Hindawi
Journal of Advanced Transportation
Volume 2019, Article ID 3072495, 9 pages
https://doi.org/10.1155/2019/3072495

Research Article

Developing Train Station Parking Algorithms: New Frameworks Based on Fuzzy Reinforcement Learning

Wei Li,[1,2] Kai Xian,[2] Jiateng Yin,[3] and Dewang Chen[4]

[1]*School of Traffic and Transportation, Beijing Jiaotong University, Beijing, 100044, China*
[2]*Beijing Transport Institute, No. 9, LiuLiQiao South Lane, Fengtai District, Beijing, China*
[3]*State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, 100044, China*
[4]*College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China*

Correspondence should be addressed to Jiateng Yin; 13111054@bjtu.edu.cn and Dewang Chen; dwchen@fzu.edu.cn

Received 20 March 2019; Accepted 15 May 2019; Published 4 August 2019

Academic Editor: Luigi Dell'Olio

Train station parking (TSP) accuracy is important to enhance the efficiency of train operation and the safety of passengers for urban rail transit. However, TSP is always subject to a series of uncertain factors such as extreme weather and uncertain conditions of rail track resistances. To increase the parking accuracy, robustness, and self-learning ability, we propose new train station parking frameworks by using the reinforcement learning (RL) theory combined with the information of balises. Three algorithms were developed, involving a stochastic optimal selection algorithm (SOSA), a Q-learning algorithm (QLA), and a fuzzy function based Q-learning algorithm (FQLA) in order to reduce the parking error in urban rail transit. Meanwhile, five braking rates are adopted as the action vector of the three algorithms and some statistical indices are developed to evaluate parking errors. Simulation results based on real-world data show that the parking errors of the three algorithms are all within the ±30cm, which meet the requirement of urban rail transit.

1. Introduction

Urban rail transit systems include underground railway, streetcar, light rail transit, monorail, automated guide transit, and magnetic levitation, which have received increased attention in large cities since urban rail system has remarkable advantages in fast speed, safety, punctuality, environment friendliness, and land saving features. In modern rail transit lines, train station parking (TSP) is an important technique that needs to be addressed. Specifically, TSP refers to the running train stopping at the parking spot precisely when train enters the train station. Imprecise TSP may impact the efficiency, convenience, and safety of the urban rail transit. In particular, most newly built transit stations install platform screen doors (PSDs) to isolate the platform and rail track. PSDs can reduce the cold and hot air exchange between the platform area and rail track area. They can also prevent passengers from falling or jumping onto the rail track and ensure safety of the passengers [1, 2]. Imprecise parking could directly lead to serious consequences that the train door can not be accessed normally and passengers can not get on and off the train easily [3]. In such cases, the train will be delayed which causes the dramatic decrease of corridor capacity [4].

A lot of existing studies have addressed the TSP problem. For example, Yoshimoto et al. applied the predictive fuzzy control technology, which performs better parking accuracy by comparing with proportional-integral-derivative control method [5]. Yasunsbo et al. used fuzzy inference for automatic train parking control. The parking area was divided into several sections in order to use different fuzzy inference rules [6, 7]. Hou et al. introduced the terminal iterative learning control to train automatic stopping control. It updated the current controller using the prior parking errors in the previous iterations [8]. After several times of iterations, the train parking error was reduced below 0.5m, while no explicit statistical results were presented in this study. Zhou studied the regression of Gaussian process in machine learning and Boosting regression algorithm for the accurate parking problem, and these two methods are compared with the linear regression method [9]. Soft computing methods have been

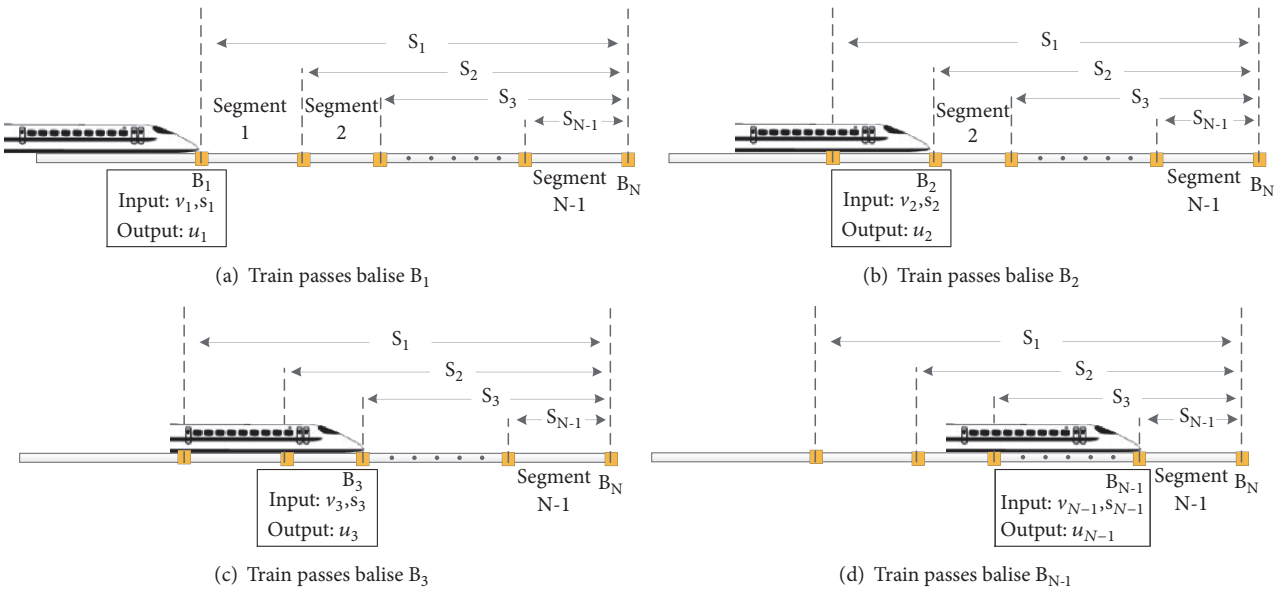

Figure 1: Parking area and balises distribution [17].

applied to train parking in urban rail transit by Chen and Gao, and three models were presented, involving a linear model, a generalized regression neural network model, and an adaptive network based fuzzy inference (ANFIS) model [10]. The ANFIS can keep the parking errors within 30cm with the probability being greater than 99.9% in simulation. However, the simulation experiments are conducted under ideal environment and no disturbances are taken into account in this study.

Based on the braking characteristics of trains in urban rail transit, two simplified models were developed to estimate the TSP error in [11]. Then, a precise stop control algorithm based on LQR (linear-quadratic-regulation) was proposed combining the vehicles dynamics model and system identification theory, which obtained the train station parking errors within 50cm. Chen et al. tried to address the train parking problem using a new machine learning technique and proposed a novel online learning control strategy with the help of the precise location data of balises installed in stations [12–14]. Experiments were conducted to test the algorithm performance under the variation of different train parameters, and the best algorithm achieved an average parking error of 3.8cm. Even though a lot of studies are proposed in the existing literature, we note that the train station parking is a very complex problem due to many influencing external factors, for example, the weather, train running states, or onboard passengers, while these uncertain factors have not been explicitly tackled in former literature [15].

The theory of reinforcement learning (RL) provides a normative account that describes how agents may optimize their control under uncertain environment [16]. As a typical algorithm in RL field, Q-learning uses action-value function to approximate the optimal control strategies of an agent and has been widely successfully implemented into many complex control problems. In particular, Q-learning utilizes the information through the interactions between an *agent* and the *environment* and gradually learns the optimal control strategies for a multistage decision problem. And this process is actually very similar with TSP where a train can be termed as an agent. Therefore, this paper aims to introduce the RL theory and Q-learning algorithms to solve the TSP problem in urban rail transit systems.

Specifically, in this paper, we first establish a simulation training platform based on the real-world data, involving the distance, speed limit, gradient, and position of balises between the Jiugong station and the Xiaohongmen station of Yizhuang Line in Beijing Subway. Three algorithms were proposed—a stochastic optimal selection algorithm (SOSA), a Q-learning algorithm (QLA), and a fuzzy function Q-learning algorithm (FQLA). Performances of three algorithms were analyzed and the experimental results were compared. The rest of the paper is organized as follows. In Section 2, TSP problem is described including five different braking rates, and the TSP simulation platform is introduced. In Section 3, three algorithms for reducing and estimating the parking errors are developed based on the reinforcement learning (RL) theory. Seven statistical indices are defined to calculate and evaluate the accuracy and reliability of these three algorithms. In Section 4, experimental results of these three algorithms are compared and analyzed in detail using the field data in simulation platform. Finally, conclusions and future research are outlined in Section 5.

2. Problem Description and Environment Construction of RL

2.1. Train Station Parking Problem. As shown in Figure 1, several balises are installed on the tracks in rail stations, and the train can correct its position and velocity once it goes through each balise. Typically, the train enters the station

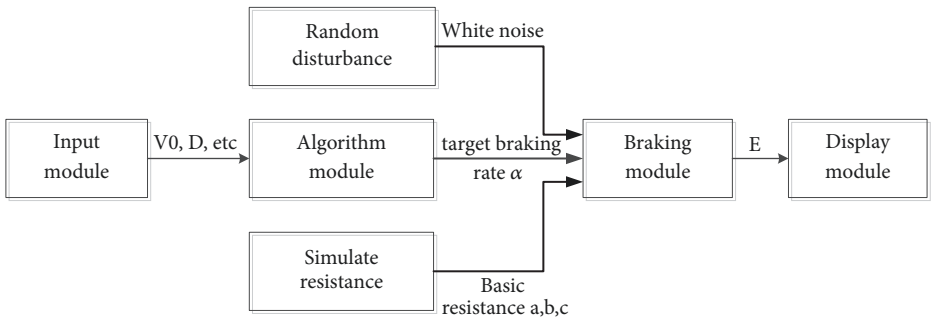

Figure 2: Structure of the parking simulation platform.

area and starts to brake at the initial spot (first balise S_1) which is L meters away from the stopping spot (last balise S_N). This area between S_1 and S_N is also called parking area. There are N balises distributed in this area (including $S_1 \cdots S_N$) according to the actual distribution of the urban rail transit. Meanwhile the distribution of the balises satisfies the fundamental requirement that the balises near the stopping spot are dense and the balises far from the stopping spot are sparse.

There are different braking rates in each balise. Every time the train reaches a balise, we will get 5 different train braking rates through the following calculation, also called 5-element vector A. They are the following:

(1) Nonlearning strategy (NLs) braking rate: we can calculate the theoretical braking rate a_i^t by using the kinematic formula. This strategy does not consider the influence of interference on parking. The target braking rate a_i^g we want equals the theoretical braking rate in this nonlearning strategy.

$$V_T^2 - V_i^2 = 2a_i^t S_i \xrightarrow{V_T=0} a_i^t = -\frac{V_i^2}{2S_i} \tag{1}$$

$$a_i^g = a_i^t \tag{2}$$

where V_T is the train speed at the last balise. V_i represents the train speed at the current balise. S_i is the distance between the current balise and the parking spot.

(2) Fixed learning strategy (FLs) braking rate: based on NLs, we add a fixed learning rate η and Δa_i which is the deviation of actual braking rate with theoretical braking rate.

$$V_{i+1}^2 - V_i^2 = 2a_i^r D_i \xrightarrow{V_{i+1}\neq 0} a_i^r = \frac{\left(V_{i+1}^2 - V_i^2\right)}{2D_i} \tag{3}$$

$$\Delta a_i = a_i^r - a_i^t \tag{4}$$

$$a_i^g = a_i^t - \eta \Delta a_i \tag{5}$$

The calculation method of a_i^t is the same as NLs where V_i represents the train speed at the current balise. V_{i+1} represents the train speed at the next balise. D_i is the distance between the current balise and the next balise.

(3) Variable learning strategy (VLs) braking rate: use variable learning rate, which takes into account the train speed passing through the balises and arrangement of the balises, instead of fixed learning rate.

$$\eta_i = \lambda \frac{V_i}{S_i} \tag{6}$$

$$a_i^g = a_i^t - \eta_i \Delta a_i \tag{7}$$

The definition of V_i and S_i is the same as in NLs. The definition of Δa_i is equivalent to Δa_i of FLs where λ is a constant, named fixed adjustment coefficient.

(4) Gradient descent learning strategy (GDLs) braking rate: objective function of GDLs is as follows:

$$J = 0.5\left(S_i - S_i^e\right)^2 \tag{8}$$

where S_i represents the actual remaining distance. S_i^e is the estimated remaining distance. We have

$$a_i^g = a_i^t - \eta_i^G \frac{\partial J}{\partial a_i^t} \tag{9}$$

where η_i^G represents the gradient descent learning rate.

(5) Newton descent learning strategy (NDLs) braking rate: objective function of NDLs is the same as GDLs, and we get

$$a_i^g = a_i^t - \eta_i^N \frac{\partial J / \partial a_i^t}{\partial^2 J / \partial \left(a_i^t\right)^2} \tag{10}$$

where η_i^N represents the Newton descent learning rate.

In practice, we can choose one of the above control strategies once a train goes through a balise to adjust the train braking rate, in order to stop the train precisely at the last balise S_N under uncertain environment. To model the uncertain environment, we next construct a simulation platform to simulate the train parking process.

2.2. Parking Simulation Platform. The parking system is used to simulate the actual train parking process in rail stations. The system mainly includes an input module, algorithm module, braking module, and display module, shown as Figure 2.

Parameters are set up in the input module according to the field data. The input parameters include initial speed V_0

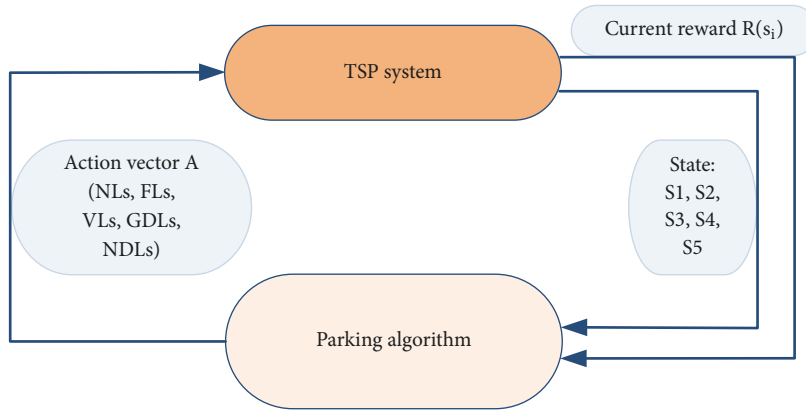

Figure 3: Diagram of state, reward, and action vector.

when the train starts braking, the length of the parking area L, the number of balises n, the distance between each balise D, resistance effect, white noise, type of algorithm, disturbance, and number of iterations. These parameters provide the initialized circumstance and related configuration information. We can change the simulation circumstance by changing the parameters above.

The main function of the algorithm module is to generate the target braking rate a. Algorithm module can calculate the 5 different train braking rates mentioned above and then select the optimal or the most appropriate target braking rate according to the algorithms.

Simulation resistance and random disturbance represent the actual urban rail transit system. The simulation resistance generates the current train resistance based on the current speed, using the formula related to the speed of the train in the form of a quadratic equation, called basic resistance unit equation:

$$W_0 = \alpha v^2 + \beta v + \varphi \tag{11}$$

where W_0 represents the basic resistance unit (N/t). v is speed of train (km/h). φ represents the part of rolling resistance unrelated to the train speed. β is parameter that is used to describe wheel flange friction resistance, and α is associated with the square of speed. The white noise generator can generate random white noise to test the anti-interference ability of the algorithms.

The braking module is used to park the train, according to the target braking rate a. The transfer function of the braking module includes time delay, time constant, and train operating mode transition. The transfer function of the braking module is

$$G_a(s) = \frac{a}{1 + T_p s} e^{-T_d s} \tag{12}$$

where a is the target braking rate, T_d represents the time delay, and T_p is the time constant.

Finally, display module mainly shows the experimental results, including parking error and target braking rate.

3. Algorithm Design and Performance Indices

3.1. Stochastic Optimal Selection Algorithm (SOSA). In SOSA, s_1 to s_n correspond to current n states of algorithm at the moment. We assume that the reward function about current state is R. R can transfer each state into the real number $s_i \xrightarrow{\mathbf{R}} \mathbb{R}$. Δ represents the distance between the running train and the current nearest balise. If the train passes the current nearest balise, the value of Δ is negative; otherwise it is positive. $R(s_i)$ is the relationship between the current state and the current reward, shown as

$$R(s_i) = \begin{cases} +3, & 0cm \le |\Delta| \le 10cm, \\ +2, & 10cm \le |\Delta| \le 20cm \\ +1, & 20cm \le |\Delta| \le 30cm, \\ -3, & 30cm \le |\Delta| \end{cases} \tag{13}$$

$$(i = 1, 2, \ldots, n)$$

When the distance Δ is less than 10cm, the reward function $R(s_i)$ is the largest among all the results. When the distance Δ is beyond ±30cm, the reward function $R(s_i)$ is smallest and negative. The reward function immediately indicates the reward of the train in its current state. In this algorithm, we do not have to consider the long-term reward of the state. The bigger the value of the reward function is, the better the performance of this algorithm will be.

We consider the train braking rate of the controller module as the output action vector A. When a train passes through a balise in the simulation, the system will change the target braking rate to ensure an accurate parking. Because of system time delay and time constant, the actual braking rate of the train will approach the target braking rate gradually.

In this paper, we adopt five braking rates. They are non-learning strategy (NLs) braking rate, fixed learning strategy (FLs) braking rate, variable learning strategy (VLs) braking rate, gradient descent learning strategy(GDLs) braking rate, and Newton descent learning strategy (NDLs) braking rate. The action vector A is a 5-element vector, including the braking rates mentioned above. A=[NLs, FLs, VLs, GDLs, NDLs], shown as Figure 3. When a train passes through

a balise, the SOSA will choose one strategy from vector A randomly as the current target braking rate. After the train passes through all balises, also called one episode, the algorithm records the parking error and action sequence. The system will start the next episode from the initial spot until $R(s_i)$ is large enough or bigger than a preset value θ.

The SOSA consist of the following steps.

Step 1. For all the states, initialize the system parameters; assume that the train enters the parking area.

Step 2. After passing through each balise, calculate the action vector A and current reward $R(s_i)$. Choose an action from A randomly as the current target braking rate.

Step 3. Store the current reward in the array; then repeat Step 2 until the train reaches the stopping spot. Store the parking error and action sequence. Go to Step 4.

Step 4. If $R(s_i) < \theta$, go to Step 1; or else use the current action sequence as the train braking rate.

3.2. Q-Learning Algorithm. In QLA, s_1 to s_n correspond to current n states of algorithm at the moment. The action vector A=[NLs, FLs, VLs, GDLs, NDLs]. Each state and action are connected with the value of $Q(s, a)$. According to the $Q(s, a)$, the algorithm will pick an action from vector A. Besides, the algorithm can also update the Q function according to the ε-greedy action selection mechanism [18].

$Q(s_i, a)$ function is the estimated total reward under the state s_i and action a. We have

$$\begin{aligned} Q(s,a) = E\left[R(s_0,a) + \gamma R(s_1,a) + \gamma^2 R(s_2,a) + \cdots \right. \\ \left. + \gamma^i R(s_i,a) \mid s_0 = s\right] = R(s,a) \\ + \gamma \sum_{S'} P_{sa}(s') Q(s',a) \end{aligned} \tag{14}$$

where the discounted factor is $0 \leq \gamma < 1$. In this paper discounted factor γ equals 0.99. s represents the current state and s' is the next state when the last state is s. P_{sa} is state transition distribution probability. It is the probability that state transfers from s_i to s_j using the action a. The train actually runs along the railway track. So no matter what action is chosen, the next state must be s_{i+1}. We decide to let $P_{sa}(s')$=1. The $R(s, a)$ is defined the same as the $R(s)$ in stochastic optimal selection algorithm.

The bigger the $Q(s, a)$ is, the larger the immediate reward in each balise will be. In order to obtain the maximum Q, the algorithm needs to update the Q function, as shown in

$$\pi(s) := \operatorname{argmax}_a \sum_{s'} P_{sa}(s') Q(s',a) \tag{15}$$

The optimal Q can be expressed as a Q^*. It represents the total reward according to the above updating strategy; we get

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') \max_a Q^*(s',a) \tag{16}$$

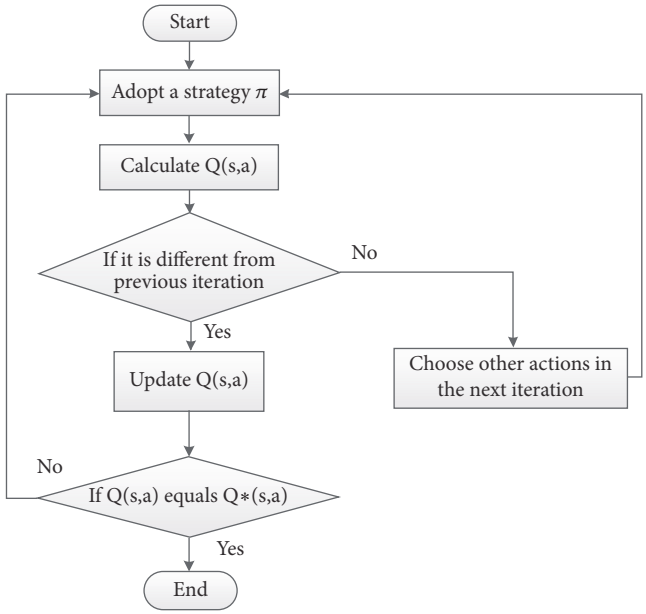

Figure 4: Training flowchart of QLA.

The overall training processes of QLA are presented in Figure 4, consisting of the following steps.

Step 1. Initialize $Q(s, a)$.

Step 2. Adopt a strategy π with any action sequence. When passing by each balise, record the sequence of state, action, and immediate reward value, using (13). After each episode, calculate the $Q(s, a)$ according to (14).

Step 3. If the current strategy and the previous selected strategy are different, then update the strategy using the action selection mechanism. Select an action with larger probability that can make $Q(s, a)$ become the largest one, like (15). On the contrary, QL will select other actions with a smaller probability.

Step 4. Repeat the steps mentioned above; when $Q(s, a) \longrightarrow Q^*(s, a)$, we can achieve best strategy $\pi \longrightarrow \pi^*$.

3.3. Fuzzy Q-Learning Algorithm. FQLA is basically the same as QLA. It is an improved version of QLA. Without using the discrete function, instead the fuzzy function Q-learning uses a continuous function where the value of immediate reward is a typical Gauss type membership function, i.e.,

$$y = f(x, \sigma, c) = e^{(x-c)^2/2\sigma^2} \tag{17}$$

and the fuzzy member function is shown in Figure 5.

The FQLA is almost the same as QLA.

Step 1. Initialize the action-value function $Q(s, a)$.

Step 2. Adopt a strategy π with any action sequence. When passing by each balise, record the sequence of state, action,

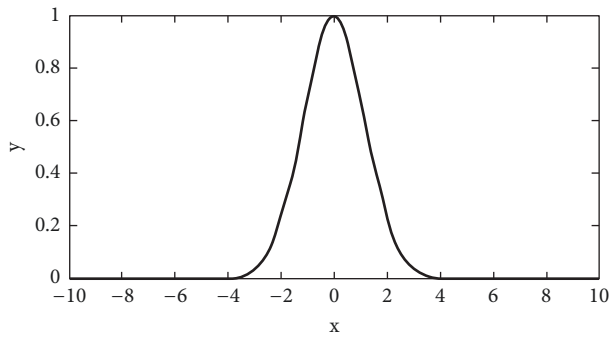

Figure 5: Reward as Gauss type membership function.

and immediate reward value, using (17). After each episode, calculate the $Q(s, a)$ according to (14).

Step 3. If the current strategy and the previous selected strategy are different, then update the strategy using the action selection mechanism. Select an action with larger probability that can make $Q(s, a)$ become the largest one, like (15). On the contrary, QL will select other actions with a smaller probability.

Step 4. Repeat the steps mentioned above; when $Q(s, a) \longrightarrow Q^*(s, a)$, we can achieve best strategy $\pi \longrightarrow \pi^*$.

3.4. The Performance Indices. To compare the performance of three algorithms, we need to define the parking performance indices. We have the following evaluation performance indices.

(1) The convergence number (N_c) is t he number of learning trails when the parking errors tend to be stable.

(2) Average errors under different speed ($\overline{E}_{ds}$): in order to test the merits of the parking algorithm, we need to have many simulation trials. Our simulations have produced a great deal of parking information. Among them, the parking error is one of the most important indices in the parking information, because it is the most direct indicator of the parking quality. With n different initial speeds, algorithm can get n parking error after each episode. The $\overline{E}_{ds}$ is the absolute average of n parking error, as shown in the following formula:

$$\overline{E}_{ds} = \frac{\sum_{i=1}^{n} |\mu_i|}{n} \tag{18}$$

where μ_i is parking error at different initial speeds.

(3) Average error ($\overline{E}$): $\overline{E}$ is the average of $\overline{E}_{ds}$, as shown in

$$\overline{E} = \frac{\sum_{i=1}^{m} \overline{E}_{dsi}}{m} \tag{19}$$

(4) The maximum error of the parking error (μ_{max}): the maximum error of the parking error here is an important indicator of the performance indices. μ_{max} is the maximum parking error among all the parking errors. If the μ_{max} is too big, it will seriously affect the parking safety of the train; we have

$$\mu_{max} = \max\left(\mu_1, \mu_2, \ldots, \mu_n\right) \tag{20}$$

(5) The minimum of the average error ($\overline{E}_{min}$): the minimum error of the parking error is also an important indicator. $\overline{E}_{min}$ is the minimum among all the $\overline{E}_{ds}$. We have

$$\overline{E}_{min} = \min\left(\overline{E}_{ds1}, \overline{E}_{ds2}, \ldots, \overline{E}_{dsm}\right) \tag{21}$$

(6) Root of mean square (σ): the σ reflects the parking error fluctuations (discrete degree). In the process of parking algorithm, we need to assess the fluctuation which indicates the stability of the algorithm; we have

$$\sigma = \sqrt{\frac{\sum_{i=1}^{m} \left(\overline{E} - \overline{E}_{dsi}\right)^2}{m - 1}} \tag{22}$$

(7) Parking probability in the required range (P_p): P_p is a measurement of the estimated reliability. This is an important index to control the automatic parking ability. The parking probability will be unavailable, if the P_p is too small to meet the requirement of practical use.

$$P_i = \frac{\left(x - \overline{E}\right)}{\sigma} \tag{23}$$

$$P_p = \left(1 - 2\left(1 - \int_{-\infty}^{P_i} \frac{1}{\sigma\sqrt{2\pi}} e^{-P_i^2/2} dP_i\right)\right) \times 100\% \tag{24}$$

A large number of studies have given the parking error of the train within the range [-30cm,30cm] and made the parking probability requirement larger than 99.5% [8, 10, 13]. For example, the third line of Dalian rail transit in China requires train parking accuracy to be within [-30cm,30cm] and the required reliability is 99.5% [10]. In Beijing Subway, the allowable parking error is also 30cm with over 99.8% reliability.

4. The Experimental Results

To approach the real-world running scenaric, according to the above mathematical model, we choose field data between the Jiugong station and the Xiaohongmen station of Yizhuang Line of Beijing Subway in the train station parking simulation platform. We set that the length of the parking area is 100 meters. There are six balises in this parking area. The distance between the balises and the parking spot is 100m, 64m, 36m, 16m, 4m, and 0m, respectively. The arrangement of the balises meets the actual balises.

The maximum velocity (i.e., speed limit) for the train to enter into the area is 14.28m/s. While there exists time delay in the train braking system, we select 11.5m/s as the maximum initial speed of a train based on actual operating data. In the algorithms, we did not consider the parking error under one single speed. By using single speed we can not fully test the reliability and adaptability of the algorithm. So we set that the initial speed varies from 9m/s to 11.5m/s, and we pick 10 different initial speeds in every 0.25m/s speed interval. The purpose is to consider the influence under the variable initial speed. The threshold of the SOSA is [-5cm, 5cm]. We use Gauss membership function in FQLA to

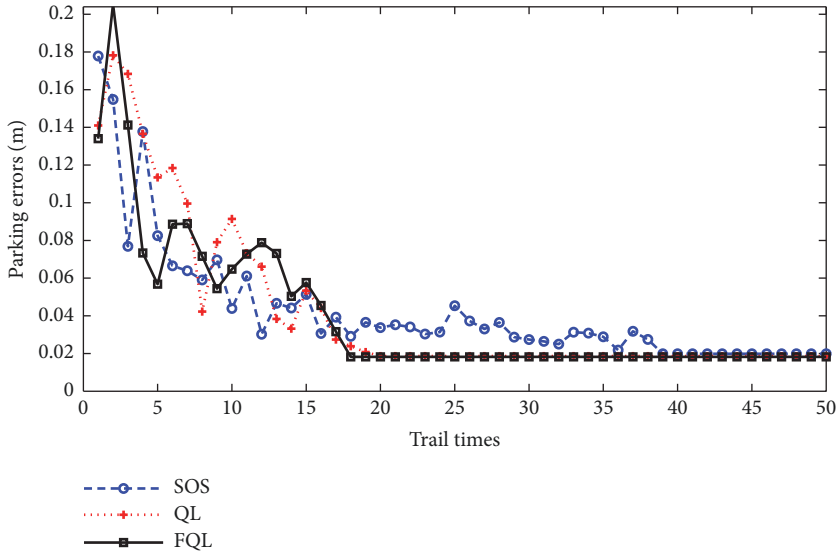

Figure 6: 50 trials of three algorithms about 10 different initial speeds.

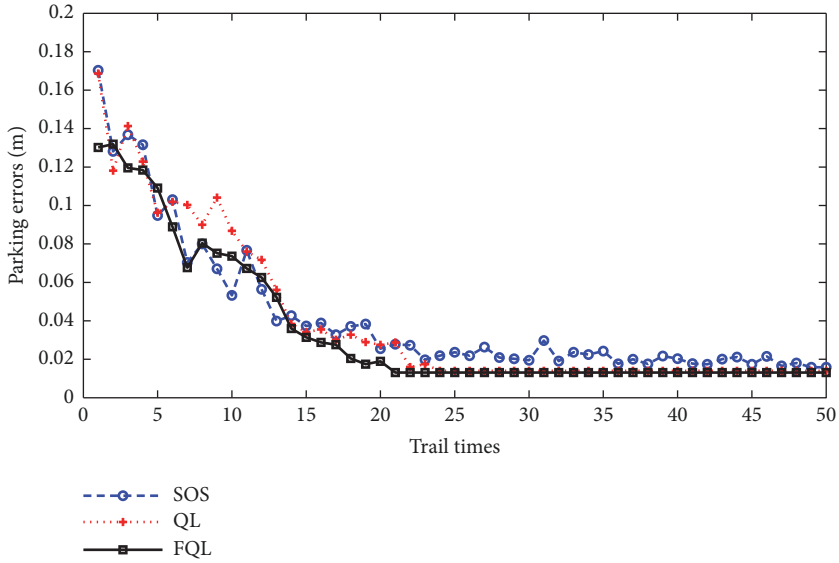

Figure 7: 50 trials of three algorithms about 100 different initial speeds.

substitute the current reward $R(s_i)$ in QLA. After debugging and experimenting, we pick parameters σ=0.1, c=0. Figure 6 presents the errors $\overline{E}_{ds}$ of 10 different initial speeds during 50 trials. As can be seen in Figure 6, the convergence iterations N_c are different by the three algorithms. The parking error $\overline{E}_{ds}$ is becoming smaller and smaller after times of trail, and the results also demonstrate that most of the parking errors can be smaller than ±30cm in the end. With respect to the computational efforts, we note that these three algorithms are very close because they are all based on the theory of reinforcement learning. Since each trial includes 10 times of train station parking process with different initial speed, it takes about 15 seconds to complete each trial and the total training time is about 750 seconds.

We also conduct more trials to test the variation of N_c when there are 100 initial speeds. When the numbers of initial speed reach 100, we obtain the variation of train parking errors, as shown in Figure 7.

The N_c we get is larger than the N_c received under 10 initial speeds. We can conclude from Figures 6 and 7 that the more initial speeds are set, the slower convergence and larger convergence number it will be.

The performance comparison of these three algorithm can be seen in Tables 1 and 2. It is clear that QLA and FQLA have smaller N_c than SOSA. This indicates that the convergence is much faster in QLA and FQLA, especially when the number of initial speeds is increased.

As can be seen from the figures and tables, the developed three algorithms work well on the field data between two stations in Beijing Subway, which can improve most of the performance indices, especially in decreasing $\overline{E}$. QLA and FQLA have smaller $\overline{E}$ than SOSA. The N_c rate of FQLA is

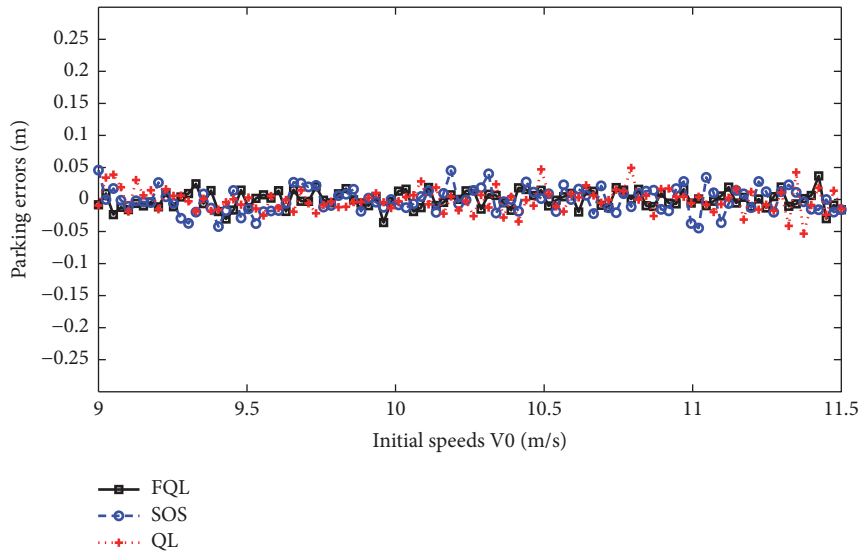

Figure 8: Each parking error after 50 trials about different initial speeds.

Table 1: Performance indices of three algorithms with 10 initial speeds.

Indicators	N_C	$\overline{E}(cm)$	$\overline{E}_{min}(cm)$	$\sigma(cm)$	P_p
SOSA	39	4.27	1.99	3.34	99.99%
QLA	20	4.24	1.86	4.32	99.99%
FQLA	18	3.99	1.83	3.95	99.99%

Table 2: Performance indices of three algorithms with 100 initial speeds.

Indicators	N_C	$\overline{E}(cm)$	$\overline{E}_{min}(cm)$	$\sigma(cm)$	P_p
SOSA	49	4.18	1.58	3.67	99.99%
QLA	24	3.87	1.40	4.14	99.99%
FQLA	21	3.46	1.31	3.65	99.99%

about 19 times, and it has the fastest convergence among the three algorithms. FQLA has the smallest $\overline{E}$ and $\overline{E}_{min}$. QLA achieves the largest σ. When it comes to P_p, the P_p of the three algorithms is almost the same and they are all larger than 99.5%. We can obtain the following conclusions: (1) these three algorithms can adapt to the field data, which have good performance indices and gradually reduce the parking error; (2) in general, FQLA ranks the first and SOSA ranks the third. SOSA has the worst N_c, $\overline{E}$, and $\overline{E}_{min}$; (3) three algorithms can all meet the requirements of accurate parking and are easy to be implemented. However, the intelligence, exploration, and self-learning of SOSA are much worse than those of QLA and FQLA.

In addition, we also test the performance of algorithms with different initial speed after 50 times of trails. Figure 8 presents the parking errors under each different initial speed.

In Figure 8, we see that $\overline{E}_{ds}$ is within the range of ±30cm and the errors of QLA and FQLA are closer to 0cm than that of SOSA. The results also show us that the error fluctuation of FQLA performs better than SOSA and QLA. We can almost come to similar conclusions from Tables 1 and 2. FQLA performs the best among the three algorithms, indicating that adding fuzzy functions into Q-learning can further improve its performance for TSP. In addition, it is worth to mention that [13] proposed several online learning algorithms for train station parking problem. The train parking errors by these online learning algorithms are about 5cm to 6cm. We see from the above results that our developed algorithms can further reduce the parking errors compared with existing studies.

5. Conclusions

To address the TSP issue in urban rail transit, this paper developed one stochastic algorithm and two machine learning algorithms based on theory of reinforcement learning. Furthermore, we propose five braking rates as the action vector of the three algorithms within the scope of RL. Performance indices were defined to evaluate the performance and reliability of the issue. We evaluated the proposed algorithms in the simulation platform with field data collected in Beijing Subway. Our results show that these three algorithms can achieve good results that guarantee the train station parking error within 30 cm. In addition, fuzzy Q-learning algorithm performs the best among these algorithms due to its fast and stable convergence. In practical applications, the rail managers could use the simulation platform to optimize the value function with the aid of our developed algorithms and then use the well-trained Q functions to generate the optimal train parking actions in real time to reduce the parking errors of trains.

In the future, we will explore new algorithms and other methods based on soft computing and machine learning, to further increase the accuracy and reliability for the train parking. In particular, we will combine reinforcement learning

with deep learning algorithms to further enhance the flexibility and accuracy for TSP. Meanwhile, there still exist many practical issues which need to be taken into consideration in this paper. For example, we only consider the TSP problem in urban rail transit, while the train braking system is much more complex in high-speed railways. Compared with urban rail transit networks, in which each line is a relatively enclosed system that is independent of each other and each train moves like a "shuttle" on the fixed tracks, the high-speed railway networks are relatively open systems with heterogeneous trains. The trains usually have different circulation plans, and each train may travel among different railway lines according to its circulation plan. Therefore, the design of TSP algorithms for these open systems would be definitely much more complex than that of urban transit systems in order to be flexible with various external circumstances. We will fulfill these issues in our future research.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors claim that there are no conflicts of interest in this paper.

Acknowledgments

This research was supported by the National Natural Science Foundation of China through Grants 61790570, 61790573, and U1734210, by the Key Laboratory of Intelligent Metro of Universities in Fujian through Grant 2018IMUF001, by the Beijing Jiaotong University Education Foundation through Grant 9907006519, and by the China Postdoctoral Science Foundation Funded Project through Grant 2019M650474.

References

[1] J. Yin, T. Tang, L. Yang, J. Xun, Y. Huang, and Z. Gao, "Research and development of automatic train operation for railway transportation systems: a survey," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 548–572, 2017.

[2] E. Coscetta and A. Carteni, "The hedonic value of railways terminals: a quantitative analysis of the impact of stations quality on travelers behavior," *Trasportation Research Part A*, vol. 61, pp. 41–52, 2014.

[3] E. Ben-Elia and D. Ettema, "Rewarding rush-hour avoidance: a study of commuters' travel behavior," *Transportation Research Part A: Policy and Practice*, vol. 45, no. 7, pp. 567–582, 2011.

[4] J. Yin, L. Yang, X. Zhou, T. Tang, and Z. Gao, "Balancing a one-way corridor capacity and safety-oriented reliability: a stochastic optimization approach for metro train timetabling," *Naval Research Logistics (NRL)*, vol. 66, no. 4, pp. 297–320, 2019.

[5] K. Yoshimoto, K. Kataoka, and K. Komaya, "A feasibility study of train automatic stop control using range sensors," in *Proceedings of the 2001 IEEE Intelligent Transportation Systems*, pp. 802–807, USA, August 2001.

[6] S. Yasunobu, S. Miyamoto, and H. Ihara, "A fuzzy control for train automatic stop control," *Transactions of the Society of Instrument and Control Engineers*, vol. 2, no. 1, pp. 1–8, 2002.

[7] S. Yasunobu and S. Miyamoto, "Automatic train operation system by predictive fuzzy control," *Industrial Applications of Fuzzy Control*, vol. 1, no. 18, pp. 1–18, 1985.

[8] Z. Hou, Y. Wang, C. Yin, and T. Tang, "Terminal iterative learning control based station stop control of a train," *International Journal of Control*, vol. 84, no. 7, pp. 1263–1274, 2011.

[9] J. Zhou and D. Chen, "Applications of machine learning methods in problem of precise train stopping," *Computer Engineering and applications*, vol. 46, no. 25, pp. 226–230, 2010.

[10] D. Chen and C. Gao, "Soft computing methods applied to train station parking in urban rail transit," *Applied Soft Computing*, vol. 12, no. 2, pp. 759–767, 2012.

[11] G. He, *Research on, Train Precision Stop Control Algorithm Based on LQR*, Beijing Jiaotong University, Beijing, China, 2009.

[12] R. Chen and D. Chen, "Heuristic self-learning algorithms and simulation of train station stop," *Railway Computer Application*, vol. 22, no. 6, pp. 9–13, 2013.

[13] D. Chen, R. Chen, Y. Li, and T. Tang, "Online learning algorithms for train automatic stop control using precise location data of balises," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1526–1535, 2013.

[14] J. Yin, D. Chen, and L. Li, "Intelligent train operation algorithms for subway by expert system and reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2561–2571, 2014.

[15] J. Liu, "Analysis of station dwelling error and adjustment solution for operation after screen door's installation," *Urban Mass Transit*, vol. 1, pp. 44–47, 2009.

[16] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[17] J. Yin, D. Chen, T. Tang, L. Zhu, and W. Zhu, "Balise arrangement optimization for train station parking via expert knowledge and genetic algorithm," *Applied Mathematical Modelling*, vol. 40, no. 19-20, pp. 8513–8529, 2016.

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