Knowledge-Based Systems 271 (2023) 110584

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A reinforcement learning-based approach for online bus scheduling

Yingzhuo Liu, Xingquan Zuo *, Guanqun Ai, Yahong Liu

School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, China

ARTICLE INFO

Article history:
Received 19 December 2022
Received in revised form 15 April 2023
Accepted 16 April 2023
Available online 21 April 2023

Dataset link: https://github.com/BUPTAIOC/Transportation

Keywords:
Bus scheduling
Online scheduling
Deep reinforcement learning

ABSTRACT

Bus Scheduling Problem (BSP) is vital to save operational cost and ensure service quality. Existing approaches typically generate a bus scheduling scheme in an offline manner and then schedule vehicles according to the scheme. In practice, uncertain events such as traffic congestion occur frequently, which may make the originally planned bus scheduling scheme infeasible. This study proposes a Reinforcement Learning-based Bus Scheduling Approach (RL-BSA) for online bus scheduling. In RL-BSA, each departure time in a bus timetable is regarded as a decision point, and an agent makes a decision at the departure time to select a vehicle to depart at the time. The BSP is modeled as a Markov Decision Process (MDP) for the first time in literature. The state features are devised, which consist of real-time information of vehicles, including remaining working time, remaining driving time, rest time, number of executed trips and vehicle type. A reward function combining a final reward and a step-wise reward is devised. An invalid action masking approach is used to avoid the agent from selecting vehicles not meeting constraints. The agent is trained by interacting with a simulation environment and then the trained agent can schedule vehicles in an online manner. Experiments on real-world BSP instances show that RL-BSA can significantly reduce the number of vehicles used compared with the manual scheduling approach and Adaptive Large Neighborhood Search (ALNS). Under uncertain environment, RL-BSA can cover all departure times in the timetable without increasing the number of vehicles used.

1. Introduction

Bus Scheduling Problem (BSP) refers to arranging for vehicles to perform planned trips in a bus timetable to minimize operational cost [1] and ensure service quality. A bus line has a timetable with a large number of fixed departure times, each of which corresponds to a planned trip. Current approaches typically consider BSP as an optimization problem, and adopt exact approaches [2–4] or heuristics [1,5,6] to generate bus scheduling schemes in an offline manner. In practice, uncertain events (e.g. abnormal weather, traffic accidents and traffic congestion) occur frequently. They may enlarge the travel time of some trips of vehicles and make some vehicles unable to arrive at the destination as planned, which makes the planned bus scheduling scheme infeasible.

The approaches for uncertain BSP can be divided into two categories, i.e., rescheduling approaches [7–9] and robust ones [10–12]. Rescheduling approaches handle uncertain events by regenerating a new bus scheduling scheme. They will regenerate scheduling schemes frequently when uncertain events often occur, which will increase the operational cost. In addition, rescheduling needs to be carried out in a real-time manner, i.e., a bus scheduling scheme must be regenerated in a very short time, which is a challenging task for a large-scale BSP. Robust scheduling approaches improve the robustness of a bus scheduling scheme to uncertain events by sacrificing its optimality, which will decrease the utilization rate of vehicles and increase operational cost.

Reinforcement Learning (RL) [13,14] is an approach which continuously improves the decision-making ability through trial and error. Some studies adopted RL to solve optimization problems, such as bus timetable optimization [15], vehicle routing problem [16] and job shop scheduling problem [17]. Compared with traditional approaches [18–20] for such problems, RL obtains a policy by interacting with the environment, and then uses the policy to make real-time decisions to obtain a solution to an optimization problem. Since RL can generate a solution through real-time decision-making, it is more suitable for uncertain optimization problems.

It is a challenging task to apply RL to BSP because BSP needs to be modeled as a Markov Decision Process (MDP) [21] and a suitable RL agent model must be constructed. This study proposes a Reinforcement Learning-based Bus Scheduling Approach (RL-BSA) for a single-depot BSP. In RL-BSA, the BSP is regarded as a sequential decision-making problem. Each departure time in the timetable is a decision point, and RL agent makes a decision

* Corresponding author at: School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China.
E-mail address: zuoxq@bupt.edu.cn (X. Zuo).

https://doi.org/10.1016/j.knosys.2023.110584
0950-7051/

at each departure time to select a vehicle to depart at the time based on the vehicles' real-time information. After decisions are made for all departure times in the timetable, a complete bus scheduling scheme is generated. The state of the next departure time (decision point) is only related to the state of current departure time, not to previous states. Therefore, the BSP has Markov property [21] and can be modeled as a MDP.

To model the BSP as a MDP, the state space, action space and reward function need to be devised. In RL-BSA, each action is to select a vehicle to perform the planned trip corresponding to the current departure time (decision point). The state features contain the information of vehicles, including remaining working time, remaining driving time, rest time, number of trips, and vehicle type. A reward function [22,23] combining a final reward and a step-wise reward is devised. The final reward is provided at the last decision point of the whole episode, while the step-wise reward is given at every decision point. Double Dueling Deep Q-learning (D3QN) [24] is used as the RL agent. To ensure the agent selects the vehicle satisfying constraints of the BSP, a mask layer [25] is added to the end of the neural network of D3QN.

To our knowledge, there are no studies on using RL to solve BSP to date. This paper tends to make the following contributions:

(1) A Reinforcement Learning-based Bus Scheduling Approach (RL-BSA) is proposed for the first time. The BSP is regarded as a sequential decision-making problem, and each departure time in the timetable is regarded as a decision point. RL agent (D3QN) is used to select a vehicle to perform the planned trip corresponding to the current departure time in the timetable. A mask layer is added to the neural network of D3QN to avoid selecting vehicles not satisfying constraints of the BSP.

(2) The BSP is modeled as a MDP. Some new state features and actions are specifically devised for the BSP. A reward function is devised, which includes a final reward and a step-wise reward.

(3) RL-BSA is applied to real-world BSP instances, and experiments show that it outperforms the manual scheduling approach and Adaptive Large Neighborhood Search (ALNS) [26]. Under uncertain environment, RL-BSA can cover all departure times in a timetable (i.e., ensure the service quality) without increasing the number of vehicles used (i.e., without increasing the operational cost).

The remainder of this paper is organized as follows: Section 2 reviews the related work. Section 3 introduces the BSP. In Section 4, the reinforcement learning based bus scheduling approach is presented. Section 5 gives experimental results. Finally, conclusions are drawn in Section 6.

2. Related works

Bus scheduling problem is a complex optimization problem. Existing solution approaches can be divided into two categories: scheduling approaches under static environment and scheduling approaches under uncertain one.

2.1. Bus scheduling approaches under static environment

Most of solution approaches for BSP do not consider uncertain events that may occur during bus scheduling procedure. Those approaches can be divided into exact approaches and heuristics.

There are many studies using exact approaches to solve BSP under static environment. With the objective of minimizing operational cost, Freling et al. [2] proposed a linear program combined with an auction-based-algorithm for a single-depot BSP. Riberiro et al. [3] proposed an approach based on a column generation algorithm to solve a multi-depot BSP, which is modeled as a network flow model. To minimize the fleet size and operational cost, Kliewer et al. [4] used a temporal–spatial network to model a multi-depot BSP, and the model was solved by an optimization software package. To reduce operational cost and vehicle exhaust emissions, Li et al. [27] modeled the BSP as a temporal–spatial network, and applied CPLEX to solve it. Haase et al. [28] established a set-partition model to schedule vehicles and crews simultaneously, with the objective of minimizing operational and fixed cost. The model was solved by a branch-and-bound algorithm integrating a column generation algorithm. Lin et al. [29] proposed a programming model solved by a branch-and-bound algorithm to minimize the numbers of vehicles and drivers. Hadjar et al. [30] used a branch-and-bound algorithm to solve a multi-depot BSP with time windows, and proposed a time windows reduction approach to speed up the search process. Adler et al. [31] proposed an integer programming model for a multi-depot BSP with new energy vehicles, and used a branch-and-price algorithm to solve it. Boyer et al. [32] proposed a variable neighborhood search algorithm to schedule vehicles and crews simultaneously. Gkiotsalitis et al. [33] proposed a mixed-integer nonlinear programming model for a multi-depot electric BSP with time windows. Further, they linearized the formulation of the problem by reformulating it as a mixed-integer linear program that can be solved optimally. Janovec et al. [34] proposed a linear programming model for electric BSP and used a standard integer programming solver to solve it.

Many heuristics have been proposed to solve BSP under static environment. Freling et al. [1] proposed an approach combining a column generation algorithm and a Lagrangian relaxation algorithm to simultaneously schedule vehicles and crews. Zuo et al. [5] proposed an improved nondominated sorting genetic algorithm II (NSGA-II) combined with a departure times adjustment procedure (DTAP) to minimize the number of drivers and vehicles. Shui et al. [6] combined a clonal selection algorithm with the DTAP to solve BSP. For electric BSP, Wen et al. [26] proposed a mixed integer programming formulation as well as an adaptive large neighborhood search (ALNS) heuristic. To minimize the number of vehicles used, Ceder et al. [35] proposed a deficit-function-based heuristic to solve a multi-depot BSP. Kulkarni et al. [36] proposed a column generation and three heuristics to solve a multi-depot BSP. To solve a multiple-type electric BSP, Yao et al. [37] proposed a heuristic approach to generate a scheduling scheme to arrange recharging trips and departure times for each bus. Duan et al. [38] proposed a grouping coding genetic algorithm combined with a Monte Carlo approach to generate a scheduling scheme to reduce operational cost. To simultaneously optimize bus timetabling and schedule electric buses, Teng et al. [39] proposed a multi-objective particle swarm optimization algorithm to get a set of Pareto-optimal solutions. Zhao et al. [40] proposed a tabu search algorithm for a multi-line BSP with a single depot. Yang et al. [41] proposed a bi-objective optimization model for regional BSP to minimize the total waiting cost of passengers and maximize the comprehensive service rate of buses. NSGA-II is adopted to obtain a Pareto solution set for such problem. Carosi et al. [42] proposed a multi-commodity flow type model and a diving-type matheuristic approach for bus timetable optimization and BSP. Wang et al. [43] studied a multi-depot electric BSP and proposed a genetic algorithm based column generation approach to solve it. Liu et al. [44] proposed a two-stage solution approach combining a simulated annealing and a local search for an electric BSP. Liu et al. [45] further proposed a construction-and-repair based method to solve a BSP with branch lines.

2.2. Bus scheduling approaches under uncertain environment

Scheduling approaches under static environment generate a scheduling scheme in an offline manner, and cannot adjust the

scheme during the scheduling procedure. Uncertain events, such as traffic congestion and vehicle breakdown, may occur during the implementation of the scheduling scheme, resulting in the scheme becoming infeasible. Solution approaches for uncertain BSP can be divided into rescheduling approaches and robust scheduling ones.

Rescheduling approaches regenerate a new scheduling scheme to replace the old one. Huisman et al. [7] proposed a cluster rescheduling heuristic for a BSP with uncertain travel time. Wang et al. [8] used NSGA-II to regenerate a scheduling scheme to deal with traffic congestion. When traffic congestion occurs, a set of vehicle blocks is reconstructed, which is used to regenerate a set of solutions. Shen et al. [9] proposed a scheduling approach based on a hierarchical task network and designed two dynamic bus scheduling strategies. The first one reschedules individual vehicle independently, and the second one reschedules multiple vehicles simultaneously. A robust rescheduling and holding approach [46] and a sequential hill climbing approach [47] were proposed to reschedule departure times of the trips to cope with uncertain events. Li et al. [48] developed a sequential and parallel auction algorithm to solve an uncertain BSP. To minimize operational and delay cost of uncertain events, Li et al. [49] proposed a prototype decision support system that recommended solutions for a single-depot rescheduling problem. Approaches for real-time vehicle schedule recovery in public transportation services were reviewed in [50]. Tang et al. [51] proposed a static model with buffer-distance strategy to tackle adverse impacts caused by trip time stochasticity, and used a dynamic model to periodically reschedule an electric bus fleet during a day's operation.

Robust scheduling approaches can generate a scheduling scheme that is robust to uncertain events. To optimize planned cost and cost caused by disruptions, Naumann et al. [10] modeled the BSP as a temporal–spatial network, and presented a stochastic programming model for robust BSP. To enhance the schedule's robustness, Shen et al. [11] redefined the compatibility of any pair of trips and proposed a network flow model with stochastic trip time. To minimize the sum of the expected value of the random schedule deviation, Yan et al. [12] established a robust optimization model for a BSP with uncertain travel time, and a Monte Carlo simulation was used to solve the model.

In summary, scheduling approaches under static environment generate a bus scheduling scheme in advance. The generated scheduling scheme cannot ensure its scheduling performance or even becomes infeasible under uncertain environment. To handle uncertain events, rescheduling approaches regenerate a new bus scheduling scheme to replace the old one while robust scheduling approaches generate a bus scheduling scheme, which is robust to uncertain events. Rescheduling approaches need to update the bus scheduling scheme frequently when uncertain events often occur, which cannot ensure the scheduling performance and may increase the operational cost. Robust scheduling approaches generate a robust bus scheduling scheme via overestimating the travel time or distance of trips, which may decrease the utilization rate of vehicles and increase operational cost.

This study models the BSP as a MDP for the first time in literature, and a reinforcement learning based scheduling approach is proposed to solve the problem. In practice, a planned vehicle scheduling scheme needs to be made in advance to schedule drivers since driver scheduling is based on a vehicle scheduling scheme. If there is no such a complete vehicle scheduling scheme, it is impossible to know how many vehicles are used, how many trips each vehicle performs, which vehicle a driver should drive and when a driver should go to work. Therefore, RL-BSA generates a bus scheduling scheme using RL agent trained offline. Once uncertain events occur, RL agent can schedule vehicles in an online (real-time) manner according to the real-time information of vehicles, to make each planned trip in the timetable be executed by exactly one vehicle and meanwhile minimize the number of vehicles used.

3. Bus scheduling problem

BSP [1] aims to assign a fleet of vehicles to all planned trips corresponding to departure times in a fixed bus timetable, to make each planned trip be executed by exactly one vehicle. Each planned trip in the timetable has a fixed departure time and an arriving time. The optimization objective of BSP is to minimize the number of vehicles used and simultaneously satisfy constraints of BSP.

The BSP studied in this paper has a bus line with two Control Points (i.e., CP_1 and CP_2), each of which has a timetable (i.e., T_1 and T_2). Each timetable contains a number of fixed departure times, i.e., $T_1 = \left\{t_1^1, t_1^2, t_1^3, \ldots, t_1^{N1}\right\}$ and $T_2 = \left\{t_2^1, t_2^2, t_2^3, \ldots, t_2^{N2}\right\}$. The timetables of two CPs are integrated into one timetable T, and departure times in T are sorted in chronological order. Each departure time in T corresponds to a planned trip.

A trip of a vehicle is a directed route of a vehicle from one CP to the other. Fig. 1 shows all trips of a vehicle v. Each trip i of vehicle v has a departure time d_i^v and an arriving time a_i^v, such that its travel time is $h_i^v = a_i^v - d_i^v$. The interval between the arriving time of the ith trip and the departure time of the $(i+1)$th trip is rest time $r_i^v = d_{i+1}^v - a_i^v$. A vehicle block is composed of all the trips of vehicle v in a day, as shown in Fig. 1. Suppose that the vehicle block contains N^v trips, then the total driving time of vehicle v is $H^v = \sum_{i=1}^{N^v} h_i^v$, total rest time is $R^v = \sum_{i=1}^{N^v} r_i^v$, and total working time is $W^v = H^v + R^v$. Two vehicle types are commonly used in practice, i.e., long vehicle and short one. The maximum working time of a long vehicle is twice that of a short one.

Constraints of the BSP are as follows:

(1) Trip i of vehicle v must be completed before its next trip $(i+1)$.

(2) After finishing trip i, vehicle v must have a rest time not less than r_{min}, i.e., $r_i^v \geq r_{\min}$.

(3) The total driving (working) time of vehicle v should not exceed its maximum driving (working) time. If vehicle v is a short vehicle, its maximum driving (working) time is $T_H(T_W)$; If vehicle v is a long vehicle, its maximum driving (working) time is $2T_H(2T_W)$. This constraint is a soft constraint and actual working time of a vehicle is allowed to be a little longer than the maximum working time.

The optimization objectives of the BSP are as follows:

(1) Minimize the number of vehicles used, N_u.

(2) Each departure time in timetable T must be executed by exactly one vehicle, i.e., the number of departure times in T, $|T|$, is equal to the total number of trips of all vehicles, $\sum_{v=1}^{N_u} N^v$.

(3) Minimize the number of vehicles with odd trips, N_o, to allow drivers to start and finish work at the same CP.

4. Reinforcement learning-based bus scheduling approach

In RL-BSA, the BSP is modeled as a MDP. Each departure time in timetable T is regarded as a decision point. A RL agent makes a decision at each departure time (decision point) in the timetable according to the real-time information of vehicles, to select a vehicle to depart at the departure time (i.e., cover the departure time).

All departure times in timetable T need to be covered by a fleet of vehicles. Those vehicles travel between the two CPs to cover all departure times in T. For each departure time t_c in the timetable of a CP, the CP may have one or more vehicles that can be selected to depart at time t_c. The information of each vehicle v can be easily observed at time t_c, including its rest time r_i^v, accumulated driving time h^v and number of executed trips n^v. An available vehicle refers to the vehicle that has returned to the CP at time t_c, has a rest time larger than r_{min}, and has accumulated driving time less than its maximum driving time. Obviously, only

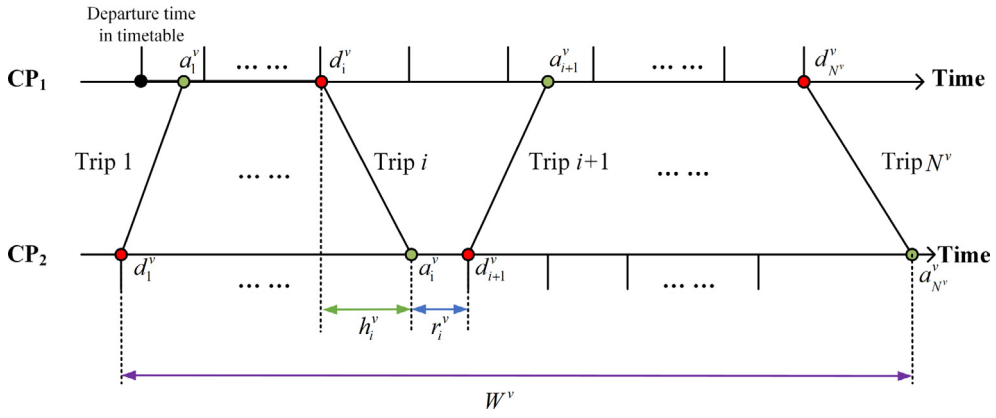


Fig. 1. A vehicle block with multiple trips.

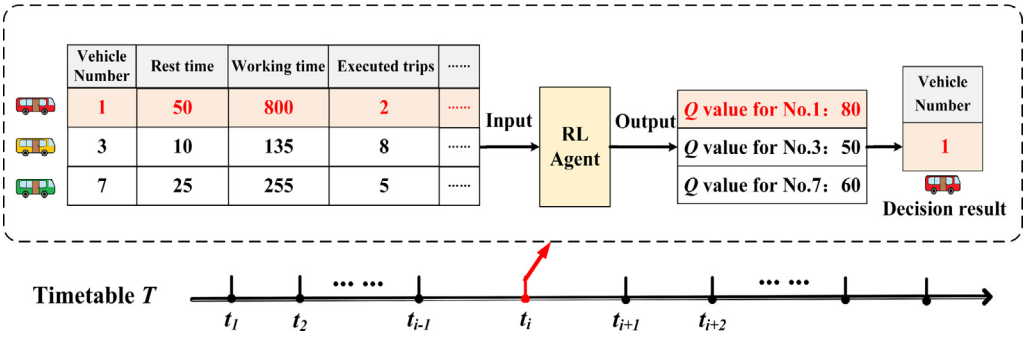


Fig. 2. An example of an agent selecting a vehicle based on real-time information.

available vehicles can be selected to depart from the CP at time t_c. The RL agent makes a decision according to the information of all available vehicles to select an available vehicle to depart from the CP at time t_c. By this way, RL agent makes a decision for each departure time in timetable T to cover the departure time. After RL agent makes decisions for all departure times, a bus scheduling scheme is formed.

Fig. 2 shows an example of decision-making process of RL agent. For departure time t_i in the timetable of a CP, there are 3 available vehicles in the CP, namely vehicle 1, 3 and 7. The RL agent witnesses the information of all available vehicles, and then outputs an action value (Q value) for each vehicle. The vehicle with the largest Q value is selected to cover departure time t_i. Q value refers to the expected value of accumulated reward obtained in the subsequent decision sequence after the agent conducts an action, and is used to evaluate the value of the action.

In the example shown in Fig. 2, vehicle 1 has the longest rest time, longest accumulated working time, and largest number of remaining trips. If vehicle 1 is not selected immediately, it will not have enough time to perform its remaining trips, resulting in low vehicle utilization. Therefore, vehicle 1 has the largest Q value 80, such that vehicle 1 is selected to cover departure time t_i.

The framework of RL-BSA is shown in Fig. 3. The environment is mimicked by a real-time simulator of bus scheduling, which can update the vehicle information in real-time according to the decisions of the agent. The agent (Section 4.2) outputs the Q values of all actions (Section 4.1.2) according to the state (Section 4.1.1) obtained from the environment. A mask layer of the agent is used to filter out the unavailable actions, and then the action with the largest Q value is selected. After the action (i.e., selecting a vehicle to cover the current departure time) acts on the environment, a new state will be observed from the environment. Meanwhile, an immediate reward (Section 4.1.3) is provided to the agent. The agent repeats the above process to interact with the simulation environment to accumulate data for training. A reinforcement learning algorithm (Section 4.2.3) is used to train the agent to improve its decision-making ability.

4.1. MDP model of bus scheduling problem

The BSP is modeled as a MDP. State space, action space and reward function of the MDP are designed as follows.

4.1.1. State space

State s contains the information that the agent observes from the environment. As shown in Fig. 2, RL agent makes a decision to select a vehicle based on the information of all available vehicles, such that the state includes the information of each available vehicle.

For each available vehicle v, we devise the following 5 features that are the most related to the scheduling of vehicles:

(1) Number of executed trips n^v: It indicates whether vehicle v has an even number of trips. A vehicle with an even number of trips means that the driver starts and finishes work at the same CP.

(2) Remaining working time w_r^v: For vehicle v, up to the current departure time (decision point), suppose that its accumulated driving time is h^v, accumulated rest time is r^v, and accumulated working time is w^v, where $w^v = h^v + r^v$. The remaining working time of vehicle v is $w_r^v = T_W - w^v$ if v is a short vehicle, and is $w_r^v = (2T_W) - w^v$ if v is a long vehicle, where T_W ($2T_W$) is the maximum working time of a short (long) vehicle. The w_r^v together with the number of executed trips n^v represents the urgency degree of vehicle v completing its remaining trips. If both w_r^v and n^v are small, vehicle v needs to be selected to depart at the current departure time; otherwise, vehicle v may not finish its remaining trips, resulting in a low vehicle utilization.

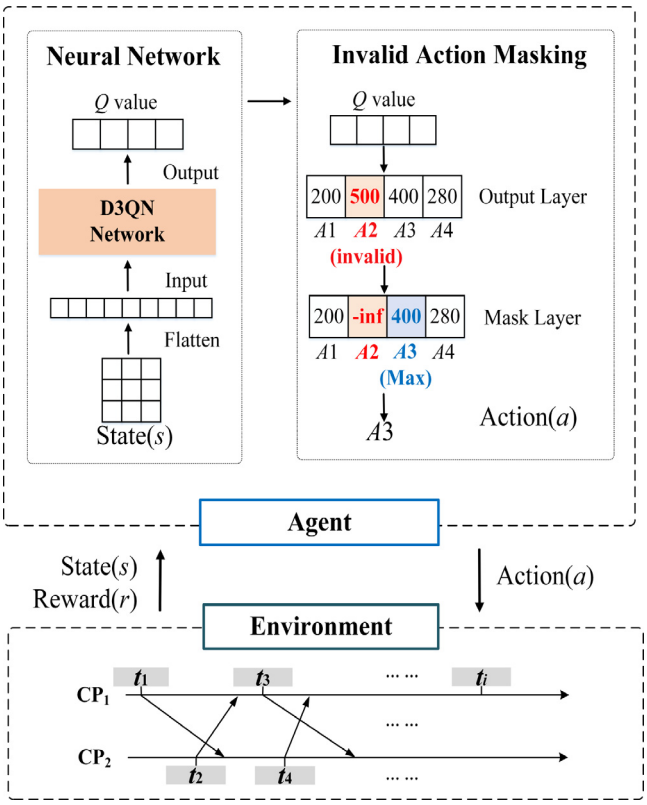


Fig. 3. Framework of RL-BSA.

(3) Remaining driving time h_r^v: $h_r^v = T_H - h^v$ if v is a short vehicle and $h_r^v = (2T_H) - h^v$ if v is a long vehicle. It together with the number of executed trips n^v reflects the urgency degree of vehicle v to complete its remaining trips.

(4) Rest time r^v: r^v is the continuous rest time of vehicle v from the moment of it arriving at the CP. A long rest time of a vehicle will result in a low vehicle utilization. This feature allows the agent to select a vehicle according to its rest time to avoid a long rest time and a low vehicle utilization.

(5) Vehicle type c^v: Two vehicle types are considered, i.e., long and short vehicles. Using more long vehicles can reduce the number of vehicles used.

A state can be represented by a matrix, each row of which contains the above 5 features of a vehicle, as shown in Fig. 4. Suppose that the current decision point (departure time) is t_c. The top N_a rows in the matrix include the features of N_a available vehicles, denoted by a vehicle set V_a. Each vehicle in V_a is an available vehicle and has performed at least one trip before time t_c. Uncertain events, such as traffic congestion, may result in a change in the number of vehicles in V_a. The change of the number of vehicles in V_a reflects uncertain events of the environment. The agent makes a decision (select a vehicle) according to the changed state, to handle uncertain events. What is more, as the number of available vehicles may vary along with the change of decision point (departure time) during the decision process, the number of rows, N_a, in the matrix is not fixed for each decision point.

However, the neural network of RL requires the same input dimensions. To make the matrix have the same number of rows at each decision point, we use features of a new vehicle (i.e., a vehicle that has not started to work before time t_c) and unavailable vehicles to fill the matrix to make it have N rows. N is the estimated maximum number of available vehicles for each decision point. The features of each new vehicle are the same, such that the features of new vehicles can be represented by one row in the matrix. That it, the (N_a+1)th row includes the features of a new vehicle, denoted by vehicle set V_n. There is one new vehicle in V_n only. The remaining rows are filled by the features of $(N-1-N_a)$ unavailable vehicles to make the matrix have the same number rows (N rows) at each decision point. The set of unavailable vehicles is represented by V_u. Each vehicle in V_u is not allowed to be selected by the agent. To distinguish the features of vehicles in V_n and V_u, features of each vehicle in $V_n(V_u)$ are all filled with 0 (-1).

4.1.2. Action space

The action space A is $V_a \cup V_n \cup V_u$. Each action a in A means selecting a vehicle v. If $v \in V_a$, it means that available vehicle v is selected and departs at the current departure time t_c directly. If $v \in V_n$, it means that a new vehicle is used and departs at t_c. The type (long or short vehicle) of the new vehicle is identified randomly. If $v \in V_u$, v is an unavailable vehicle and cannot be selected. A masking approach (Section 4.2.2) is used to avoid the agent conducting an action corresponding to an unavailable vehicle in V_u.

4.1.3. Reward function

The reward function is essential for the agent to learn the policy to make a decision (select a vehicle). There is a large number of departure times in a timetable, and RL agent needs to make a decision for each departure time. Thus, the decision sequence is quite long. In this case, using the final reward only will lead to the sparse reward problem, which makes it difficult for RL algorithm to converge. Hence, this study devises a reward function combining a final reward and a step-wise reward. The final reward is provided at the last decision point of the whole episode, while the step-wise reward is given at every decision point.

(1) Final reward

To minimize the number of vehicles used, N_u is chosen as a part of the final reward. In addition, long vehicles are preferred to improve the vehicles' utilization rate, such that the number of short vehicles, N_s, is used as a penalty term in the final reward.

A driver is usually arranged to start to work at the CP closed to the driver's home. In practice, a vehicle is driven by one driver in a day. Therefore, a vehicle should return to the CP where it departs after completing all its trips in a day, to facilitate the driver going home. That means the number of all trips executed by each vehicle should be an even number. Hence, the number of vehicles with odd number of trips, N_o, is used as a penalty term in the final reward.

To improve the utilization rate of vehicles, the number of full-trip vehicles, N_f, is used as a reward item in the final reward. A full-trip vehicle refers to a long vehicle with the maximum number of trips, N_t, which is calculated by

$$N_t = \begin{cases} \left\lfloor \frac{2T_H}{T_a} \right\rfloor - 1 & \text{if } \left\lfloor \frac{2T_H}{T_a} \right\rfloor \bmod 2 = 0 \\ \left\lfloor \frac{2T_H}{T_a} \right\rfloor & \text{Otherwise} \end{cases} \quad (1)$$

$2T_H$ is the maximum driving time of a long vehicle, and T_a is the average travel time of all trips in timetable T. To ensure that N_t is an even number, 1 is subtracted from $\lfloor \frac{2T_H}{T_a} \rfloor$ if $\lfloor \frac{2T_H}{T_a} \rfloor$ is an odd number. What is more, if the number of executed trips of a short vehicle reaches $N_t/2$, it will be converted to a long vehicle to execute more trips.

To balance the workload of drivers, the driving time of different vehicles should be similar. To do so, the variances of driving time of long vehicles and short vehicles are calculated respectively. The average of them, σ_h, is used as a penalty term in the final reward.

Vehicle	h_r^v	r^v	n^v	w_r^v	c^v
V_a	150	15	2	50	0
	245	10	8	10	1
	360	25	5	25	1
	...	...	...	...	...
V_n	0	0	0	0	0
V_u	-1	-1	-1	-1	-1
	...	...	...	...	...

N_a

N

Fig. 4. State features of the BSP.

The final reward is the weighted sum of the above five items:

$$r_m = -w_1^1 \times N_u - w_2^1 \times N_s - w_3^1 \times N_o + w_4^1 \times N_f - w_5^1 \times \sigma_h \quad (2)$$

where weight values w_1^1, w_2^1, w_3^1, w_4^1 and w_5^1 are all positive real number.

(2) Step-wise reward

A step-wise reward is devised to guide the agent to achieve a large final reward. It serves to reduce the number of vehicles used (vehicles with odd trips) and improve the utilization rate of vehicles.

Using a new vehicle means adding a vehicle to the bus s cheduling scheme, and thus the agent should select vehicles dispatched whenever possible to reduce the number of vehicle used. To guide the agent to select vehicles that have been dispatched and avoid selecting a new vehicle, a penalty r_n occurs if there is no available vehicles in the CP and a new vehicle has to be selected. If a new vehicle is selected in the case that there are available vehicles in the CP, a penalty r_e is appended to r_n.

A long vehicle is preferred to be selected to improve the utilization rate of vehicles. To guide the agent to select a long vehicle, a reward r_l occurs if selecting a long vehicle.

The agent prefers to select a vehicle with a long continuous rest time, r^v, to improve the utilization rate of vehicles. Available vehicles in V_a are sorted in descending order according to their continuous rest time r^v, and p^v represents the rank of vehicle v. The number of vehicles in V_a is N_a. The larger p^v is, the shorter the rest time of vehicle v and the greater the penalty term should be. A penalty term $r_k = p^v/N_a$ occurs if the vehicle with rank p^v is selected.

A penalty term is used to reduce the number of vehicles with odd trips. Suppose that a vehicle v is selected and its remaining driving time after it executes the current trip is not enough to execute the next trip. In this case, a penalty item r_o occurs if the number of trips executed by the vehicle is an odd number.

The step-wise award is the weighted sum of the above five items:

$$r_b = -w_1^2 \times r_n - w_2^2 \times r_e + w_3^2 \times r_l - w_4^2 \times r_k - w_5^2 \times r_o \quad (3)$$

where weight values w_1^2, w_2^2, w_3^2, w_4^2 and w_5^2 are all positive real number.

4.2. Deep reinforcement learning agent

In the MDP model of the BSP, the state space is huge. As a result, it is intractable for a traditional RL algorithm to optimize the policy by all state–action pairs. Deep Q-learning (DQN) [52] is one of classical value-based RL algorithms, which combines neural networks with the traditional RL algorithm to deal with a large state space. DQN utilizes the maximum value of the Q values outputted by its target network to construct training labels, which results in the overestimation of the Q value and the instability of training. D3QN [24] combines the techniques of Dueling Deep Q-learning (Dueling DQN) [24] and Double Deep Q-learning (Double DQN) [53], and can effectively address the overestimation problem of DQN. Thus, we use D3QN as the agent.

4.2.1. Network structure

D3QN contains two same networks: a main network and a target network. The main network is trained to compute Q values of all actions. Parameters of the main network are denoted as θ. The Q value outputted by the main network is determined by state s, action a and parameters θ, and expressed as $Q(s, a, \theta)$. The target network provides training labels for the main network to reduce the correlation among the data. It periodically copies parameters from the main network. Parameters of the target network are denoted as θ^-, and the Q value outputted by the target network is $Q^t(s, a, \theta^-)$. As the structure of the target network is the same as that of the main network, we only introduce the structure of the main network for the BSP.

As shown in Fig. 5, the main network includes three parts: a state network, two pairs of state value network (V network) and advantage value network (A network). The state network is used to extract features of the input. D3QN decomposes the action value (Q value) into a state value (V value) and an advantage value (A value) by

$$Q(s, a, \theta) = V(s, \theta) + A(s, a, \theta) \quad (4)$$

V value is the output of the V network and denotes the overall expected rewards in the future steps, and A value is the output of the A network and denotes the advantage of an action over other actions for the overall expected rewards in the future steps. As the V value is independent of the action a and Q value is related to the action, Q values corresponding to all actions are updated when the V values are updated. For each action in the action space, each pair of V and A networks outputs a Q value and the smaller one is chosen as the output of the main network to mitigate the overestimation of Q value [54].

The input of the main network is the state features of the BSP (shown in Fig. 4) after the flattening operation, and the input dimension is $5N$; The output of the network is a vector with Q values corresponding to all actions, and the output dimension is N. The state network is a two-layer fully connected neural network (State-Net). The output of State-Net is used as the input of two pairs of V and A networks. V1-Net and A1-Net are the first pair of V and A networks and V2-Net and A2-Net are the second one. The network structure of A1-Net (V1-Net) is the same as that of A2-Net (V2-Net). Each V (A) network is a two-layer fully

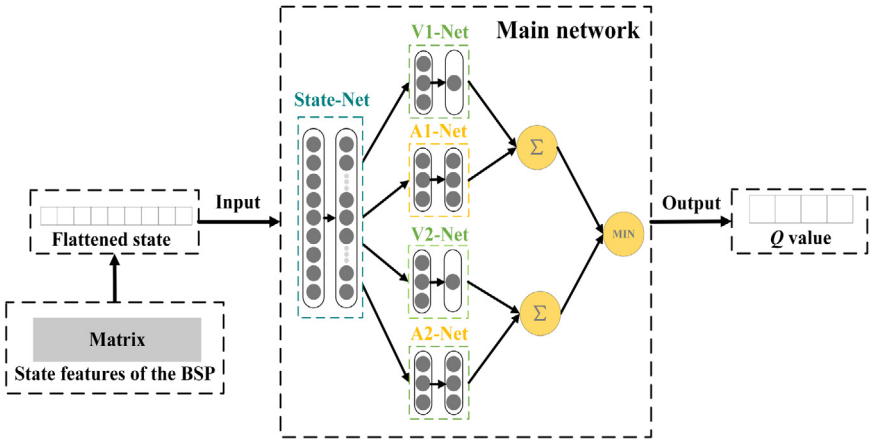


Fig. 5. Main network of D3QN.

Table 1
Operational information of 4 real-world BSP instances.

Bus lines	59	60	803	85
Earliest departure time	6:00	5:50	5:20	5:30
Latest departure time	20:00	22:00	22:00	23:00
Number of departure times in the timetable	104	120	154	170
Average travel time of trips, T_a (min)	41	55	47	35
Maximum working time of short vehicle, T_W (h)	8	8	8	8
Maximum driving time of short vehicle, T_H (h)	6.5	6.5	6.5	6.5
Maximum number of trips of long vehicle, N_t	16	12	16	16
Minimum rest time, r_{min} (min)	3	3	3	3

connected neural network. For each pair of V and A networks, outputs of the two networks are added as Q values, as shown in Eq. (4). The smaller Q values produced by the two pairs of networks are taken as the output of the main network.

4.2.2. Constraints handling

In the action space, vehicles in V_u do not correspond to available vehicles, and thus should not be selected by the agent. In this paper, an invalid action masking approach is employed to avoid selecting the actions in V_u, as shown in Fig. 3. The last layer of the neural network in D3QN outputs Q values of all actions. The Q value of each unavailable vehicle in V_u is assigned a negative value in the invalid action masking layer, so as to avoid being selected. For example, in Fig. 3, action $A2$ has the largest Q value but it corresponds to an unavailable vehicle, such that the Q value of action $A2$ is assigned a negative value. In this case, action $A3$ with second largest Q value is the output of the agent.

4.2.3. Learning process of D3QN

The learning process of D3QN can be divided into two parts: data collection and training.

(1) Data collection: The agent interacts with the simulation environment to collect data using an epsilon-greedy strategy to obtain a set of quadruples (s, a, r, s'), which are stored in the replay buffer.

(2) Training: The main network is trained by the data which randomly sampled from the replay buffer, and the target network will copy parameters from the main network periodically for updating.

The pseudocode of learning process of D3QN is shown in Algorithm 1. The goal of learning is to train a bus scheduling agent to achieve the maximum reward.

The data collection procedure is presented in lines 6 to 21. Firstly, the agent selects action a according to the current state using the epsilon-greedy strategy. Then, the agent applies the action to the environment and obtains a reward r and a new state s'. Secondly, if the amount of data in the replay buffer has reached the maximum capacity, the oldest data will be removed. Thirdly, quadruple (s, a, r, s') is appended to the replay buffer.

The training procedure is presented in lines 22 to 28. Firstly, when the replay buffer has enough samples for at least one mini-batch and the total number of decision-making steps reaches the preset threshold, the agent samples a mini-batch of data from the replay buffer. Secondly, the training label for the Q value outputted by the main network is computed using the target network. The target network generates the Q value and the action is generated from the main network, which can be formulated by $Q^t\left(s', \arg\max_{a'}\left(Q\left(s', a', \theta\right)\right), \theta^-\right)$. The training label can be calculated by

$$r + \gamma Q^t\left(s', \arg\max_{a'}\left(Q\left(s', a', \theta\right)\right), \theta^-\right). \tag{5}$$

Then, in line 24, the loss function is calculated using the label of Q value in Eq. (5). Based on the loss function, parameters of the main network are updated by Adam optimization algorithm, and parameters of the target network are updated according to the updated parameters of the main network.

5. Experimental results

To validate the performance of RL-BSA, it is applied to 4 real-world BSP instances in Qingdao city, China. The operational information of each bus line is shown in Table 1. All data of the 4 instances are available at https://github.com/BUPTAIOC/Transportation. RL-BSA is compared with the manual scheduling approach and ALNS [26].

RL-BSA is coded in python and runs on a PC with Intel (R) Core (TM) i7-8700 CPU, 3.20 GHz, 8.00 GB RAM and NVIDIA GeForce GTX 1650 GPU. The weight values in the final and step-wise rewards are presented in Table 2. Initial hyperparameters of D3QN are presented in Table 3. The main network of D3QN is identical

Table 2
Weight values in the final and step-wise rewards.

Weights	w_1^1	w_2^1	w_3^1	w_4^1	w_5^1	w_1^2	w_2^2	w_3^2	w_4^2	w_5^2
Values	800.0	2000.0	600.0	200.0	400.0	200.0	200.0	80.0	50.0	30.0

Table 3
Hyperparameters of D3QN.

Hyperparameters	M	B	α	γ	tp	N^e	β	ϵ
Values	2^{17}	2^7	0.1	0.99	2^{10}	1000	0.0001	1.0

Table 4
Settings of the main network of D3QN in RL-BSA.

Network	Layer	Number of outputs	Activation function
State-Net	1	80	ReLu
	2	256	–
V1-Net	1	80	ReLu
	2	1	–
V2-Net	1	80	ReLu
	2	1	–
A1-Net	1	80	ReLu
	2	N	–
A2-Net	1	80	ReLu
	2	N	–

Note: '–' means there is no activation function in the corresponding layer.

to the target network. Table 4 presents the number of outputs and activation function of each layer of the components (State-Net, V1-Net, V2-Net, A1-Net and A2-Net) of the main network of D3QN in Fig. 5. The N in Table 4 is the estimated maximum number of available vehicles for each decision point, and is set to be 12, 12, 16, 16 for bus lines 59, 60, 803, 85, respectively.

5.1. Offline optimization of RL-BSA

For each BSP instance, RL agent is trained by interaction with the simulation environment, and then the trained RL agent is used to schedule vehicles in the simulation environment to generate an offline bus scheduling scheme. Table 5 compares the performance of RL-BSA, ALNS and the manual scheduling approach. Each run of the trained RL agent produces the same bus scheduling scheme, such that RL-BSA is run only once. As ALNS is a stochastic algorithm, its each run may obtain different solutions. Thus, ALNS performs 10 independent runs and statistic results are presented in Table 5. Three metrics are used to compare the three approaches: the number of vehicles used N_u, number of vehicles with odd trips N_o, and number of uncovered departure times N_d.

Table 5 shows that RL-BSA can achieve the solution with the smallest N_u for three bus lines. The manual bus scheduling scheme has the smallest number of vehicles N_u for bus line 85. However, the manual bus scheduling scheme violates some constraints in Table 1. The maximum number of trips of long vehicles is 20, which is higher than 16 (the maximum number of trips allowed). For the metric N_o, RL-BSA is slightly better than manual scheduling approach and ALNS. The metric N_d equals 0 for both RL-BSA and ALNS, which means that they can cover all the departure times in the timetable of each bus line.

A bus scheduling scheme generated by RL-BSA for bus line 85 is shown in Fig. 6. The horizontal axis represents time, and the vertical axis represents the vehicle number. Each row represents all the trips performed by a vehicle in one day, and each square represents a trip. The length of each square represents the travel time of a trip. The longer a square, the longer the travel time of the trip represented by the square. This scheme has 12 vehicles, which are all long vehicles. All vehicles have an even number of trips and the total driving time is less than the maximum driving time (13 h for a long vehicle of bus line 85). All departure times in the timetable are covered by trips of vehicles.

Algorithm 1 Learning process of D3QN for the BSP

Hyperparameters: Replay buffer size M, minibatch size B, update rate of the target network α, discount factor γ, pre-training steps tp, the number of episodes N^e, decay rate β, the parameter of epsilon-greedy strategy ϵ.
Notations: Parameters in the main neural network θ, parameters in the target neural network θ^-, replay buffer m, step number i, the length of an episode T.
Initialize parameters θ and θ^- with random values.
Initialize m to be empty and i to be 1.
Initialize ϵ to be 1.
for episode=1 to N^e **do**
  Initialize state matrix s to make each element in the first row be 0 and each other element be -1.
  for i=1 to T **do**
    Choose an action a according to the Q value of the main network by the ϵ-greedy strategy.
    if $i = T$ **then**
      Execute action a and observe the final reward r_m and a new state s'.
      $r = r_m$
    else
      Execute action a and observe the step-wise reward r_b and a new state s'.
      $r = r_b$
    end if
    if $|m| > M$ **then**
      Remove the oldest quadruple from m.
    end if
    Add the quadruple (s, a, r, s') into m.
    $s = s'$, $i = i + 1$.
    if $|m| > B$ and $i > tp$ **then**
      Sample randomly B quadruples (s, a, r, s') from m.
      Calculate the loss function J:

$$J = \sum \frac{1}{B}\left[r + \gamma Q^t\left(s', \arg\max_{a'}\left(Q\left(s', a', \theta\right)\right), \theta^-\right) - Q(s, a, \theta)\right]^2$$

      Update θ with ∇J using Adam back propagation.
      Update θ^- with θ: $\theta^- = \alpha\theta^- + (1-\alpha)\theta$.
      Update ϵ: $\epsilon = \epsilon - \beta$
    end if
  end for
end for

5.2. Online optimization of RL-BSA

Uncertain events, such as abnormal weather, traffic accidents and traffic congestion, may enlarge the travel time of some trips and make the offline bus scheduling scheme infeasible. Once an uncertain event occurs, the trained RL agent used to generate the offline bus scheduling scheme can be directly applied to online scheduling of vehicles according to real-time information of vehicles. It means that vehicles are scheduled according to the

Table 5
Comparison of RL-BSA, ALNS and manual scheduling approach under static environment.

Lines	Approaches		N_u	N_o	N_d	Lines	Approaches		N_u	N_o	N_d
	Manual		11	0	0		Manual		15	2	12
		Max	13	0	0			Max	14	4	0
85	ALNS	Min	13	0	0	60	ALNS	Min	14	0	0
		Avg	13	0	0			Avg	14	2	0
	RL-BSA		12	0	0		RL-BSA		14	2	0
	Manual		11	0	6		Manual		22	2	0
		Max	11	4	0			Max	17	4	0
59	ALNS	Min	11	2	0	803	ALNS	Min	17	0	0
		Avg	11	3	0			Avg	17	2	0
	RL-BSA		8	0	0		RL-BSA		16	2	0

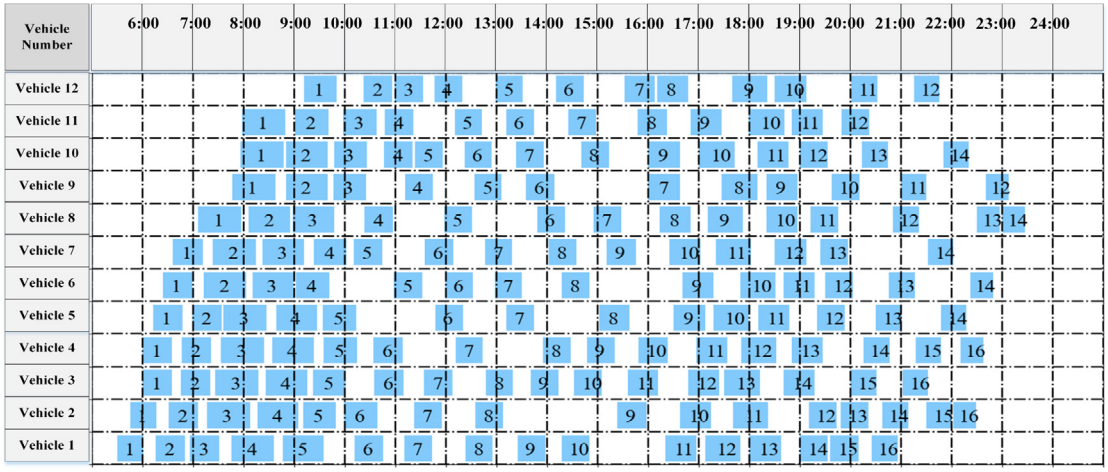


Fig. 6. A vehicle scheduling scheme generated by RL-BSA for bus line 85.

offline bus scheduling scheme when no uncertain event occurs. Once there are uncertain events, the RL agent begins to schedule vehicles in an online manner from the moment when uncertain events occur.

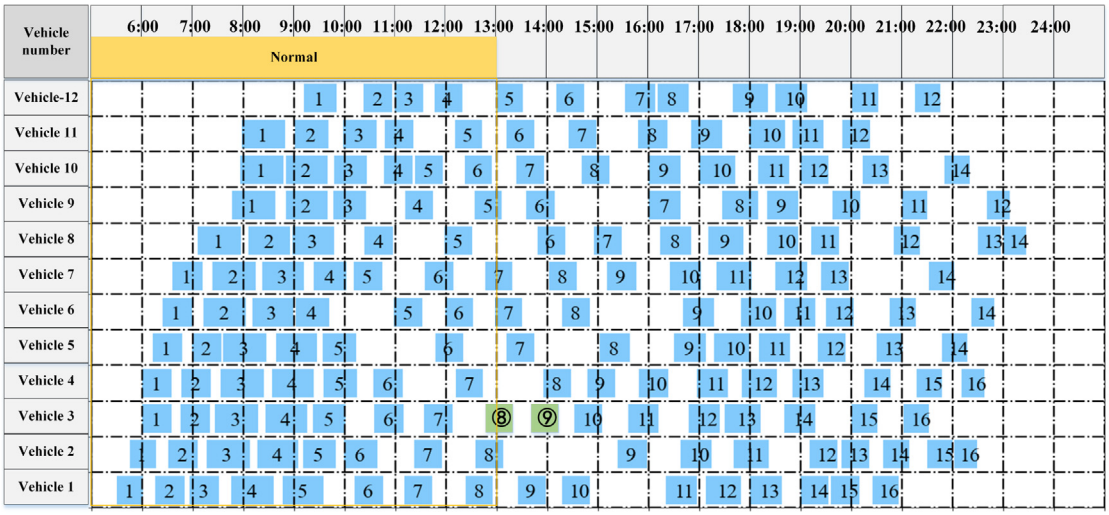
To mimic the online scenario, assume that traffic congestion occurs at 11:00 am and lasts for two hours for bus line 85. We double the travel time of all trips with departure time in the period from 11:00 am to 12:59 pm. It means that the travel time of each running bus during this period is extended by about 33 min. For the manual bus scheduling scheme, when travel time of some trips becomes longer, the vehicles running in this period may not return to the CP in time according to the scheme, which may make the scheme infeasible. In practical bus operation, more vehicles are usually added to maintain the feasibility of bus scheduling scheme or the bus scheduling scheme is adjusted manually to make as many departure times as possible be covered by vehicles.

Fig. 7(a) is the original bus scheduling scheme (also shown in Fig. 6) for bus line 85, and Fig. 7(b) is the bus scheduling scheme generated by RL agent when the traffic congestion occurs. Fig. 7(a) shows the normal travel time of vehicles' trips in the period from 11:00 am to 12:59 pm. The travel time of vehicles' trips in this period is prolonged due to traffic congestion, as shown in Fig. 7(b). The extended travel time results in some trips of vehicles not returning to the CP in time, such that the subsequent trips of those vehicles cannot be performed according to the original scheduling scheme. For example, the 9-th trip of vehicle 3 covers the departure time 13:40 pm originally, i.e., the 9-th trip of vehicle 3 departs at 13:40 pm in Fig. 7(a). The travel time of the 8-th trip of vehicle 3 is prolonged due to traffic congestion in this period, making arriving time of the 8-th trip be greater than the planned departure time of the 9-th trip. Fig. 7(b) shows that the 8-th trip of vehicle 3 is finished at 13:52 pm, greater than 13:40 pm. Thus, vehicle 3 cannot arrive at the CP in time to execute its 9-th trip at 13:40 pm. That is, the 9-th trip of vehicle 3 cannot cover the departure time of 13:40 pm as planned due to traffic congestion. We marked the two trips in green in Fig. 7(a) and (b).

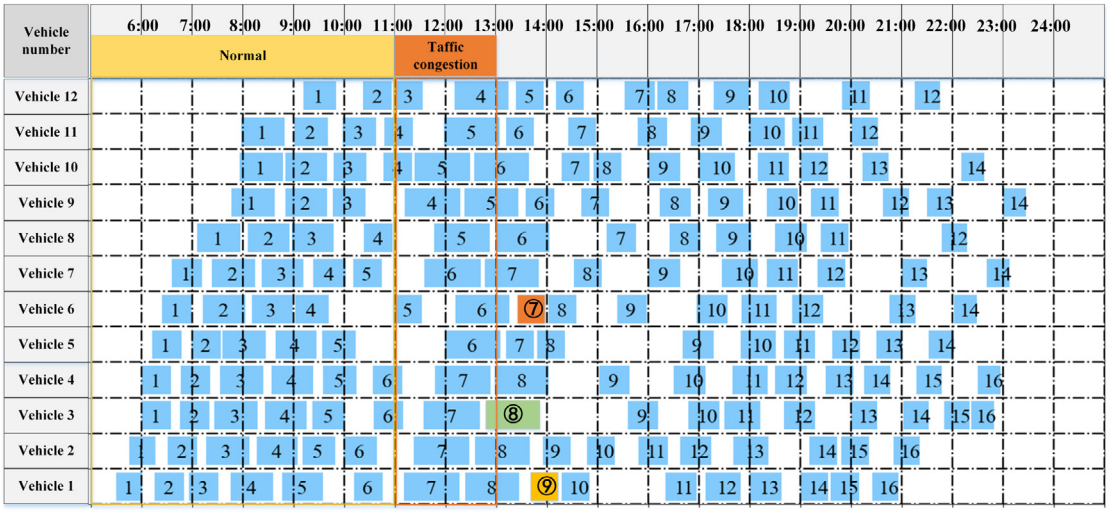
To address the above problem, RL agent schedules vehicles in an online manner to make each departure time in the timetable be covered by exactly one trip of a vehicle. To cover the departure time of 13:40 pm, RL agent makes a decision at the departure time to select the 9-th trip of vehicle 1 (marked in yellow in Fig. 7(b)) to depart at 13:40 pm (cover the departure time). As a result, the departure time of 13:25 pm covered originally by the 9-th trip of vehicle 1 is not covered. To handle this situation, RL agent makes a decision at 13:25 pm to select the 7-th trip of vehicle 6 (marked in orange in Fig. 7(b)) to depart at 13:25 pm to cover the departure time of 13:25 pm. RL-BSA makes decisions on departure times of other vehicles in the same logic, to make each departure time after the traffic congestion be covered exactly by one trip of a vehicle. Comparing Fig. 7(a) and (b), we can see that RL agent can schedule vehicles in real-time once an uncertain event occurs, without increasing the number of vehicles used, N_u. The metrics of N_o and N_d are the same for scheduling schemes in Fig. 7(a) and (b).

Fig. 7 shows that the scheduling scheme generated by online decisions of RL-BSA is very similar to the original one. It means that the duties of drivers do not need to be adjusted. The number of trips for most vehicles remain unchanged. Only vehicle 8 and vehicle 9 have a slight change in the number of trips. The former changes from 14 to 12, and the latter changes from 12 to 14.

The off-duty time of each vehicle in the scheduling scheme generated by RL-BSA is only 3 min later on average than that in original scheduling scheme. Compared to the prolonged travel time (about 33 min) due to traffic congestion, 3 min are very tiny.



(a) Original scheduling scheme



(b) Scheduling scheme generated by online decisions of RL-BSA

Fig. 7. Online scheduling of RL-BSA under traffic congestion.

Fig. 8 compares the total working time of each vehicle under normal traffic condition and traffic congestion. We can see that the total working time of each vehicle does not change much. Total working time of 2 vehicles (vehicles 1 and 12) does not change. Total working time of 6 vehicles becomes slightly longer in the scheduling scheme of RL-BSA. Other 4 vehicles (vehicles 2, 5, 6 and 8) have shorter total working time in the scheduling scheme of RL-BSA. That means that the scheduling scheme generated by online decisions of RL-BSA has little effect on the working time of drivers.

5.3. Ablation experiments

To validate the effectiveness of the components of RL-BSA, ablation experiments are conducted for the constraint handling approach and deep reinforcement learning algorithm.

5.3.1. Constraint handling approach

To validate the effectiveness of the invalid action masking approach, it is compared with an invalid action penalty approach, in which an additional reward is added to the original reward. If an available vehicle is selected, the additional reward is 0; otherwise, the additional reward is -100. The additional reward is to reduce the possibility of selecting an unavailable vehicle. Experiments are conducted to compare RL-BSA and RL-BSA with the invalid action penalty approach (named RL-BSA-P) for bus line 85. Fig. 9(a) and (b) show the convergence curves of the accumulated reward of RL-BSA and RL-BSA-P, respectively. The vertical axis is the accumulated reward of an episode, and the horizontal axis is the number of decision steps. RL-BSA converges at about 20000 decision steps, while RL-BSA-P never converges. It means that the invalid action masking approach can effectively avoid selecting unavailable vehicles. The invalid action penalty approach makes the agent pay too much attention to not selecting unavailable vehicles, resulting in non-convergence of the accumulated reward of RL-BSA-P.

5.3.2. Deep reinforcement learning algorithm

To validate the effectiveness of D3QN of RL-BSA, RL-BSA is compared against RL-BSA with DQN. The D3QN of RL-BSA is replaced with DQN to form an approach RL-BSA-N. Experiments are conducted to compare RL-BSA and RL-BSA-N for bus line 85. Table 6 shows the Mean Square Error (MSE) between Q values outputted by the main network and the training labels computed in Eq. (5). The smaller MSE is, the more accurate Q value is estimated. Table 6 shows that the Q value estimation of RL-BSA

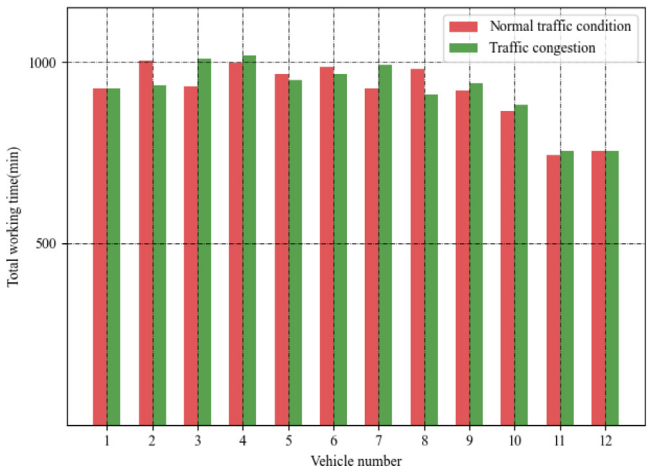


Fig. 8. Total working time of vehicles under normal traffic condition and traffic congestion.

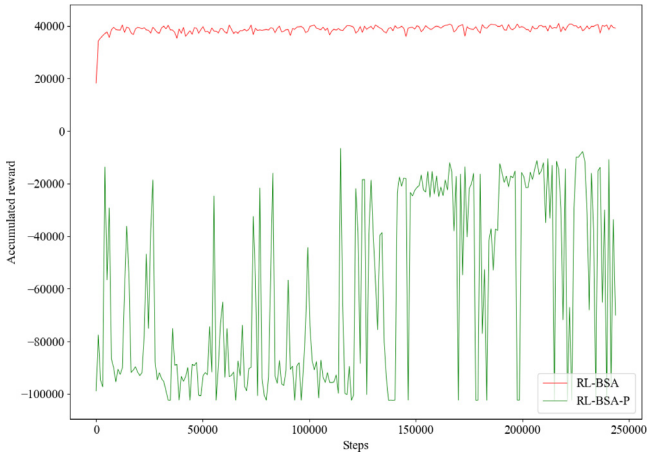


Fig. 9. Comparison of RL-BSA and RL-BSA-P.

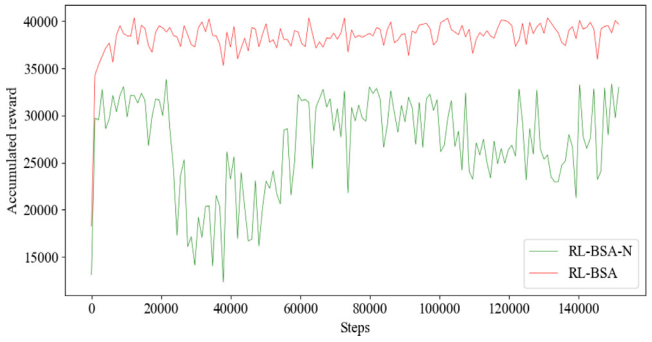


Fig. 10. Comparison of RL-BSA and RL-BSA-N.

is more accurate, and the error of Q value is much smaller than that of RL-BSA-N. Table 6 also presents the metrics of scheduling schemes generated by RL-BSA and RL-BSA-N. The number of vehicles used of RL-BSA is one less than that of RL-BSA-N, and the number of vehicles with odd trips is two less than that of RL-BSA-N. As shown in Fig. 10, both approaches converge. RL-BSA-N has a lower accumulated reward than RL-BSA and its convergence curve fluctuates greatly. This may be due to inaccurate estimate of the Q value. RL-BSA has a more stable convergence curve.

Table 6
Comparison of RL-BSA and RL-BSA-N.

Approaches	MSE	N_u	N_o	N_d
RL-BSA	407.01	12	0	0
RL-BSA-N	941289.31	13	2	0

6. Conclusion

In this paper, an online bus scheduling approach based on reinforcement learning (RL-BSA) is proposed to solve a single-depot Bus Scheduling Problem (BSP). Different from existing solution approaches for such problem, the BSP is modeled as a Markov Decision Process (MDP). Each departure time in a bus timetable is regarded as a decision point. A RL agent makes a decision at each departure time in the timetable to select a vehicle to depart at the departure time. By this means, RL-BSA can schedule vehicles in an online manner to generate a bus scheduling scheme.

In RL-BSA, D3QN is used as the RL agent. Several new state features are devised that involve the real-time information of vehicles. A reward function with a final reward and a step-wise reward is devised. An invalid action masking approach is used to avoid selecting unavailable vehicles.

For the BSP under static environment, experiments on 4 real-world bus lines show that RL-BSA is superior to the manual scheduling approach and Adaptive Large Neighborhood Search (ALNS). For the BSP under uncertain environment, RL-BSA can schedule vehicles in an online manner and effectively cope with uncertain events. What is more, decision time of the agent of RL-BSA is very short (less than 5 ms) and can fully satisfy the real-time requirement of online scheduling under uncertain environment.

Future research directions include: (1) study feature extraction approaches to extract more effective state features to enhance the performance of RL-BSA, and (2) study a reinforcement learning-based scheduling approach for a multi-depot BSP.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that support this study are available at https://github.com/BUPTAIOC/Transportation.

Acknowledgment

This work was supported by National Natural Science Foundation of China under Grant 61873040.

References

[1] Richard Freling, Dennis Huisman, Albert P.M. Wagelmans, Applying an integrated approach to vehicle and crew scheduling in practice, Comput.-Aided Sched. Public Transp. (2001) 73–90.
[2] Richard Freling, Albert P.M. Wagelmans, José M. Pinto Paixão, Models and algorithms for single-depot vehicle scheduling, Transp. Sci. 35 (2) (2001) 165–180.
[3] Celso C. Ribeiro, François Soumis, A column generation approach to the multiple-depot vehicle scheduling problem, Oper. Res. 42 (1) (1994) 41–52.
[4] Natalia Kliewer, Taieb Mellouli, Leena Suhl, A time–space network based exact optimization model for multi-depot bus scheduling, European J. Oper. Res. 175 (3) (2006) 1616–1627.
[5] Xingquan Zuo, Cheng Chen, Wei Tan, MengChu Zhou, Vehicle scheduling of an urban bus line via an improved multiobjective genetic algorithm, IEEE Trans. Intell. Transp. Syst. 16 (2) (2014) 1030–1041.

[6] Xinguo Shui, Xingquan Zuo, Cheng Chen, Alice E. Smith, A clonal selection algorithm for urban bus vehicle scheduling, Appl. Soft Comput. 36 (2015) 36–44.
[7] Dennis Huisman, Richard Freling, Albert P.M. Wagelmans, A robust solution approach to the dynamic vehicle scheduling problem, Transp. Sci. 38 (4) (2004) 447–458.
[8] Chunlu Wang, Hongyi Shi, Xingquan Zuo, A multi-objective genetic algorithm based approach for dynamical bus vehicles scheduling under traffic congestion, Swarm Evol. Comput. 54 (2020) 100667.
[9] Yindong Shen, Zhongyi Zeng, Zhewei Wu, Dynamic vehicle scheduling based on htn, in: 2017 36th Chinese Control Conference, CCC, 2017, pp. 3066–3071.
[10] Marc Naumann, Leena Suhl, Stefan Kramkowski, A stochastic programming approach for robust vehicle scheduling in public bus transport, Procedia-Soc. Behav. Sci. 20 (2011) 826–835.
[11] Yindong Shen, Jia Xu, Jingpeng Li, A probabilistic model for vehicle scheduling based on stochastic trip times, Transp. Res. B 85 (2016) 19–31.
[12] Yadan Yan, Qiang Meng, Shuaian Wang, Xiucheng Guo, Robust optimization model of schedule design for a fixed bus route, Transp. Res. C 25 (2012) 113–121.
[13] Xilin Xin, Yidong Tu, Vladimir Stojanovic, Hai Wang, Kaibo Shi, Shuping He, Tianhong Pan, Online reinforcement learning multiplayer non-zero sum games of continuous-time Markov jump linear systems, Appl. Math. Comput. 412 (2022) 126537.
[14] Zhihe Zhuang, Hongfeng Tao, Yiyang Chen, Vladimir Stojanovic, Wojciech Paszke, An optimal iterative learning control approach for linear systems with nonuniform trial lengths under input constraints, IEEE Trans. Syst., Man, Cybern.: Syst. (2022) 1–13.
[15] Guanqun Ai, Xingquan Zuo, Gang Chen, Binglin Wu, Deep reinforcement learning based dynamic optimization of bus timetable, Appl. Soft Comput. 131 (2022) 109752.
[16] Bo Lin, Bissan Ghaddar, Jatin Nathwani, Deep reinforcement learning for the electric vehicle routing problem with time windows, IEEE Trans. Intell. Transp. Syst. 23 (2021) 11528–11538.
[17] Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, Xu Chi, Learning to dispatch for job shop scheduling via deep reinforcement learning, Adv. Neural Inf. Process. Syst. 33 (2020) 1621–1632.
[18] Habib Allah Ravaghi Ardabili, Mahmoud-Reza Haghifam, Seyyed Mostafa Abedi, A stochastic Markov model for maintenance scheduling in the presence of online monitoring system, IEEE Trans. Power Deliv. 37 (2021) 2831–2842.
[19] Yong Jin, Jia Xu, Sixu Wu, Lijie Xu, Dejun Yang, Enabling the wireless charging via bus network: Route scheduling for electric vehicles, IEEE Trans. Intell. Transp. Syst. 22 (2020) 1827–1839.
[20] Shiyao Zhang, Ka-Cheong Leung, Joint optimal power flow routing and vehicle-to-grid scheduling: Theory and algorithms, IEEE Trans. Intell. Transp. Syst. 23 (2020) 499–512.
[21] Richard Bellman, A Markovian decision process, J. Math. Mech. (1957) 679–684.
[22] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, Jost Tobias Springenberg, Learning by playing solving sparse reward tasks from scratch, in: International Conference on Machine Learning, 2018, pp. 4344–4353.
[23] Satinder Singh, Richard L. Lewis, Andrew G. Barto, Jonathan Sorg, Intrinsically motivated reinforcement learning: An evolutionary perspective, IEEE Trans. Auton. Men. Dev. 2 (2010) 70–82.
[24] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, Nando Freitas, Dueling network architectures for deep reinforcement learning, in: International Conference on Machine Learning, 2016, pp. 1995–2003.
[25] Meng Long, Xiexin Zou, Yue Zhou, Edward Chung, Deep reinforcement learning for transit signal priority in a connected environment, Transp. Res. C 142 (2022) 103814.
[26] Min Wen, Esben Linde, Stefan Ropke, P. Mirchandani, Allan Larsen, An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem, Comput. Oper. Res. 76 (2016) 73–83.
[27] Jing-Quan Li, K. Larry Head, Sustainability provisions in the bus-scheduling problem, Transp. Res. Part D: Transp. Environ. 14 (1) (2009) 50–60.
[28] Knut Haase, Guy Desaulniers, Jacques Desrosiers, Simultaneous vehicle and crew scheduling in urban mass transit systems, Transp. Sci. 35 (3) (2001) 286–303.
[29] Yongjie Lin, Shuliang Pan, Lei Jia, Nan Zou, A bi-level multi-objective programming model for bus crew and vehicle scheduling, in: 2010 8th World Congress on Intelligent Control and Automation, 2010, pp. 2328–2333.
[30] Ahmed Hadjar, François Soumis, Dynamic window reduction for the multiple depot vehicle scheduling problem with time windows, Comput. Oper. Res. 36 (7) (2009) 2160–2172.
[31] Jonathan D. Adler, Pitu B. Mirchandani, The vehicle scheduling problem for fleets with alternative-fuel vehicles, Transp. Sci. 51 (2) (2017) 441–456.
[32] Vincent Boyer, Omar J. Ibarra-Rojas, Yasmín Á Ríos-Solís, Vehicle and crew scheduling for flexible bus transportation systems, Transp. Res. Part B: Methodol. 112 (2018) 216–229.
[33] K. Gkiotsalitis, C. Iliopoulou, K. Kepaptsoglou, An exact approach for the multi-depot electric bus scheduling problem with time windows, European J. Oper. Res. (2022).
[34] Maroš Janovec, Michal Kohání, Exact approach to the electric bus fleet scheduling, Transp. Res. Procedia 40 (2019) 1380–1387.
[35] Avishai Avi Ceder, Public-transport vehicle scheduling with multi vehicle type, Transp. Res. C 19 (3) (2011) 485–497.
[36] Sarang Kulkarni, Mohan Krishnamoorthy, Abhiram Ranade, Andreas T. Ernst, Rahul Patil, A new formulation and a column generation-based heuristic for the multiple depot vehicle scheduling problem, Transp. Res. B 118 (2018) 457–487.
[37] Enjian Yao, Tong Liu, Tianwei Lu, Yang Yang, Optimization of electric vehicle scheduling with multiple vehicle types in public transport, Sustainable Cities Soc. 52 (2020) 101862.
[38] Mengyuan Duan, Geqi Qi, Wei Guan, Chaoru Lu, Congcong Gong, Reforming mixed operation schedule for electric buses and traditional fuel buses by an optimal framework, IET Intell. Transp. Syst. 15 (10) (2021) 1287–1303.
[39] Jing Teng, Tong Chen, Wei Fan, Integrated approach to vehicle scheduling and bus timetabling for an electric bus line, J. Transp. Eng., Part A: Syst. 146 (2) (2020) 04019073.
[40] Zhao Xinchao, Sun Hao, Lu Juan, Li Zhiyu, Dp-tabu: An algorithm to solve single-depot multi-line vehicle scheduling problem, Complex Intell. Syst. 8 (6) (2022) 4441–4451.
[41] Xinfeng Yang, Yicheng Qi, Research on optimization of multi-objective regional public transportation scheduling, Algorithms 14 (4) (2021) 108.
[42] Samuela Carosi, Antonio Frangioni, Laura Galli, Leopoldo Girardi, Giuliano Vallese, A matheuristic for integrated timetabling and vehicle scheduling, Transp. Res. B 127 (2019) 99–124.
[43] Chunlu Wang, Congcong Guo, Xingquan Zuo, Solving multi-depot electric vehicle scheduling problem by column generation and genetic algorithm, Appl. Soft Comput. 112 (2021) 107774.
[44] Yahong Liu, Chunyang Cheng, Hongyi Shi, Xingquan Zuo, Shaohua Chen, A two-stage approach with a departure time based solution representation for electric bus vehicle scheduling, IEEE Access 10 (2022) 112799–112811.
[45] Yahong Liu, Xingquan Zuo, Guanqun Ai, Xinchao Zhao, A construction-and-repair based method for vehicle scheduling of bus line with branch lines, Comput. Ind. Eng. 178 (2023) 109103.
[46] Oskar A.L. Eikenbroek, Konstantinos Gkiotsalitis, Robust rescheduling and holding of autonomous buses intertwined with collector transit lines, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC, 2020, pp. 1–7.
[47] Konstantinos Gkiotsalitis, Bus rescheduling in rolling horizons for regularity-based services, J. Intell. Transp. Syst. 25 (4) (2021) 356–375.
[48] Jing-Quan Li, Pitu B. Mirchandani, Denis Borenstein, Parallel auction algorithm for bus rescheduling, in: Computer-Aided Systems in Public Transport, 2008, pp. 281–299.
[49] Jing-Quan Li, Denis Borenstein, Pitu B. Mirchandani, A decision support system for the single-depot vehicle rescheduling problem, Comput. Oper. Res. 34 (2007) 1008–1032.
[50] Monize Sâmara Visentini, Denis Borenstein, Jing-Quan Li, Pitu B. Mirchandani, Review of real-time vehicle schedule recovery methods in transportation services, J. Sched. 17 (6) (2014) 541–567.
[51] Xindi Tang, Xi Lin, Fang He, Robust scheduling strategies of electric buses under stochastic traffic conditions, Transp. Res. C 105 (2019) 163–182.
[52] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[53] Hado Van Hasselt, Arthur Guez, David Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, 2016, pp. 2094–2100.
[54] Scott Fujimoto, Herke Hoof, David Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, 2018, pp. 1587–1596.