Deep Reinforcement Learning Based Dynamic Bus Timetable Scheduling with Bidirectional Constraints

Jiahao Xie[1], Zhuo Lin[1], Jieli Yin[2], Zhaoyu Lai[2], Xijun Wang[1(✉)], and Xiang Chen[1]

[1] School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China
wangxijun@mail.sysu.edu.cn
[2] Guangzhou Jiaoxintou Technology Co., Ltd, Guangzhou 510145, China

Abstract. Bus timetable scheduling is a vital issue in reducing the operational costs of bus companies and improving service quality. Existing methods schedule the timetable only for one direction of the bus line, which may lead to the issue of inconsistent departure numbers for the upward and downward directions on the same bus line. For bus companies, it is very important to have an equal number of departures in both directions. To ensure each bus can return to its original departure point, subsequent bus schedules should be based on a timetable that ensures an equal number of departures in both directions. This study proposes a Deep Reinforcement Learning-based dynamic bus Timetable Scheduling method with Bidirectional Constraints (DRL-TSBC). In DRL-TSBC, the problem of bus timetable scheduling in both directions is formulated as a Markov Decision Process (MDP). A Deep Q-Network (DQN) is applied to determine whether to depart in both directions every minute. We construct a state that includes information on bus lines in both directions. Considering the need for consistency in the number of departures in both directions, we design a reward function to ensure an equal number of departures in both directions, while also balancing the bus company's costs with the passengers' experience. Through experiments, we find that DRL-TSBC can effectively reduce the average waiting time for passengers compared to the scheme currently used in reality. At the same time, when dealing with dynamic passenger flow changes, DRL-TSBC can make adjustments to adapt to the changes in passenger flow.

Keywords: Traffic Big Data ; Bus Timetable Scheduling ; Online Scheduling ; Deep Reinforcement Learning ; DQN

1 Introduction

In urban transportation systems, the public transit system is a crucial component. A well-developed and convenient public transit system can reduce energy consumption, protect the environment, and significantly alleviate traffic congestion. However, due to the current issues of long waiting times and severe

passenger flow congestion in public transportation, many people opt not to use public transit when traveling [1]. At the same time, the time required for passengers to travel using public transportation often increases unnecessarily, leading to a significant problem of economic waste [2]. Therefore, it is very necessary to improve the efficiency of the bus system.

Optimizing the bus departure timetable is an effective method to enhance the efficiency of the public transportation system and reduce the operating costs of bus companies. An efficient bus departure timetable should balance the number of departures with passenger waiting times to meet the different needs of the bus company and passengers [3]. Additionally, in practical applications, for both directions of a bus line, bus companies only require timetables where the number of departures is the same for both directions. This is to ensure that when arranging buses to complete the timetable, each bus can return to its original departure point, avoiding the waste caused by the need for buses to return. This requires that when generating the departure timetable, the relationship between the number of departures in both directions should also be considered. There are several prior bus timetable optimization methods, such as heuristic algorithm [4], genetic algorithm [5, 6], and reinforcement learning [7, 8]. All these methods are unable to adjust online while the passenger flow changes in both directions and at the same time, achieve a consistent number of departures in both directions. However, the passenger flow may suddenly change due to various reasons [9]. After these changes, the bus company still hopes to get a timetable with the same number of departures in both directions. In these cases, the timetables obtained from the above methods cannot meet the needs of bus companies and passengers well.

To address the problems mentioned above, we propose a Deep Reinforcement Learning-based dynamic bus Timetable Scheduling method with Bidirectional Constraints (DRL-TSBC), which can generate a bus timetable based on dynamically changing passenger flow while ensuring the consistency of the number of departures in both directions. Moreover, in the case of sudden changes in passenger flow, DRL-TSBC can make real-time adjustments to cope with the changes.

2 Related Works

2.1 Traditional Bus Timetable Scheduling Methods

Heuristic algorithms are extensively utilized in scheduling bus timetables. Ceder et al. [10] used a heuristic algorithm to create a timetable to maximize the synchronization among the buses. Han et al. [11] used genetic algorithms for multi-objective optimization, and generated bus timetables while considering both passengers and bus companies. Gkiotsalitis et al. [12] developed a novel genetic approach for optimizing bus timetables, taking into account the uncertainty in passenger travel times. Tang et al. [6] implemented the Non-dominated Sorting Genetic Algorithm-II to efficiently find Pareto optimal solutions for a

bus timetable optimization model. Yu et al. [13] used genetic algorithms to optimize bus departure intervals and achieved shorter average waiting times for passengers.

Mathematical programming algorithms are also used in the bus timetable scheduling problem. Ceder [14] used point check data and use of ride check data to minimize the required bus runs and number of buses on the timetable. Wihartiko et al. [5] solved the bus timetable scheduling problem by integer programming model with a modified genetic algorithm. Shang et al. [15] proposed an initial bus timetabling method that considers passenger satisfaction to optimize bus frequency and headway.

All the methods above can only generate fixed bus timetables, and cannot dynamically adjust the bus timetable based on real-time passenger flow changes.

2.2 Reinforcement Learning in Bus Timetable Scheduling

In practical applications, bus companies first generate a bus departure timetable (i.e., bus timetable scheduling), and then select buses to complete this timetable according to it (i.e., bus scheduling). This divides the bus scheduling problem into two parts, and current reinforcement learning methods optimize these two issues separately.

For bus timetable scheduling, Yan et al. [16] proposed a multi-agent reinforcement learning network to optimize the timetable under the influence of uncertain factors such as weather, thereby reducing the operating costs of bus systems. Li et al. [17] proposed a method leveraging existing human experience combined with deep reinforcement learning to dynamically optimize the interval time for the next departure, adapting to changes in traffic conditions and passenger demand. Zhao et al. [8] introduced a deep reinforcement learning approach to dynamically determine bus dispatching at the starting bus stop for high-frequency bus service lines, enabling bus scheduling at any time granularity. Ai et al. [7] proposed a deep reinforcement learning model that can make decisions every minute, and achieve online bus timetable scheduling that can adapt to sudden changes in passenger flow.

In summary, there are many applications of reinforcement learning in bus timetable scheduling. However, all methods mentioned above can't simultaneously schedule both directions of a bus line and ensure that the number of departures in both directions is consistent. At the same time, some bus scheduling methods require a timetable based on the consistency of the number of departures in both directions [18, 19]. Therefore, the simultaneous scheduling in both directions implemented by our DRL-TSBC is of great significance.

3 The Bus Timetable Scheduling Problem

The bus timetable scheduling problem includes two parts: how to implement dynamic bus timetable scheduling in a single direction, and how to ensure the number of departures in both directions is consistent.

In the bus system, the majority of bus lines consist of two directions: upward and downward. For these two directions, there may be inconsistencies in the number of stops or different stops, but for a bus line, the first and last stops are the same. When generating the bus departure timetable, we consider the real-time passenger flow in their respective directions. In both directions, passengers hope to wait less time, while bus companies hope to reduce the number of departures to save costs. Dynamic bus timetable scheduling means that at each decision point, the decision to depart is made by judging the passenger flow situation at the current time. Meanwhile, bus companies will limit the maximum departure intervals T_{max} and minimum departure intervals T_{min}, and only when the interval between this departure and the last departure is within the range is it allowed. For a bus line with two directions, it is necessary to have the same number of departures. This means that when arranging buses, all buses can return to their original departure points, which can avoid waste. So this problem also includes how to achieve consistent departure times when scheduling two directions at the same time.

When generating the bus timetable, for the upward and downward directions, we decide whether to depart or not based on the passenger flow data at the current time. If it is decided that a direction needs to depart, then a bus will be dispatched from the starting stop in that direction. This bus will move forward on the line over time. After arriving at a stop, passengers will get on and off the bus. At this time, some passengers will not be able to get on the bus due to the bus being full. We call these passengers stranded passengers. In bus operations, if stranded passengers occur, it will greatly reduce the satisfaction of passengers [20], so we need to minimize the number of stranded passengers as much as possible.

In the simulation environment, the bus system changes every minute, which is consistent with the method mentioned in [6, 7]. The controller will decide whether to depart every minute. The simulation environment includes the passenger flow information of the two directions of the current bus line. It will simulate when passengers will get on the bus at what stops throughout the day and what their destination stops are. It will also simulate the operation of buses on the line according to the traffic conditions. The controller will make a decision based on the environmental information of the current minute and apply this decision to the environment to control whether to depart at that time.

4 Dynamic Bus Timetable Scheduling with Bidirectional Constraints Based on Deep Reinforcement Learning

4.1 MDP Model of Bus Timetable Scheduling Problem

We reformulate the bus timetable scheduling problem using a MDP. The state space, action space, and reward function of the MDP are designed as follows.

State Space In the learning process of DQN, perceiving the environmental state from the state space is a very important part. The agent needs to obtain

important features of the current environment from the state space for efficient learning. To represent the different states in both directions of a bus line, we define the state space at mth minute as follows:

$$s_m = [a_m^1, a_m^2, x_m^1, x_m^2, x_m^3, x_m^4, y_m^1, y_m^2, y_m^3, y_m^4] \tag{1}$$

In the state space, we set all states within $[0, 1]$. Since similar states may occur at different times, we incorporate the states representing time into the state space. $a_m^1 = t_h/24$ and $a_m^2 = t_m/60$. t_h represents the current hour, and t_m represents the current minute within that hour.

To represent the information in both upward and downward directions, we have designed a pair of states with the same structure, denoted as x_m and y_m. Here, x_m represents the state of the upward direction, and y_m represents the state of the downward direction. The construction of the states is the same for both directions. We will take the state x in the upward direction as an example to introduce the structure of these states.

In the bus system, the maximum full load rate is an important indicator, which reflects the degree of passenger demand for the bus, which is calculated $x_m^1 = C_{max}^{m,up}/C_{max}$, where $C_{max}^{m,up}$ is the maximum cross-sectional passenger flow that the bus departing in the mth minute in the upward direction can achieve. C_{max} is the capacity of the bus. When $x_m^1 = 1$, it means that the bus is fully loaded, which means that some passengers may be stranded.

x_m^2 represents the normalized passenger waiting time, which is calculated by $x_m^2 = W_m^{up}/\mu$, where W_m^{up} is the sum of the waiting times of all passengers who take the bus departed at the mth minute in the upward direction, which can be calculated as Eq. (2).

$$W_m = \sum_{k=1}^{(K-1)} \sum_{i=1}^{l_m^k} (t_b^{m,i,k} - t_a^{m,i,k}) \tag{2}$$

where $t_a^{m,i,k}$ and $t_b^{m,i,k}$ represent the time when passenger i arrives at station k and gets on the bus at the mth minute. K denotes the total count of bus stops. l_m^k represents the cumulative number of passengers boarding the bus at the kth station during the mth minute. μ is a parameter that normalizes x_m^2 to a number within $[0, 1]$.

The utilization rate of the carrying capacity serves as a crucial parameter, which reflects whether the departure at the current moment is necessary, so it needs to be used as a state. Thus, x_m^3 is calculated by $x_m^3 = o_m^{up}/e_m^{up}$, where o_m/e_m is the utilization rate of carrying capacity [7], the carrying capacity that can be provided by a bus for one departure e_m is calculated by $e_m = \alpha \times C \times (K-1)$, where K is the number of stops in this direction, and C is the number of seats on the bus. Considering the limited standing space on the bus, the actual number of passengers accommodated is determined by $\alpha \times C$, where α is set to 1.5 [21]. o_m signifies the carrying capacity utilized by passengers on the bus departing at the mth minute, calculated as the total number of passengers boarding at all stations, which is calculated by Eq. 3.

$$o_m = \sum_{k=1}^{K-1} (c_m^k + l_m^k - h_m^k) \tag{3}$$

If a bus departs at the mth minute and arrives at the kth stop, c_m^k denotes the number of passengers on board, l_m^k represents the number of passengers boarding, and h_m^k indicates the number of passengers getting off.

In addition, to guide the DQN agent to achieve consistency in the number of departures in both directions, we provide relevant guidance in the reward function, and the number of departures in the current direction is also part of the state, which is calculated by $x_m^4 = c_m^{up}/\delta$, where c_m^{up} is the number of departure in the upward direction at mth minute, and δ is a parameter that normalizes x_m^4 to a number within $[0, 1]$.

Similarly, we can define the states in the downward direction by using the downward parameters.

Action Space To describe the decision-making of the agent regarding the departure status in two directions, action a can be represented by a vector, $a = (a^{up}, a^{down})$, where a^{up} is the upward action, and a^{down} is the downward action. For each direction, 0 represents "no departure", and 1 represents "departure". For example, $a = (0, 1)$ means that only the downward direction departs, and $a = (1, 1)$ means that both directions depart.

Reward Function To guide the DQN agent to achieve the goal, that is, to dynamically schedule the timetable to real-time passenger flow in two directions, and to ensure that the number of departures in both directions is consistent, we have crafted a reward function that directly connects to the main objective. The impact of current passenger flow can be represented by the utilization rate of carrying capacity [21] and the probability of passengers being stranded [7]. At the same time, the consistency constraint of the number of departures in two directions can be represented by the difference in the number of departures in the two directions.

In the problem we are discussing, we need to consider two directions on a bus line, so we divide the reward function into two parts, r^{up} and r^{down}. At the mth minute, we calculate the reward through $r_m = r_m^{up} + r_m^{down}$.

We deal with the two directions separately. For one of the directions, there are only two actions, "departure" or "no departure". In this direction, we can define the utilization rate of carrying capacity as o_m/e_m. Suppose a bus departs at the mth minute, the capacity that this car can provide is e_m, and the actual capacity consumption is o_m. o_m/e_m is a number less than 1, if this value is larger, it indicates that there is a large passenger flow at the current moment, so it is more necessary to dispatch a bus at the mth minute. On the other hand, if $1 - o_m/e_m$ is larger, it indicates that not dispatching a bus at the current moment is more in line with the state of low passenger flow. So if the action

is "departure", then o_m/e_m will be part of the reward, and if the action is "no departure", $1 - o_m/e_m$ will be part of the reward.

If the action is "no departure", the increase in passenger waiting time caused by this should be considered in the reward function. Consequently, if the decision is to opt for "no departure," the agent faces a penalty of $\omega \times W_m$. Here, W_m aggregates the waiting times of all passengers boarding the bus departing at the mth minute. ω is a parameter that keeps $\omega \times W_m$ consistently within $[0, 1]$.

Furthermore, if stranded passengers occur, it will greatly reduce the satisfaction of passengers [20]. Therefore, whether or not to depart, we introduce a penalty $\beta \times d_m$, indicating that the current scheduling plan has already incurred a penalty brought about by stranded passengers. d_m denotes the total count of passengers unable to board the bus that departed at the mth minute due to it being at full capacity. We set $\beta = 0.2$ so that when there are too many stranded passengers, the reward will become a negative number [7].

To achieve an equal number of departures in the two directions, we need to evaluate the impact of the difference in the number of departures in the two directions. Assuming the number of departures in the current direction is more than the other direction. If the action is "no departure", a positive reward will be obtained. If the action is "departure", a negative reward will be obtained. Conversely, if the number of departures in the current direction is less than the other direction, "no departure" will bring a negative reward, and "departure" will bring a positive reward. This is calculated by $\zeta \times (c_m^{up} - c_m^{down})$, where c_m^{up} is the number of departures in the upward direction at mth minute, c_m^{down} is the number of departure in the downward direction at mth minute. We set $\zeta = 0.002$ to ensure that it will not have too much impact on the scheduling to meet dynamic passenger flow.

In summary, we use up and down to distinguish the states from the two directions. And we use 0 to represent "no departure", and 1 to represent "departure", the reward function of upward direction and downward direction is shown in Eq. (4) and Eq. (5).

$$r_m^{up} = \begin{cases} 1 - (o_m^{up}/e_m^{up}) - (\omega \times W_m^{up}) - (\beta \times d_m^{up}) + \zeta(c_m^{up} - c_m^{down}), & a^{up} = 0 \\ (o_m^{up}/e_m^{up}) - (\beta \times d_m^{up}) - \zeta(c_m^{up} - c_m^{down}), & a^{up} = 1 \end{cases} \tag{4}$$

$$r_m^{down} = \begin{cases} 1 - (o_m^{down}/e_m^{down}) - (\omega \times W_m^{down}) - (\beta \times d_m^{down}) \\ -\zeta(c_m^{up} - c_m^{down}), & a^{down} = 0 \\ \left(o_m^{down}/e_m^{down}\right) - (\beta \times d_m^{down}) + \zeta(c_m^{up} - c_m^{down}), & a^{down} = 1 \end{cases} \tag{5}$$

The sum of the rewards calculated in both directions $r_m = r_m^{up} + r_m^{down}$ represents the immediate reward the DQN agent obtains.

4.2 Deep Reinforcement Learning Agent

For a bus line, the times of the first and last buses are fixed, and there will be a departure at these two times. In DRL-TSBC, in addition to the stipulated

first and last bus times, the agent will choose an action every minute based on the current state. If this action is to depart, then the corresponding time will be recorded in the departure timetable of the corresponding direction. After choosing an action ("no departure", "upward direction departure", "downward direction departure" or "both directions departure"), the agent will get the next state and the reward for executing the current state from the simulation environment, and learn at a certain frequency.

Due to the need to handle both directions of a bus line simultaneously, we encounter a sizable state space, potentially triggering the dimensionality disaster problem. To solve this issue, we employ DQN [22], which melds deep learning networks with Q-learning, serving as the agent.

DQN demonstrates superior performance compared to traditional reinforcement learning methods when it comes to addressing problems characterized by high-dimensional state and action spaces. At the same time, DQN uses target networks and experience replay, which can better balance estimation errors and variance, and improve the convergence of the algorithm. Throughout the DQN learning process, the agent gathers experience from the simulation environment, comprising the current state s, action a, reward r, and subsequent state s', which are stored as tuples in the experience replay buffer D in the format (s, a, r, s').

Algorithm 1 Learning process of DRL-TSBC

Hyperparameters: Replay buffer size M, batch size B, discount factor γ, probability ϵ, the maximum number of episodes E, learning frequency P, update frequency O, first bus time t_s, last bus time t_e.
Initialize the replay memory D, the current network parameters θ, and the target network parameters θ^- such that $\theta^- = \theta$.
Initialize the current network $Q(s, a; \theta)$ and the target network $Q(s, a; \theta^-)$.
for eposode = 1 to E **do**
    Initialize the bus system.
    for i = t_s to t_e **do**
        Select a random action a with probability ϵ
        Otherwise select $a = \arg\max_a Q(s, a; \theta)$
        Calculate I^{up} and I^{down} using a, and update a using I^{up} and I^{down}
        Execute action a and observe the reward r and the next state s'
        Add the quadruple (s, a, r, s') into D and remove the oldest quadruple if $|D| > M$
        $s = s', i = i + 1$
        if $|D| > M$ and $i \mod P == 0$ **then**
            Sample randomly B quadruples (s, a, r, s') form D
            Calculate the loss function L by Eq. (6)
            Update θ with L using Adam back propagation
            Update target parameters, $\theta^- = \theta$, after every O times of learning

During each training, a mini-batch of experience is randomly sampled from D. The Q-learning target is calculated using the old, fixed parameters θ^-. The parameters θ^- are updated every fixed number of rounds. The Mean Squared Error (MSE) is optimized between the Q-network and the target Q-network. The loss function is calculated by Eq. (6).

$$\mathcal{L}_i\left(\theta_i\right)=\mathbb{E}_{s,a,r,s' \sim D_i}\left[\left(r+\gamma \max_{a'} Q\left(s', a' ; \theta_i^-\right)-Q\left(s, a ; \theta_i\right)\right)^2\right] \tag{6}$$

where the loss function $\mathcal{L}_i(\theta_i)$ represents the error between the predicted Q-value and the target Q-value. $Q(s', a'; \theta_i^-)$ is the predicted Q-value for the next state-action pair using target network parameters θ_i^-. $Q(s, a; \theta_i)$ is the predicted Q-value for the current state-action pair using online network parameters θ_i. γ is the discount factor for future rewards.

When selecting actions, if the action directly output by the model conflicts with the constraints of the dispatch interval, we will update the actual action value. For example, if $a = (1, 0), I_m^{up} < T_{min}, I_m^{down} > T_{max}$, we will update the action to $a = (0, 1)$, which means only the downward direction is a departure. In addition, we use the ϵ-greedy algorithm to allow the DQN agent to explore more quickly. The pseudo-code of the learning process is shown in Algorithm 1.

As we stipulate that there must be a departure at the last departure time, and the departure timetable needs to meet the constraints of the maximum and minimum departure intervals, the final results we get from using DQN often result in the number of departures differing by one between the upward and downward directions. Therefore, during the testing process, we make some adjustments to the last few buses to achieve a consistent number of departures in both directions. We achieve this by deleting the penultimate departure time. Then, with the T_{max} as the interval, we adjust the departure time of the bus forward until no adjustment is needed, that is, meets the departure interval constraint.

5 Experimental Results

5.1 Real-world Dataset

We used the card-swiping data of passengers on lines 208 and 211 in a China city on August 14, 2023. Each record of data includes the unique identification code of the passenger, the boarding station, the destination station, and the boarding time.

To estimate the waiting time of passengers, we use a normal distribution to infer the time when passengers arrive at the boarding station. For this, we used the time data of buses entering and leaving each bus station on the line that day, which includes when a bus enters and leaves each station. We estimate the time when passengers arrive at the boarding station by using the normal distribution, with the mean being the time when passengers get on the bus, and the standard deviation being half of the difference between the time when passengers get on the bus and the time when the previous bus leaves. The number of bus seats is

32 for these lines and α is set to 1.5. The information of the dataset is shown in Table 1.

Table 1. Bus lines information

Lines	Direction	Service time	Stations	Records
208	Upward	6:00-21:00	26	3157
208	Downward	6:00-21:00	24	2604
211	Upward	6:00-22:00	17	2266
211	Downward	6:00-22:00	11	2126

5.2 Baseline Algorithms

DRL-TSBC is an algorithm that performs real-time scheduling in two directions simultaneously. For offline scheduling comparison, we used the manual scheduling scheme currently in operation on the bus line. This real-world scheme also has the same number of departures in both directions. In addition, we also compare with the DRL-TO algorithm [7], which uses the DRL algorithm to schedule in a single direction. We will analyze whether it can achieve the same number of departures in both directions.

For online scheduling, Currently, there is no other algorithm that can perform online scheduling in both directions simultaneously and achieve the same number of departures in both directions. Therefore, we will only show the ability of DRL-TSBC to respond to real-time passenger flow changes when applied online.

5.3 Experimental Settings

The structure of the DQN network is a fully connected network [22]. Specifically, the network comprises 12 hidden layers, each with 500 hidden units, utilizing ReLU as the activation function and a learning rate of 0.001. Hyperparameters of DQN are shown in Table 2. All methods were tested on a laptop equipped with an AMD Ryzen 9 7945HX CPU, an NVIDIA RTX 4060 Laptop GPU, and 32GB of RAM. DRL-TSBC and DRL-TO were implemented in Python using the Pytorch library.

Table 2. Hyperparameters of DQN

Parameters	B	γ	M	ϵ	E	P	O
Values	64	0.4	3000	0.1	50	5	100

5.4 Experimental Results of Offline Scheduling

Table 3. Experimental results of DRL-TSBC, DRL-TO, and the Manual timetable

Lines		208		211	
Direction		Up	Down	Up	Down
Manual	NDT	72	72	76	76
	AWT(m)	4.06	4.6	4.68	3.5
	NSP	3	176	0	0
DRL-TO	NDT	69	72	75	76
	AWT(m)	3.7	3.2	3.6	3.1
	NSP	0	7	0	0
	ω	1/1000	1/500	1/500	1/300
DRL-TSBC	NDT	73	73	75	75
	AWT(m)	3.7	3.8	4.0	3.3
	NSP	0	0	0	0
	ω	1/1000		1/900	

Table 3 displays the performance of bus timetables scheduled by the Manual method, DRL-TO, and DRL-TSBC. We assessed the number of departure times (NDT) in the timetable, the average waiting time (AWT) of passengers, and the number of stranded passengers (NSP). NDT and NSP are two contradictory indicators, more detail will be discussed in section 5.6. By changing ω in Eq. (4) and Eq. (5), we obtained scheduling results in which the number of departures is close to the manual methods.

Since DRL-TO cannot achieve consistent scheduling of departures for both up and down directions, we change ω to make it as consistent as possible with the number of departures, while being close to the manual method. In real-world bus systems, trying to find two ω that make the number of departures for both up and down directions consistent through multiple pieces of training of different parameters ω is time-consuming and unrealistic.

Table 3 shows that, when the number of departures is close, both DRL-TO and DRL-TSBC can better arrange the timetable to achieve a shorter average passenger waiting time, and almost no passengers will be stranded.

Compared to DRL-TO, the timetable of DRL-TSBC slightly increased the average waiting time of passengers. This is because when designing the reward function, DRL-TSBC introduced a reward for maintaining the consistency of the number of departures in both directions. However, it still has a significant improvement over the manual method, and the slight increase in the average passenger waiting time to achieve consistency in the number of departures in both directions is of greater significance.

Fig. 1 compares the real need of passengers with the carrying capacity provided by bus schedules generated by different methods. We define the real need of passengers as how many passengers are currently on the bus, and the carrying

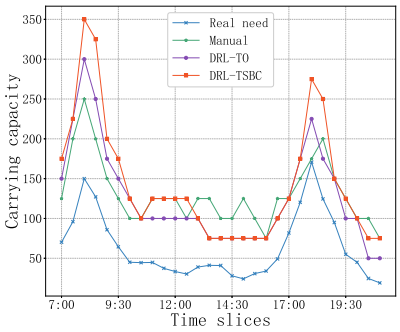

Fig. 1. Comparison of carrying capacity between bus timetables generated by different methods in the upward direction of line 208.

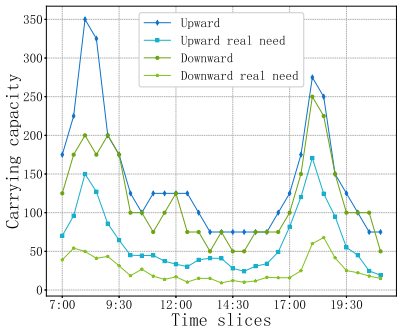

Fig. 2. Comparison of the carrying capacity provided by the DRL-TSBC bus schedule for the upward and downward directions of line 208 with the real need.

capacity provided by the schedule as how many seats the buses currently on the route can provide in total. It can be seen that DRL-TSBC and DRL-TO provide more carrying capacity than the manual method during peak hours, that is, the departures are more frequent. During off-peak hours, it provides less passenger capacity than the manual method, that is, the departure frequency is lower. This also explains why DRL-TSBC can achieve a shorter average waiting time for passengers. In the four tested directions, DRL-TSBC has reduced the average passenger waiting time by 12.1% compared to the manual method and effectively reduced the number of stranded passengers.

Fig. 2 shows the comparison between the scheduling results of DRL-TSBC for the up and down directions of line 208 and the real need. It can be seen that DRL-TSBC can well meet the real needs of passengers in both directions and achieve an equal number of departures in both directions when there is a difference in real needs in the two directions.

5.5 Experimental Results of Online Scheduling

The online scheduling of DRL-TSBC uses a trained model to generate real-time bus timetables. DRL-TSBC will adjust according to the changing passenger flow, obtain a departure timetable schedule that matches the passenger flow situation, and ensure that the number of departures in both upward and downward directions is equal.

To simulate the sudden increase in passenger flow caused by unexpected events, if we simulate the situation of the evening peak hours advancing, the scheduling results obtained by DRL-TSBC are shown in Fig. 3. It can be seen that as the evening peak hours advance, DRL-TSBC provides more departures

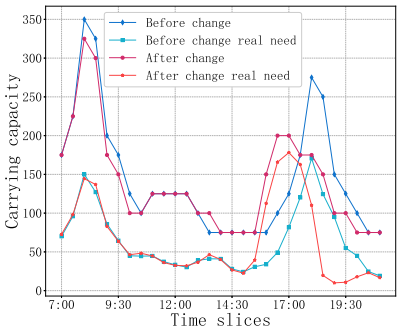

Fig. 3. Comparison of the scheduling results and actual demand before and after advancing the peak evening passenger flow in the upward direction, line 208.

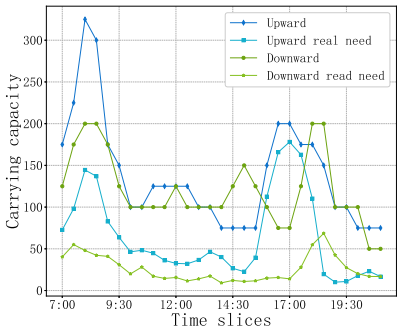

Fig. 4. Scheduling results of upward and downward directions after advancing the peak evening passenger flow in the upward direction, line 208.

earlier to cope with this change. Fig. 4 shows the scheduling results of the downward direction after the evening peak hours of the upward direction are advanced. It can be seen that DRL-TSBC's scheduling of the downward direction is not affected by the changes in the upward direction. The number of departures in both directions obtained by DRL-TSBC is 75, which is still consistent.

Online scheduling experiments show that DRL-TSBC can dynamically meet passenger demand while achieving consistent departure times for both upward and downward directions.

5.6 Analysis of Performance with Different ω

This section shows the performance evaluation of DRL-TSBC across different values of ω and evaluates whether it can achieve an equal number of departures in both upward and downward directions. We trained and tested lines 208 and 211 with different values of ω, and the test results are shown in Table 4. We can see that when ω is different, DRL-TSBC can always achieve the same number of departures in both upward and downward directions. And when ω decreases, the number of departures in both directions will decrease at the same time and remain consistent. Given that the average waiting time for passengers and the number of departures represent conflicting objectives, we can see that the average waiting time for passengers increases as ω decreases.

This means that while ensuring the number of departures in both upward and downward directions is equal, we can adjust ω to prioritize either the cost of the bus company or the experience of the passengers.

Table 4. Performance of DRL-TSBC under different values of ω in line 208 and 211.

	Line 208			Line 211		
ω	NDT	AWT upward	AWT downward	NDT	AWT upward	AWT downward
1/500	89	2.6	2.5	77	3.4	3.1
1/1000	77	3.5	3.3	68	4.8	3.9
1/2000	72	3.8	4.0	58	6.7	5.8
1/3000	70	4.7	4.2	54	7.9	6.5
1/4000	64	4.9	4.9	52	7.8	6.5

6 Conclusion

To achieve dynamic bus scheduling and maintain a consistent number of departures in both upward and downward directions, we proposed a method for dynamic bus timetable scheduling based on deep reinforcement learning. We modeled the bidirectional bus scheduling as a Markov Decision Process (MDP), using a Deep Q-network (DQN) agent to decide whether to depart in both directions. We constructed a state space that included the current time, full load rate, and the number of departures in both directions. We also designed a reward function that included capacity utilization rate, the average waiting time of passengers, and the difference in the number of departures between the upward and downward directions.

Experiments on real data showed that DRL-TSBC could effectively implement dynamic bus timetable scheduling with bidirectional constraints. Compared with other methods, DRL-TSBC could effectively ensure the consistency of the number of departures in both upward and downward directions while reducing the average waiting time of passengers.

References

1. Wang, C., Guo, C., Zuo, X.: Solving multi-depot electric vehicle scheduling problem by column generation and genetic algorithm. Applied Soft Computing **112**, 107774 (2021)
2. Parbo, J., Nielsen, O.A., Prato, C.G.: User perspectives in public transport timetable optimisation. Transportation Research Part C: Emerging Technologies **48**, 269–284 (2014)
3. de Grange, L., Troncoso, R., Briones, I.: Cost, production and efficiency in local bus industry: An empirical analysis for the bus system of santiago. Transportation Research Part A: Policy and Practice **108**, 1–11 (2018)
4. Ceder, A., Golany, B., Tal, O.: Creating bus timetables with maximal synchronization. Transportation Research Part A: Policy and Practice **35**(10), 913–928 (2001)
5. Wihartiko, F., Buono, A., Silalahi, B.: Integer programming model for optimizing bus timetable using genetic algorithm **166**(1), 012016 (2017)

6. Tang, J., Yang, Y., Hao, W., Liu, F., Wang, Y.: A data-driven timetable optimization of urban bus line based on multi-objective genetic algorithm. IEEE Transactions on Intelligent Transportation Systems **22**(4), 2417–2429 (2020)
7. Ai, G., Zuo, X., Chen, G., Wu, B.: Deep reinforcement learning based dynamic optimization of bus timetable. Applied Soft Computing **131**, 109752 (2022)
8. Zhao, Y., Chen, G., Ma, H., Zuo, X., Ai, G.: Dynamic bus holding control using spatial-temporal data–a deep reinforcement learning approach pp. 661–674 (2022)
9. Tišljarić, L., Carić, T., Abramović, B., Fratrović, T.: Traffic state estimation and classification on citywide scale using speed transition matrices. Sustainability **12**(18), 7278 (2020)
10. Ceder, A., Tal, O.: Timetable synchronization for buses. In: Computer-Aided Transit Scheduling: Proceedings, Cambridge, MA, USA, August 1997, pp. 245–258. Springer (1999)
11. Luhua, S., Yin, H., Xinkai, J.: Study on method of bus service frequency optimal modelbased on genetic algorithm. Procedia Environmental Sciences **10**, 869–874 (2011)
12. Gkiotsalitis, K., Alesiani, F.: Robust timetable optimization for bus lines subject to resource and regulatory constraints. Transportation Research Part E: Logistics and Transportation Review **128**, 30–51 (2019)
13. Yu, B., Yang, Z., Yao, J.: Genetic algorithm for bus frequency optimization. Journal of Transportation Engineering **136**(6), 576–583 (2010)
14. Ceder, A.: Bus frequency determination using passenger count data. Transportation Research Part A: General **18**(5-6), 439–453 (1984)
15. Shang, H.Y., Huang, H.J., Wu, W.X.: Bus timetabling considering passenger satisfaction: An empirical study in beijing. Computers & Industrial Engineering **135**, 1155–1166 (2019)
16. Yan, H., Cui, Z., Chen, X., Ma, X.: Distributed multiagent deep reinforcement learning for multiline dynamic bus timetable optimization. IEEE Transactions on Industrial Informatics **19**(1), 469–479 (2022)
17. Li, J., Dong, H., Zhao, X., Tao, L., Liang, C., Zhang, Y.: Practical bus timetable optimization method based on deep reinforcement learning. In: 2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST). pp. 581–587. IEEE (2022)
18. Liu, Y., Zuo, X., Ai, G., Liu, Y.: A reinforcement learning-based approach for online bus scheduling. Knowledge-Based Systems **271**, 110584 (2023)
19. Liu, Y., Zuo, X., Li, X., Nie, S.: A genetic algorithm with trip-adjustment strategy for multi-depot electric bus scheduling problems. Engineering Optimization pp. 1–20 (2023)
20. Matthews, R.A.: The science of murphys law. Scientific American **276**(4), 88–91 (1997)
21. Gao, P., Zuo, X., Bian, Q., Zhao, X.: A memetic algorithm to optimize bus timetable with unequal time intervals. In: Proceedings of the genetic and evolutionary computation conference companion. pp. 1336–1344 (2019)
22. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)