cdots, s_t^j, a_t^j, r_t^j, \cdots\right) \tag{10}$$

For the sake of illustration, the trajectory τ is represented as follows:

$$\tau = (s_0, a_0, r_0, \cdots, s_i, a_i, r_i, \cdots, s_{M-1}, a_{M-1}, r_{M-1}) \tag{11}$$

where M is the length of the trajectory.

At each discrete step $t = \{0, 1, \cdots, M-1\}$ in the trajectory τ, the long-term discount return G_t starting from step t is defined as:

$$G_t = \sum\nolimits_{k=t}^{M-1} \gamma^{k-t} r_k \tag{12}$$

where γ is called discount factor, which is generally $(0, 1]$.

According to the Monte-Carlo gradient ascent algorithm, our goal is to maximize the expected long-term return in the trajectory. Under the discrete control strategy, the expected long-term return is approximated by the average cumulative return of the N scheduling strategy trajectories.

$$J(\theta) \approx \frac{1}{N} \sum\nolimits_{n=1}^{N} R(\tau_n) P(\tau_n|\theta) \tag{13}$$

Algorithm 2 The training method for the TTR problem

Input: The strategy network with parameter θ, original train operation schedule with N trains and G stations, discount factor γ, learning rate α, total training episode E, update interval I.
output: The trained model with parameter θ.

for episode $e = 0$ **to** $E-1$ **do**
  Randomly initialize type of delay information for train n at station g.
  for station $j = g$ **to** G **do**
    for train $i = n$ **to** N **do**
      Check for conflicts and get delayed trains and associated delayed trains.
    end for
    Receive state $s_0^j = \left\{x_0^j, x_1^j, \cdots, x_i^j, \cdots, x_{n-1}^j\right\}$.
    Replay buffer $B \leftarrow$ Train rescheduling in the bottom MDP model$\left(\theta, s_0^j, B\right)$ {Algorithm 1}.
  end for
  if epoch e *mod* update_interval $I = 0$ **then**
    $T \leftarrow |B|$
    for $t = 0$ **to** $T-1$ **do**
      Calculate long-term return G_t in (11).
    end for
    Update parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ in (13).
    Empty replay buffer B.
  end if
end for
return The trained model with parameter θ

where $R(\tau_n)$ means the total cumulative return in the trajectory and $P(\tau_n|\theta)$ denotes the probability of trajectory.

We use the long-term discount return G_t at the discrete step t instead of the total cumulative return $R(\tau_n)$ in the trajectory. The gradient $\nabla_\theta J(\theta)$ can be calculated as follows.

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum\nolimits_{n=1}^{N} \sum\nolimits_{t=0}^{M-1} G_t \nabla \log \pi_\theta(a_t|s_t) \tag{14}$$

Obviously, the network parameters are updated in a direction conducive to selecting actions with high rewards.

B. Offline Learning and Online Application

In the offline learning phase, we conduct training on a complete train operation schedule, and the training method for the TTR problem is shown in the Algorithm 2. During each iteration, the initial delay information is randomly set, which mainly consists of the following four random variables: the delayed train number, the delayed station number, the specific delay type, and the delay time. Our method mainly solves the adjustable disturbances, so the delay time is randomly initialized to an integer from 10 to 50 minutes in each episode, which allows the agent to learn experience in a continually changing environment.

In the online application phase, considering the real-time and rapidity of train schedules, the algorithm can choose the action with maximal probability at each discrete step according to the trained model.

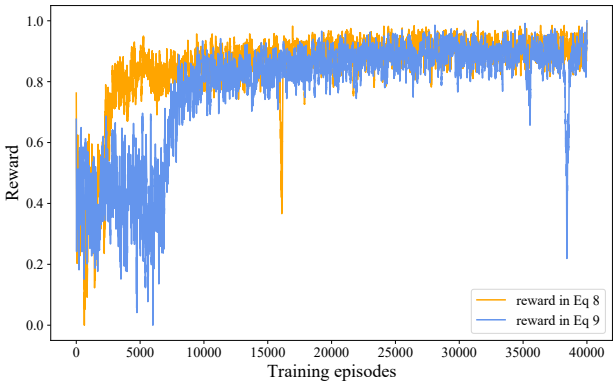

Fig. 4: Reward functions in the training.

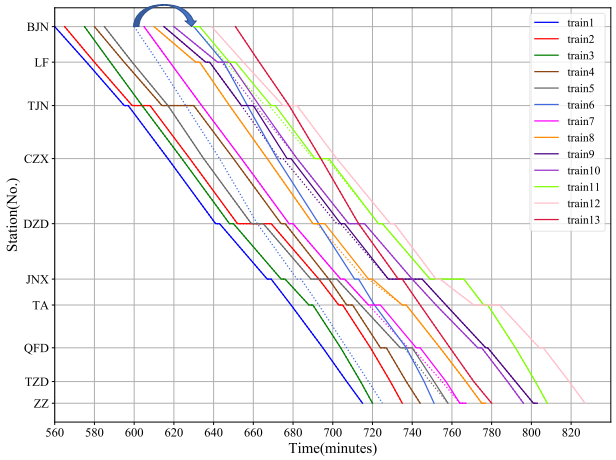

Fig. 5: Timetable rescheduling in scenario 1.

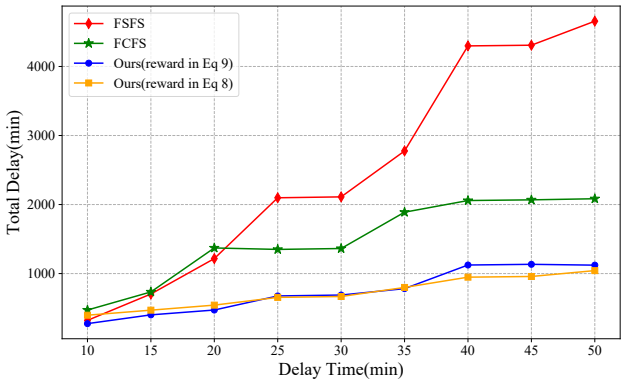

Fig. 6: Total delay with different initial delay times in scenario 1.

V. Experiments

A. Experimental Settings

To evaluate the performance of the proposed method, two numerical experiments are carried out based on the original timetable from Beijing-Shanghai high-speed railway line. Since this railway corridor provides a high-density train service, general delays will affect the operating efficiency of the entire railway line. Without loss of generality, the algorithm is evaluated with the complex scenarios consisting of 13 trains and 10 stations from Beijing to Zaozhuang, and the results of which demonstrates the superiority of our method.

In the training process, taking into account the actual carrying service of the high-speed railway line, the minimum interval time is set to 3 minutes to ensure train safety at stations and sections. The reserved delay buffer time of the operation diagram is set to 2 minutes. The parameters in reinforcement learning are as follows. The discount factor γ, learning rate α, total training episode E and update interval I are set as 1, 0.2, 40000 and 4, respectively. All experiments are performed by Python 3.6 on a PC with a 2.9 GHz Intel Core i7-10700 CPU and 16 GB RAM.

The reward function during training is shown in Fig. 4. The ordinate represents the normalized negative total delay under different disturbances. Obviously, the reward in (8) converges faster and more stably during the training process.

B. Experiments and Results Analysis

We simulate two common disturbances in railway operations. Firstly, there is a departure delay at the current station in scenario 1, as is shown in Fig. 5. Train 6 leaves BJN station 30 minutes later than the original plan, where dotted and solid lines mean the scheduled and rescheduled timetable. In our approach, train 6 overtakes the preceding train at LF, TJN, DZD, TA, QFD station to reduce delays and close to the original timetable. The involved trains ensure a safe distance by delaying the departure time at the current station and recover at the next section through the reserved buffer time. Fig. 6 shows the variation of total delay of all trains with different initial delay times in this scenario compared with the First-Come-First-Served (FCFS) and First-Scheduled-First-Served (FSFS) methods [17]. The FCFS method dynamically determines the order of services according to the arrival time, and the train which arrives first departs first. The FSFS method decides the order based on the original schedule and often pushes the original timetable backward. It can be seen that the FCFS method is superior to the FSFS method in this delay scenario. As the delay time increases, our two models with different reward functions have slight differences, but they are still better than the FCFS method. The total delay time is averagely reduced by 52.4% and 62.3% compared with the FCFS method and the FSFS method in nine experiments.

Secondly, Fig.7 shows the arrival delay at the next station. Due to the single track in the high-speed railway line, these disturbances often cause considerably large-scale delays. Train 4 arrives 25 minutes late at TJN station, which affects the regular operation plan of the following five trains. As shown in Fig. 7, the associated delayed train 7 overtakes train 4 at TJN station and overtakes train 5 at JNX station in our rescheduling approach. The agent uses the dwelling time of train 4 to give way to the faster train at TJN station. Fig. 8 shows the variation of the total delay of all trains with different initial delay times in delay scenario 2. Contrary to

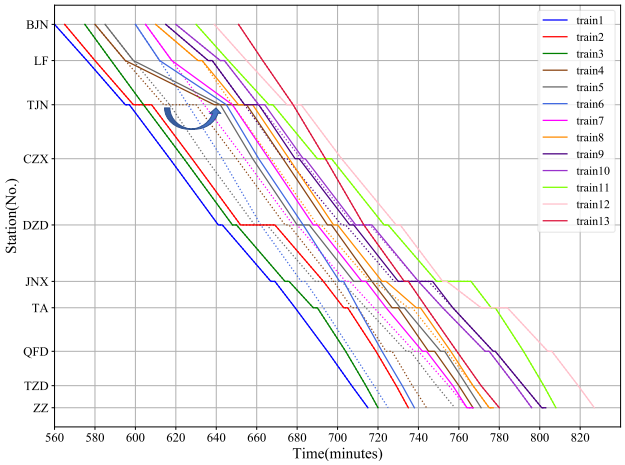

Fig. 7: Timetable rescheduling in scenario 2.

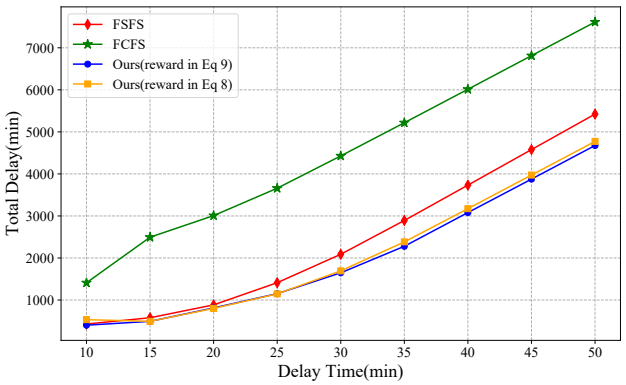

Fig. 8: Total delay with different initial delay times in scenario 2.

scenario 1, the FSFS method is a better choice than the FCFS methods in case of large-scale delays. With the increase in arrival delays, our approach shows better performance than the FSFS method. The total delay time is averagely reduced by 64.4% and 15.4% compared with the FCFS method and the FSFS method in nine experiments.

Through the experiments under two different delay cases, we find that our method has a more significant improvement than two benchmarks in departure delay scenarios. Even in considerable delay scenarios caused by arrival delays, our approach can use the rescheduling strategy at the station to reduce delays, and the results show the superiority over two benchmarks. More significantly, our approach can find a high-quality scheduling strategy in less than one second when facing different kinds of disturbances, which shows excellent potential and advantages in terms of time.

VI. Conclusion

In this paper, we propose a policy-based reinforcement learning approach for high-speed railway timetable rescheduling, where multiple safety constraints and service requirements in train operation are considered. In addition, we formulate the rescheduling problem as a sequential decision process and establish a two-stage MDP model. Based on the Monte-Carlo sampling, we design the update method and operation of the rescheduling algorithm. Finally, numerical experiments are carried out on the Beijing-Shanghai high-speed railway line and demonstrate that the proposed method can quickly find high-quality scheduling strategies under different disturbances.

In the future, we will concentrate on addressing the rescheduling problem from a microscopic perspective, such as the railway infrastructure in the environment, and set different priorities of trains and stations to make it closer to the real-world situation.

References

[1] I. A. Hansen, *Railway timetable & traffic: analysis, modelling, simulation*. Eurailpress, 2008.

[2] A. D'Ariano, F. Corman, D. Pacciarelli, and M. Pranzo, "Reordering and local rerouting strategies to manage train traffic in real time," *Transportation science*, vol. 42, no. 4, pp. 405–419, 2008.

[3] B. Ning, H. Dong, W. Zheng, J. Xun, S. Gao, H. Wang, L. Meng, and Y. Li, "Integration of train control and online rescheduling for high-speed railways: challenges and future," *Acta Automatica Sinica*, vol. 45, no. 12, pp. 2208–2217, 2019.

[4] F.-Y. Wang and S.-m. Tang, "Concepts and frameworks of artificial transportation systems," *Complex Systems and Complexity Science*, vol. 1, no. 2, pp. 52–59, 2004.

[5] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, "A tabu search algorithm for rerouting trains during rail operations," *Transportation Research Part B: Methodological*, vol. 44, no. 1, pp. 175–192, 2010.

[6] S. Dündar and İ. Şahin, "Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways," *Transportation Research Part C: Emerging Technologies*, vol. 27, pp. 1–15, 2013.

[7] A. D'ariano, D. Pacciarelli, and M. Pranzo, "A branch and bound algorithm for scheduling trains in a railway network," *European journal of operational research*, vol. 183, no. 2, pp. 643–657, 2007.

[8] B. Ning, "Parallel rail transportation system," *Chinese Journal of Intelligent Science and Technology*, vol. 1, no. 3, p. 215, 2019.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

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

[11] H. Khadilkar, "A scalable reinforcement learning algorithm for scheduling railway lines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 727–736, 2018.

[12] Y. Zhu, H. Wang, and R. M. Goverde, "Reinforcement learning in railway timetable rescheduling," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[13] M. Obara, T. Kashiyama, and Y. Sekimoto, "Deep reinforcement learning approach for train rescheduling utilizing graph theory," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 4525–4533.

[14] L. Ning, Y. Li, M. Zhou, H. Song, and H. Dong, "A deep reinforcement learning approach to high-speed train timetable rescheduling under disturbances," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3469–3474.

[15] L. Metz, J. Ibarz, N. Jaitly, and J. Davidson, "Discrete sequential prediction of continuous actions for deep rl," *arXiv preprint arXiv:1705.05035*, 2017.

[16] L. Xia, J. Xu, Y. Lan, J. Guo, W. Zeng, and X. Cheng, "Adapting markov decision process for search result diversification," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 535–544.

[17] P. Xu, F. Corman, Q. Peng, and X. Luan, "A train rescheduling model integrating speed management during disruptions of high-speed traffic under a quasi-moving block system," *Transportation Research Part B: Methodological*, vol. 104, pp. 638–666, 2017.