World Wide Web (2020) 23:2491–2511
https://doi.org/10.1007/s11280-020-00804-z

Rebalancing the car-sharing system with reinforcement learning

Changwei Ren[1] · Lixingjian An[1] · Zhanquan Gu[2] · Yuexuan Wang[1,3] · Yunjun Gao[1]

Received: 30 August 2019 / Revised: 29 January 2020 / Accepted: 21 February 2020 /
Published online: 20 April 2020

Abstract
With the sharing economy boom, there is a notable increase in the number of car-sharing corporations, which provided a variety of travel options and improved convenience and functionality. Owing to the similarity in the travel patterns of the urban population, car-sharing system often faces the problem of imbalance in the number of shared cars within the spatial distribution, especially during the rush hours. There are many challenges in redressing this imbalance, such as insufficient data and the large state space. In this study, we propose a new reward method called Double P (Picking & Parking) Bonus (DPB). We model the research problem as a Markov Decision Process (MDP) problem and introduce Deep Deterministic Policy Gradient, a state-of-the-art reinforcement learning framework, to find a solution. The results show that the rewarding mechanism embodied in the DPB method can indeed guide the users' behaviors through price leverage, increase user stickiness, and cultivate user habits, thereby boosting the service provider's long-term profit. In addition, taking the battery power of the shared car into consideration, we use the method of hierarchical reinforcement learning for station scheduling. This station scheduling method encourages the user to place the car that needs to be charged on the charging post within a certain site. It can ensure the effective use of charging pile resources, thereby rendering the efficient functioning of shared cars.

Keywords Reinforcement learning · Car-sharing system · Scheduling

1 Introduction

In recent years, with the surge of the sharing economy and an increase in the number of bicycle-sharing service providers in China, shared cars have entered the market and found their way into urban life. Given the increasing costs of car purchase and maintenance in the

This article belongs to the Topical Collection: *Special Issue on Data Science in Cyberspace 2019*
Guest Editors: Bin Zhou, Feifei Li and Jinjun Chen

✉ Zhanquan Gu
zqgu@gzhu.edu.cn

Extended author information available on the last page of the article.

first- and second-tier cities in China, more young people have started to prefer travelling by means of shared cars to buying a new one. Other factors, such as policies on the upper limit of mileage, slim chances of getting a car plate in the plate lottery system, and troubles in finding a parking place in cities, also contribute to this shift of preference. Car sharing, a new type of car rental model, is different from conventional car rental services. Car-sharing service providers offer service for users who want to rent a car on a short-term basis and often charge according to time or mileage. Their target customers are those who occasionally use cars and enjoy trying different types of cars sometimes rather than those who use cars every day. For these consumers, the car-sharing system not only saves them parking fees but also provides convenience in picking up and returning the cars.

However, owing to the similar travel patterns of most users, the car-sharing system often results in a supply-demand imbalance of cars in given locations, especially during the rush hours. For example, during the peak traffic hours in the morning, most users pick up cars from their living areas and return them to the work areas or commercial areas. As a result, the number of available cars reaches the lowest in the living areas, whereas it peaks highest in the work areas. Situations are the same during the evening rush hours. Such an imbalanced distribution of shared cars undermines not only user experience, but also the service provider's income to some extent. Therefore, how to balance the distribution of cars is a major problem that car-sharing service providers need to address immediately.

There are several challenges while balancing the distribution of shared cars. First of all, insufficient data hampers the generalization of algorithms to resolve the problem. Second, the large quantity of parking lots in cities results in an extremely large state space of the problem and an exponential increase in the computing time. Last, users' needs are subject to many factors, such as the weather and special events, which makes it difficult to design a universal framework to address the problem.

In this study, we present the Double P (Picking & Parking) Bonus (DPB) method that offers two rewarding mechanisms for the users so that they can simultaneously contribute to balancing the car-sharing system: the picking bonus and the parking bonus. The framework of the algorithm is shown in Figure 1. We model the research problem as a Markov Decision Process (MDP), where "state" denotes the number of cars for each parking station in each time period; "action," denotes the picking bonus and parking bonus for the user; and "reward," the service provider's profit calculated using the DPB Method. We use Deep Deterministic Policy Gradient (DDPG), a state-of-the-art reinforcement learning (RL) algorithm, to process the problem. Our goal is to minimize the service provider's scheduling cost. In addition, taking the power of the shared car into consideration, we use the method of hierarchical reinforcement learning for station scheduling. We propose the HDDPG algorithm that motivates the user to place the car that needs to be charged on a charging post within a certain site. This type of station scheduling method facilitates the reasonable use of charging pile resources and ensures the healty functioning of shared cars.

Our main contributions are summarized as follows:

- We propose a new car-sharing balance framework that models the research problem as a MDP with the goal of minimizing the service provider's scheduling cost, while maintaining a certain service rate.
- We propose a DPB method that introduces two types of rewarding mechanisms, the picking bonus and the parking bonus, to guide users to balance the car-sharing system.
- We consider the power of the shared car inside the site and use the hierarchical reinforcement learning algorithm to induce the user to balance the problem of the vehicle's

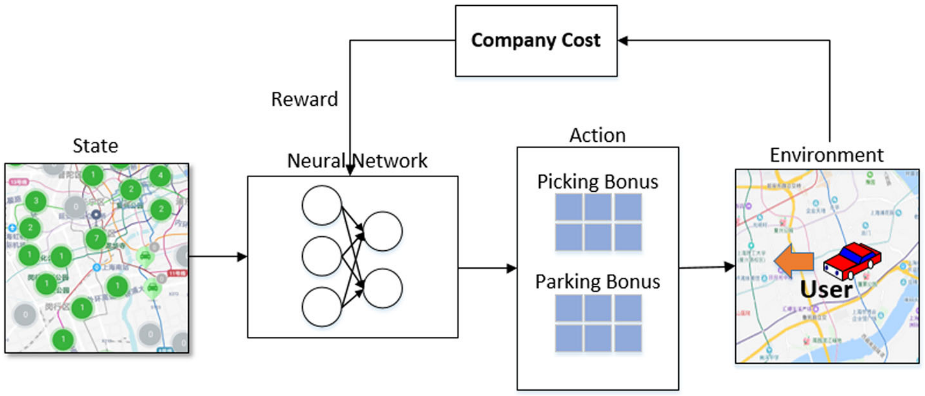

Figure 1 The reinforcement learning framework for rebalancing car-sharing

power imbalance between the vehicle in the charging pile and the uncharged pile inside the site.
– We experimented our method on the self-made data set and found that it can practically reduce the service provider's scheduling cost while improving user satisfaction.

In the rest of this article, Section 2 reviews preceding studies on related topics. Section 3 describes the system model for the research problem. Section 4 introduces the framework of the Actor-Critic algorithm for car sharing. Section 5 presents the experiments. Section 6 presents the conclusion of this study.

2 Related studies

2.1 Rebalancing method

In other forms of sharing economy, such as the bicycle-sharing system which has a similar operating mechanism as that of car-sharing systems, the problem of imbalanced distribution has received extensive research. Balanced distribution of shared bicycles is realized mainly in two ways: the service provider assigns its staff to dispatch the bicycles manually [9, 14, 15, 20] or the users actively balance the bicycle-sharing system [3, 8, 21, 26].

Solutions reached by preceding studies on shared cars in recent years to alleviate the supply-demand imbalance of the car-sharing system mainly include the following four options: The first option is manual scheduling: the service provider's staff transfer the cars from car-crowded sites to high-demand locations [18]. The second option is to introduce a mechanism of preference of order; specifically, the system decides whether to accept a booking, depending on its impact on the balance of the system. The third option is to optimize the layout; The selection of sites and allocation of cars is adequately carried out at the beginning of operations. Finally, the fourth option is adaptive scheduling,in which price is used as leverage to adjust the demand for cars at each site [19]. In the first option, a coordinated scheduling strategy among the car-sharing sites is designed considering the various aspects of car sharing. The fourth option, despite the lack of research on adaptive scheduling strategies, achieves good results as the price can be used to motivate users to change behaviors, and it alleviates the supply-demand imbalance of cars in the system in real time.

2.2 Reinforcement learning

Reinforcement learning (RL) is a learning method that inspires actions in response to the environment with the goal of maximizing the agents' cumulative rewards in their interactions with the environment [24]. Some preliminary research has been done before the advent of deep reinforcement learning (DRL). However, owing to the lack of training data and limited computing capacity,these studies only used deep neural networks to reduce the dimensionality of the input data to facilitate data processing by the conventional RL algorithm. Mnih et al. [16, 17] combined the convolutional neural network (CNN) with the Q-Learning [28] algorithm in the conventional RL algorithm and proposed a Deep Q-Network (DQN) model based on visual perception for the processing of the control tasks. The DQN algorithm is a groundbreaking work in the field of DRL. Policy gradient is a method which is usually used in strategy optimization. It updates the policy parameters by continuously calculating the expected reward of a strategy before finally reaching the optimal strategy through convergence [25]. Therefore, in the DRL problem, a deep neural network with parameters θ is used to parameterize the strategy, and the policy gradient method is used to optimize the strategy. Lillicrap et al. [13] used the DQN-extended Q-Learning algorithm to develop the Deterministic Policy Gradient (DPG) method and proposed a deep deterministic strategy algorithm based on the Actor-Critic framework (Deep Deterministic Policy Gradient, DDPG) algorithm to solve the DRL problem in the continuous action space.

Van Seijen [27] proposed a RL method called Hybrid Reward Architecture (HRA) that takes a decomposed reward function as input and learns a separate value function for each component reward function. Because each component typically depends on a subset of all features alone, the corresponding value function can be approximated more easily by a low-dimensional representation, thereby enabling more effective learning.

Hierarchical reinforcement learning [4, 5, 11] decomposes a large problem into several smaller sub-problems learned by sub-agents, which is suitable for large-scale problems. Each sub-agent only focuses on learning sub-Q-values for its subMDP. Thus, the sub-agent can neglect part of the state which is irrelevant to its current decision [1] to enable faster learning.

3 System model

3.1 Notation

We divide the shared cars across n regions, represented as $\{r_1, r_2, ..., r_n\}$. We divide the day into t identical time periods, denoted by $T = \{t_1, t_2, ..., t_n\}$. $S_i(t)$ denotes the number of cars supplied at station i at the beginning of the time period t. The n-dimensional vector of $S(t) = \{S_i(t), \forall i\}$ represents the quantity of cars supplied by each station at the beginning of the time period t.

3.2 Problem description

We define a closed-loop system consisting of n stations $\{s_1, s_2, ..., s_n\}$, each with $\{c_1, c_2, ..., c_n\}$ cars and ten charging piles (parking spots).

The user can select any start point s_i and an end point s_j on the application (APP) before leaving. The user's request, however, may fail to establish because of the imbalance in supply and demand of shared cars in a given location.

Car-sharing service providers can offer users two options to earn rewards before an order is placed. One option is to pick up the car within the time limit specified by the service provider; the other is to park the car at the parking station specified by the service provider. The provider's total cost is $cost = \sum bonus$.

The bonus set only correlates to time T and the site S but not to other factors such as the user and car. Therefore, the bonus is a discrete function of the time T and station S. Each time a bonus is provided to the user (when the user selects the start and end points), the information of parking bonus for each station is displayed real-time on the APP, and the user can select the site for car parking.

Our goal is to use algorithms to determine the rewards so that the service provider's scheduling cost can be minimized with a threshold R_{Ser} user service rate ensured.

$$\begin{aligned} & min\ cost \\ & s.t. \quad service\ rate \geqslant R_{Ser} \end{aligned} \tag{1}$$

3.3 DPB method

We propose the Double P (Picking & Parking) Bonus (DPB) method to reward users for picking up and parking the cars.

Algorithm 1 The picking bonus schema.

```
Input: User $u$, user location $l_u$, user chosen parking station $p_u$, current state $s_t$, and
       expected picking time $T$.
Output: In the current state, the user chooses to arrive at the parking station within $T$
        minutes to pick up the car.
1: Calculate the time $x$ required for the user to reach the pick-up point on average
   according to the user's current location and the location of the selected car.
2: Calculate the rewards $c$ that the user expects to pick up the car within $T$ minutes.
3: Compare $c$ with the $b_p$ (picking bonus) given by the action parking station in the
   current state.
4: if $c \leqslant b_p$ then
5:     Accept this reward and reach the parking station within $T$ minutes.
6: else
7:     Give up this reward and arrive at the parking station according to the user's
       original plan.
```

3.3.1 Picking donus

To increase the turnover speed of each car, information about the bonus for quick pick-up at each station will be presented real-time to the users when they select a car. A user who picks up a car within T minutes will receive a bonus, and this user can decide whether to accept it according to his or her current situation. Since it is the user who makes the decision, the choice will depend on the distance between the user's current location and the pick-up station. The user may make some effort for the bonus, such as brisk walking or trot. Therefore, we have designed the following function to determine whether the user will make some efforts for the reward. Our designed function is $c_i(x, i)$ - it represents the moving speed the user needs to reach to qualify for the bonus by picking up the car within T minutes at i station in time t, where x is the time that the user estimates will be needed to reach the pick-up station.

$$c_t(x, i) = \begin{cases} 0 & x \leqslant T \\ \alpha e^{\frac{x}{T}-1} & x > T \end{cases} \tag{2}$$

According to this function, the cost is 0 when the user picks up the car within T minutes. When the user spends more than T minutes, we calculate the speed that the user needs to achieve in order to pick up the car within 5 minutes.The higher the speed, the greater payment of the user. α is the weight of the reward that the user expects to receive.

3.3.2 Parking bonus

The user needs to select a parking station S_j around the destination before leaving. While selecting the parking station, the user follows a simple rule: select the station with the highest reward within Dis to park the car.

3.4 Inner-station scheduling

To further improve the robustness of the entire system, we consider another variable that affects the system efficiency: the amount of electric power in the car. As we all know, shared car companies usually use new energy vehicles as the primary means of transportation. It is, therefore, essential to ensure that each car's power is maintained within an appropriate range.

In a station, we have p charging stations and q parking spaces. In order to make the most efficient use of the charging efficiency, we have to attempt to park the p-lowest car on the charging pile. According to our survey, the current practice followed by the service providers of shared cars is the conventional method of manual scheduling, that is, sending special employees to each station to dispatch vehicles to ensure that low-power vehicles are being charged and high-power vehicles are in the respective parking spaces. This approach has many advantages: distributed human resource management facilitates intelligent scheduling, and the personnel are subject to centralized command, ensuring interaction between the sites. However, the disadvantages are apparent: manual scheduling incurs high costs, real-time communication with the user is inefficient, and so on. To resolve these limitations, we have attempted to parameterize each station and add it to the system.

The solution is as follows: For each station, define a parameter:

$$station\_entropy = \frac{log E_{park}}{log E_{charge}} \tag{3}$$

where E_{park} represents the remaining power of the car parking at the parking spot and E_{charge} represents the remaining power of the car parking at the charging pile. Further, we can obtain an intuitive understanding of the parameter $station\_entropy$ from Figure 2. It lucidly shows that the lower the remaining power the car at the parking spot has and the higher the remaining power the car at the charging pile has, the higher the value of

$station\_entropy$ will be, and the larger the difference is, the more it is needed to swap the location of the two cars.

When $station\_entropy \geq \epsilon$, the system will issue an in-station scheduling request to the nearest user and offer the corresponding reward amount. The user chooses whether to accept the reward given by the system. If the user wants to accept the reward, the vehicle in the station can be adjusted according to the instructions of the system so that the distribution within the station reaches an optimal state. The best condition is that all the low-power p cars in this site are being charged.

4 Rebalance the system as an MDP model

4.1 MDP formulation

Deep reinforcement learning can solve the limitations of the current model. We can consider the entire closed-loop system as an environment, the user's reward setting as the action taken by the agent, and the user's reaction as the reward. Our problem can be converted into an online learning problem because the rewarding mechanism can directly determine the service provider's profitability. Therefore, we modeled the problem as a MDP, which can be represented by a five-tuple (S, A, Pr, R, γ), where S denotes the "state"; A, the "action"; Pr, the transition probability matrix; R, the reward; and γ, the discount factor. The description is as follows:

– S: "State" should be able to describe the information of the entire environment at each moment; therefore, our state is mainly composed of the state information of cars at each station, which is specifically expressed as the number of cars remaining at each station because the number of piles is fixed as ten. Therefore, the vacancy information of stations can be determined according to the number of cars.
– A: "Action" consists of information on the reward settings of the service provider. The reward is constituted of the bonus amounts for not only parking a car at a specified station (called the "Parking Bonus"), but also picking up a car after booking it (called the "Picking Bonus").

  Parking Bonus: In a given time period, the system presents the bonus amount for each site so that the users can choose where to park the shared car they have rented.

  Picking Bonus: If the user picks up the car within a time interval δT of the appointment, a bonus is given. If the user picks up the car within 5 minutes after making the appointment, he/she will receive a bonus between 0 and 5 RMB.

  The vector of the bonus for picking up a car within the specified time limit in the current time period at a given station and the bonus for parking a car at a given station when the user already selects a station is presented as $a_t = (pt_{1t}, pt_{2t}, .., pt_{nt}, ps_{1t}, ps_{2t}, ..., ps_{nt})$, where pt_{it} represents the bonus for parking the car at the i-th station within the t-time limit, and ps_{it} indicates the bonus for parking at the i-th station during the t period.
– R: In reinforcement learning, the agent' behaviors are driven by the reward. Therefore, an appropriate reward can not only motivate the agent to learn expected behaviors, but also ensure a higher convergence speed of the algorithm. Since the purpose of our system is to increase the car-sharing service provider's profit, the respective amount

of the provider's profit before and after the adoption of the rewarding mechanism can be directly used as a reward function. In this way, the reward is directly linked to the provider's profit and loss so that setting the rewarding method is not only to reward the users' behavior, but also to serve the overall interests of the service provider.

– Pr: $Pr(s_{t+1} \mid s_t, a_t)$ indicates the probability of transition from State s_t to State s_{t+1} while taking Action a_t. Setting of the transition probability is an important part of MDP. However, in the definition of the shared car, state transfer is a pre-determined process rather than a random one; therefore, $Pr = 1$ is established for any state transfer. This also simplifies the model.
– γ: Because of the parameter γ, the reinforcement learning algorithm focuses on the immediate benefits rather than the optimal benefits in the long run. Generally, in an infinite state MDP, in order to enable the agent to learn better and faster, $\gamma = 0.99$ or $\gamma = 0.9$ is set. In our model, one day is divided into 24 time periods and only 24 steps is needed at a time, which allows us to combine the benefit of the current time period with that of all subsequent time periods; therefore, $\gamma = 0.99$ is more suitable for the model.

Algorithm 2 DDPG algorithm for car-sharing.

```
Input: Randomly initialize the critic network $Q_{\theta^Q}(s,a)$ and the actor network $\mu_{\theta^\mu}(s)$
       with weight $\theta^Q$ and $\theta^\mu$.
Output: Target network $Q'(s,a)$.
1: Initialize replay buffer R.
2: for episode = 1, M do
3:     Initialize a random process $\mathcal{N}$ for action exploration.
4:     Receive the initial state $s_1$.
5:     for step t = 1, N do
6:         Select action $a_t = \mu(s_t) + \mathcal{N}$ according to the current policy and exploration
           noise.
7:         Execute action $a_t$, and observe the reward $R_t$ and the next state $s_{t+1}$.
8:         Store transition $(s_t, a_t, R_t, s_{t+1})$ in experience replay buffer $\mathcal{B}$.
9:         Sample a random mini-batch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $\mathcal{B}$.
10:        Compute current state-action pair's Q-value $Q_{\theta^Q}(s_i, a_i)$.
11:        Compute action for next state by actor network: $a'_{i+1} = \mu'_{\theta^{\mu'}}(s_{i+1})$
12:        Compute next state-action pair's Q-value $Q'_{\theta^{Q'}}(s_{i+1}, a'_{i+1})$.
13:        Set $y_i = R_i + \gamma Q'_{\theta^{Q'}}(s_{i+1}, a'_{i+1})$.
14:        Update critic network by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i))^2$.
15:        Update actor network by using the sampled policy gradient
               $$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q_{\theta^\mu}(s,a)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s)|_{s_i}$$
16:        Update the target networks:
               $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
               $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
```

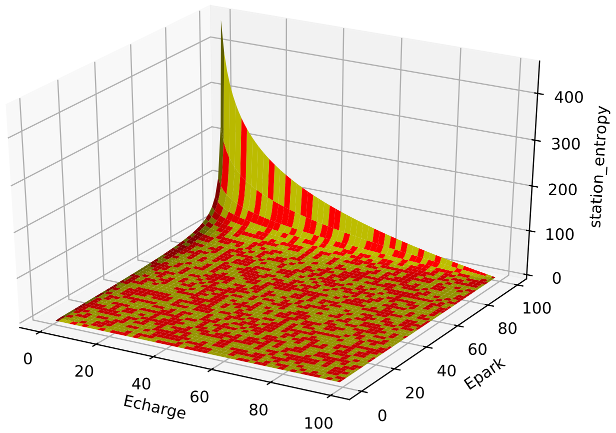

Figure 2 *station_entropy* for different remaining power levels of the charging and parked cars

4.2 Reinforcement learning method

One key point in our study is that the research problem is a continuous action space. Given the advancements in reinforcement learning, we introduce the state-of-the-art reinforcement learning algorithm. Similar to the DQN algorithm that is a model-free and an off-policy algorithm (the policy of generating behaviors is different from the policy for evaluation), the DDPG (Deep Deterministic Policy Gradient) algorithm also uses neural networks to realize function approximation. However, unlike DQN that can only solve problems when the action space is discrete and low-dimensional, the DDPG algorithm can solve problems when the action space is continuous. In addition, DQN is a value-based method, that is, there is only one value function network, while DDPG is an actor-critic method, that is, it consists of a value function network (critic network) and a policy network (actor-network). Compared with the DPG (Deterministic Policy Gradient) algorithm, DDPG algorithm simulates the policy function and the Q function by using a convolutional neural network and uses deep learning for training, which proves the accuracy and high performance of the algorithm and its ability to converge nonlinear simulation functions to the reinforcement learning method. Moreover, as DDPG introduces experience replay memory, the actor will store the transition data into the experience replay buffer and then randomly sample the mini-batch data from the buffer during training so that the sample data can be considered as uncorrelated. The target and online networks are used synchronously, thereby facilitating a stabilized learning process and secure convergence.

As shown in Figure 3, the DDPG algorithm uses a neural network to approximate the value function. This value function network is also called the critic network. Its inputs are Action a and State s, and the output is $Q(s, a)$; another neural network is used to approximate the policy function. This policy network is also called the actor network, in which the input is State s and the output is Action a. The connection between the two networks is as follows: First, the environment will give a state, and the agent will make a decisive action according to the actor network. After receiving the action, the environment will generate a reward and a new state; this process is a step. At this point, we need to update the critic network according to the reward and then update the actor network in the direction suggested by a critic. Then it moves to the next step. This training cycle continues until we achieves a good actor network.

Algorithm 3 HDDPG algorithm for car sharing.

```
Input: Randomly initialize the critic network $Q_{\theta^Q}(s,a)$ and the actor network $\mu_{\theta^\mu}(s)$
       with weight $\theta^Q$ and $\theta^\mu$.
Output: Target network $Q'(s,a)$.
1:  Initialize replay buffer R.
2:  for episode = 1, M do
3:      Initialize a random process $\mathcal{N}$ for action exploration.
4:      Receive the initial state $s_1$.
5:      for step t = 1, N do
6:          Select action $a_t = \mu(s_t) + \mathcal{N}$ according to the current policy and exploration
            noise.
7:          Execute action $a_t$, and observe the reward $R_t$ and the next state $s_{t+1}$.
8:          Store transition $(s_t, a_t, R_t, s_{t+1})$ in experience replay buffer $\mathcal{B}$.
9:          Sample a random mini-batch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $\mathcal{B}$.
10:         ★ Compute current state-action pair's Q-value
11:         Compute the decomposed Q-values $Q^j_{\theta^Q_j}(s_{ji}, a_{ji})$ for each parking station $r_j$
12:         Compute the bias-correction term $b_j(s_{j(i+1)}, a_{j(i+1)}, F(s_i, r_j))$ for each
            station $r_j$
13:         Compute current state-action pair's Q-value $Q_{\theta^Q}(s_i, a_i)$.
14:         ★ Compute next state-action pair's Q-value
15:         Compute action for next state by actor network: $a'_{i+1} = \mu'_{\theta^{\mu'}}(s_{i+1})$.
16:         Compute the decomposed Q-values $Q^j_{\theta^{Q'}_j}(s_{j(i+1)}, a'_{j(i+1)})$ for each parking
            station $r_j$
17:         Compute the bias-correction term $b_j(s_{j(i+1)}, a'_{j(i+1)}, F(s_{i+1}, r_j))$ for each
            station $r_j$
18:         Compute next state-action pair's Q-value $Q'_{\theta^{Q'}}(s_{i+1}, a'_{i+1})$.
19:         Set $y_i = R_i + \gamma Q'_{\theta^{Q'}}(s_{i+1}, a'_{i+1})$.
20:         Update critic network by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i))^2$.
21:         Update actor network by using the sampled policy gradient
                $$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q_{\theta^\mu}(s,a)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s)|_{s_i}$$
22:         Update the target networks:
                $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
                $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
```

4.3 HDDPG model for car sharing

As the problems faced in car sharing are large scale and multidimensional in nature, we considered the introduction of a Hierarchical Deep Deterministic Policy Gradient (HDDPG)

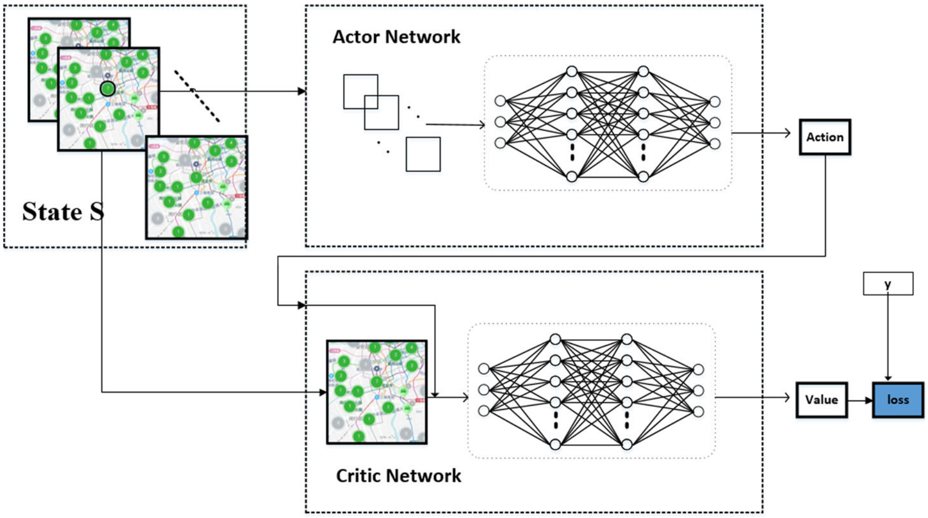

Figure 3 The DDPG algorithm for rebalancing car-sharing

algorithm, Algorithm 3. Hierarchical reinforcement learning [2, 4–7, 11], which breaks down large problems into sub-problems, learns through sub-agents. Each sub-agent learns the sub-Q-value of its corresponding MDP problem. The sub-agent can ignore some of the states that are not relevant to the current decision to learn faster.

We decompose the Q-values in the entire region into sub-Q-values according to each site, and Q-values can be estimated by the combination of sub-Q-values. $\sum_{j=1}^{n} Q^j(s_{jt}, a_{jt})$,

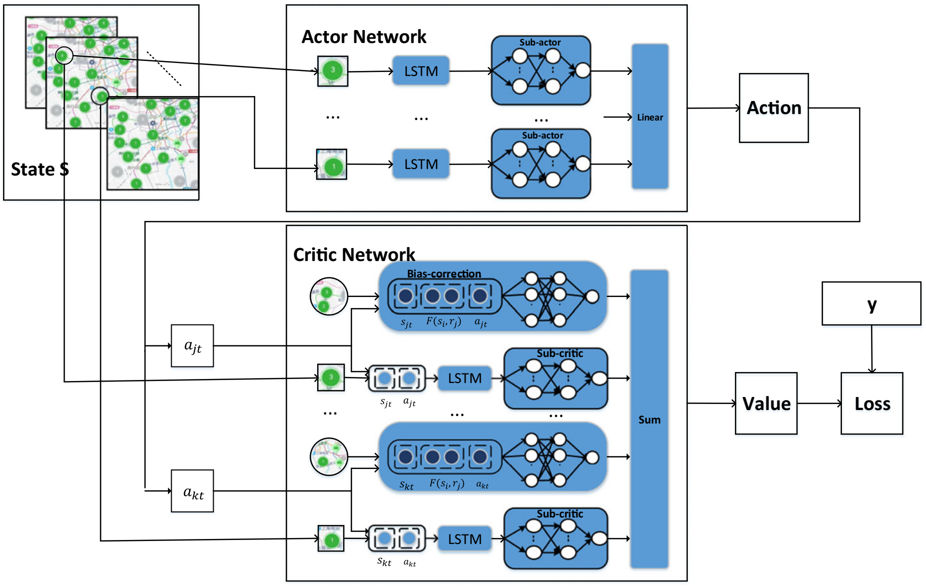

Figure 4 The HDDPG algorithm for rebalancing car-sharing

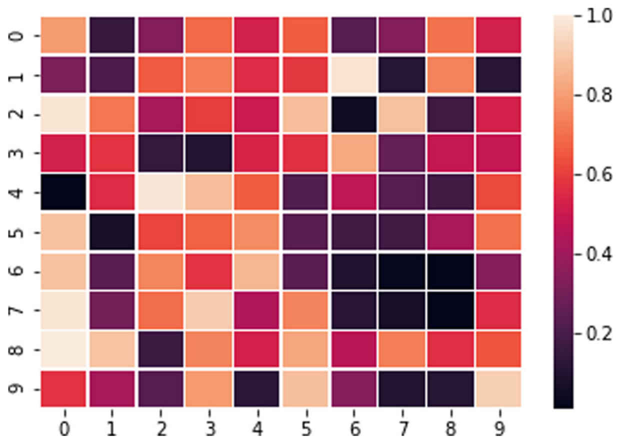

Figure 5 Order distribution

where s_{jt}, a_{jt} are represented as sub-state and sub-action at r_j site in timestep t, and μ_j represents estimator parameters. For each timestep, the current state dynamic depends on past user pick-up and the state of parking vehicles. In order to take advantage of the direct sequence relationships that occur in each state, we use the Long Short-Term Memory (LSTM) unit [10] to train each sub-agent. The LSTM network can store information about the previous state and capture complex sequence dependencies. However, directly decomposing the regions will cause a large error because of the dependence between each site and the adjacent sites. In our model,in addition to parking cars at the current site, users can also park cars at adjacent sites, leading to the direct link relationship of each site. Therefore, we need to consider this important spatial feature. We embed the bias of this site into the sub-Q-value. For each site, we use a two-layer fully-connected neural network to train the input of the offset value b as s_{jt}, a_{jt} and the current state $F(s_t, r_j)$ of the two nearest neighbors. We can use the following formula to estimate the Q-value:

$$\sum_{j=1}^{n}\left[Q^{j}_{\mu_j}(s_{jt}, a_{jt}) + b_j(s_{jt}, a_{jt}, F(s_t, r_j))\right] \tag{4}$$

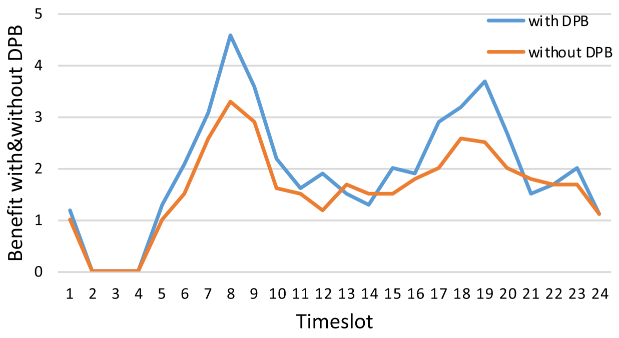

Figure 6 Benefit with and without the DPB method

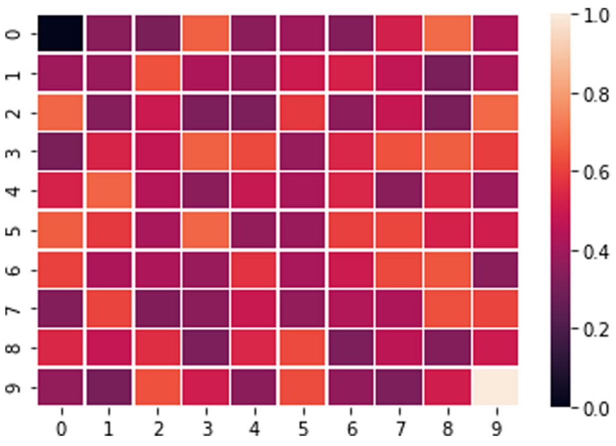

Figure 7 Vehicles distribution after morning peak

The proposed network architecture to accelerate the search process in a large state space and a large action space is shown in Figure 4.

5 Experiment

5.1 Dataset

We generate a simulation data set based on the real traffic data set provided by MOBIKE Cup Algorithm Challenge and apply it to test our DPB method. We sample the data of bikes in Shanghai by stratified sampling at a proportion of 15%. The data set contains data of up to one month from February 1 to 28, 2019. The source of the data is the simulated needs of users in Shanghai. Each record contains the following information: order ID, car ID, user ID, time, start station ID, and end station ID. (After desensitization, each station ID is represented by an index). The total number of records in the data set reaches one million.

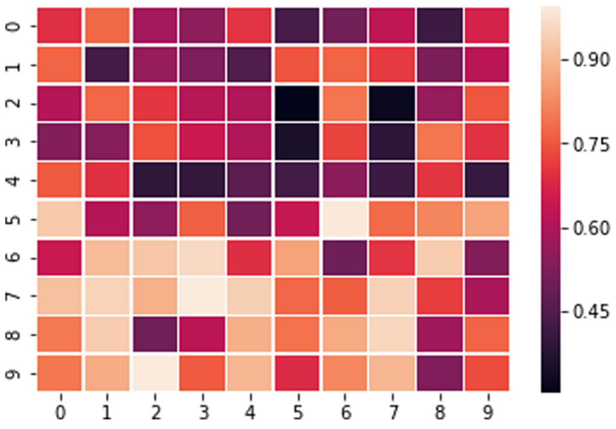

Figure 8 Vehicles distribution before evening peak

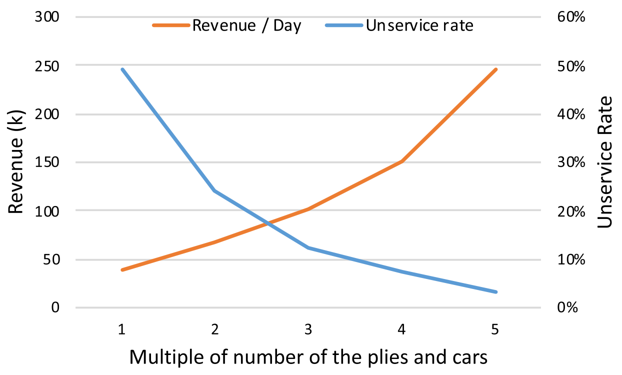

Figure 9 Unservice rate and revenue under varying supply

The specific user demand distribution is shown in Figure 5. We can find that the initial demand distribution is significantly normal when the the DPB method is not used. We will perform our method in the system later.

5.2 Experiment setup

The detailed description of the experiment on the DDPG algorithm is presented as follows.

5.2.1 Simulator

Based on the above data set, we obtained the time and space information of the user requesting to use the cars for a certain period of time. To ensure generality, we assumed that the demand curve follows the same pattern on weekdays. Each day is divided into 24 time periods, with each period lasting 1 hour. We have attempted to accept every request from the users, but if the number of parking spots at a user's star point or the number of vacancies at the parking spot fails to meet the user's need, the user's request will be unfortunately declined.

The user's response to the system will be the same as that given in Problem Description. The parking bonus of any station ranges from 0 to 5 RMB for a certain period of time, and the range of bonus for users to pick up cars within 5 minutes is also from 0 to 5 RMB. In our simulation process, while picking up a car, the user decides whether to improve his/her travel efficiency for the bonus according to Schema 1 (see the user model in Problem Description). While parking the rent car, the user selects the parking point of the highest reward within 2 km of his selected destination as his target parking spot.

5.2.2 Configurations

For DPG Model, we set a service rate threshold R_{Ser} of 50%; the expected picking time T is 5 minutes, and the parking distance Dis is 2 km. For the DDPG algorithm, we set the discount factor $\gamma = 0.99$; the target-network update rate is 10^{-3}, and the Adam algorithm [12] is used to train the Actor-Critic network. The learning rate of the two networks is 10^{-4}. We use the actor network to increase the Gaussian noise to the output of the action [13]. In order to improve the convergence performance, we use batch normalization [22], target-network [16] and experience replay [13, 17]. Our training algorithm is set to 50 episodes

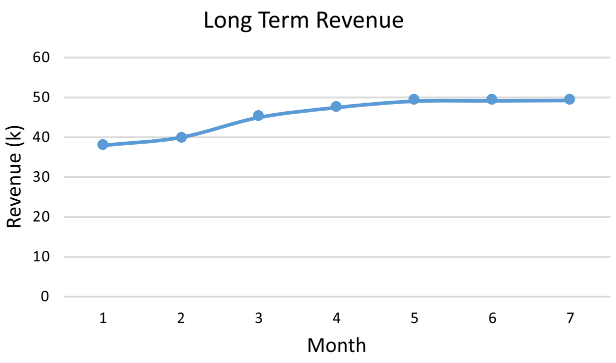

Figure 10 Revenue in long-term perspective

each time, and each episode includes $24N$ steps (N indicates the number of days). After that, we use 10 episodes for testing.

5.3 Baselines

The main differences observed in the service provider's income levels with or without the DPB method are highlighted as follows:

5.3.1 Service rate

When we do not use the reward mechanism, the service rate in the morning and evening peak periods is lower than that in other time periods because the number of orders varies across 24 hours a day. However, after adopting the price strategy of our DPB method, the service rate is found to significantly improve in both the morning and evening peak hours. The service rate reaches 50% of the target and increases by two percentage points.

5.3.2 Income situation

When the DPB method is not employed, each car cannot be used more than three times a day as the distribution of the cars cannot meet the users' demand. Therefore, if the distribution problem of cars remains unresolved, it is impossible to increase the revenue of the service providers. However, after the adjustment of the price using the DPB method, the position where users select to pick up and return cars is more manageable. As a result, the distribution of cars is more uniform and natural owing to the use of price leverage. Therefore, the service provider's revenue will also improve because of increased user satisfaction. The specific situation is described in the following paragraph.

As shown in Figure 6, it is noted that though during certain time periods, the overall benefit of using the DPB method does not outweigh that achieved without the use of this method, there is an overall increase in the users' service rates. To meet our goal (to minimize the cost/to maximize the benefit when the user service rate is 50%), the service provider's revenue may not be greater than before in the short term. However, from a longer-term perspective, we have cultivated user behavior and increased user stickiness, which will help boost the service provider's long-term benefits.

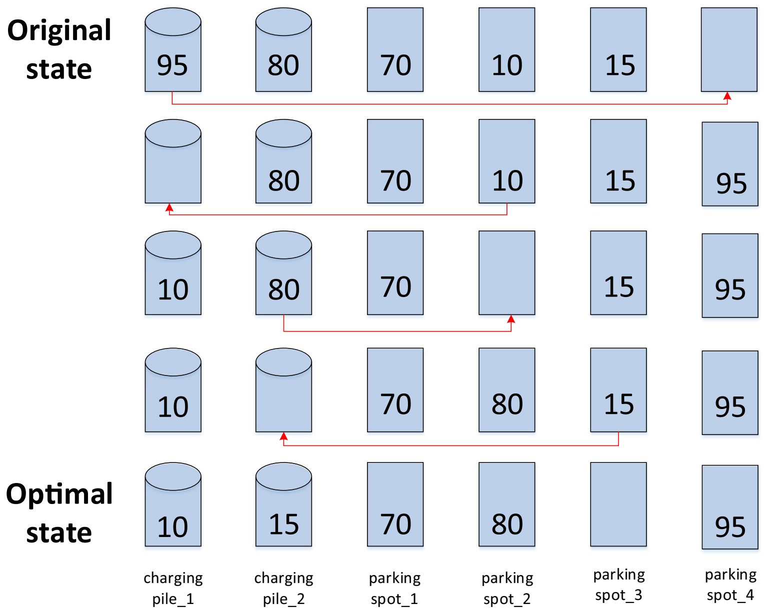

Figure 11 An example of inner-station scheduling

5.3.3 Vehicle distribution

The distribution of cars becomes more scientific with adjustment through the DPB method in comparison to that without the method. Take the distribution of cars that have just ended at the morning peak at 10:00 a.m. on January 24, 2019, as an example. The distribution of cars managed by employing the DPB method is shown in Figure 7.

When the DPB method is employed, an improvement is noted not only in users' service rate, but also in the uniformity of car distribution.

Furthermore, by analyzing the car distribution before the evening peak hours on the same day at 3:00 p.m. on January 24, 2019, it is found that the DPB method had learned to prepare for the large traffic flow before the evening peak hours as there were more cars in the work areas and fewer cars in the living areas.

As vividly shown in Figure 8, the aim of adopting the DPB method is not only to achieve uniform car distribution, but also to prepare for the minimization of expenses/expansion of the revenue.

5.4 Performance comparison

5.4.1 Varying supply

At present, there are a total of n stations, each with 10 charging piles and 5 cars at the initial moment. If we double the number of charging piles per station and the initial number of cars, the DPB method will have more room for operation and higher fault tolerance. Figure 9 shows how the changes in the user's daily unservice rate and the service provider's revenue are correlated to the supply of cars. Notably, the user's unservice rate and the supply are inversely related, while the provider's revenue and the supply are positively correlated.

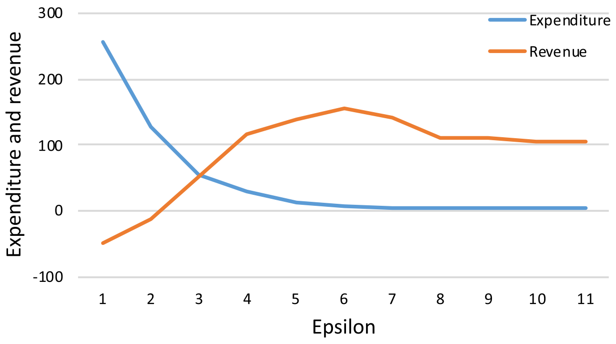

Figure 12 Expenditure and revenue for different values of ϵ

5.4.2 Long-time performance

As analyzed above, the system provides a number of rewarding mechanisms, including parking bonuses and picking bonuses, to reach a user satisfaction rate of 50%. The user service rate is increased at the expense of a large cost. At times the income from a single order may not cover the given reward (such an order is defined as a loss); however, if we adjust the car distribution, we can cultivate user habits and increase user stickiness so that the order, in the long run, can garner significant profit to the car-sharing service provider, including but not limited to an increase in user loyalty and improvement in user experience.

From a longer-term perspective, the revenue with the DPB method is as follows. As clearly shown in the Figure 10, the DPB method actually increases the car-sharing service provider's income and lowers the unservice rate. Further, from a longer-term perspective, the revenue will increase exponentially.

5.4.3 Inner-station scheduling

In previous experiments, we have always assumed that all shared cars are always in a fully charged state; however, this is not feasible in reality. To further increase the efficiency of the system, we take this variable into consideration. The car scheduling in the station determines whether each car is in a state of being charged and the current amount of power. The specific experimental improvements are given in the following paragraph.

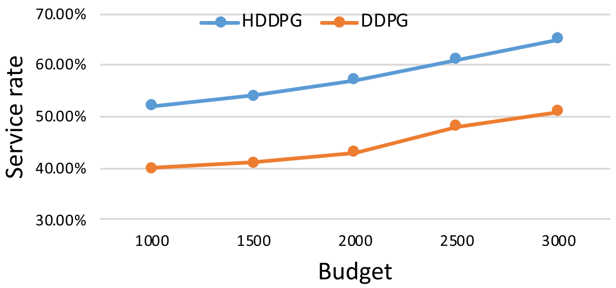

Figure 13 Service rates under various budget levels

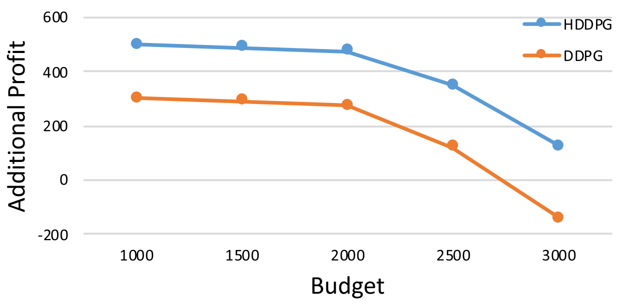

Figure 14 Additional profit under varying budget levels

Whenever a station's $station\_entropy$ is greater than the set threshold ϵ, an in-station scheduling request is issued to each user who arrives at the station (whether it is the user picking up the vehicle or the user who is returning the vehicle) until the request is accepted. Next, we assume that the user can complete the in-site scheduling tasks assigned by the system within the appropriate time and receive a certain cash reward according to the level of difficulty in scheduling.

As shown in Figure 11, the cylinders represent charging piles and rectangles represent the parking spots. The number labels indicate the remaining battery power of the car parking here. As observed, it takes four steps to transition from the original state to an optimal state. Therefore, we need to pay the user who is willing to accept the inner-station scheduling request for 20 RMB.

Moreover, ϵ is an important hyper-parameter that can affect the performance of the entire system because the value of ϵ decides the frequency of the inner-station scheduling request that will be sent to the user. For example, if $\epsilon = 1$, we need to swap the position of two cars even if the remaining battery power of those two cars are equal. If ϵ is a very large number, we hardly need to swap anything, which implies that the cars that are being charged will be charging forever and cars that are not charged will never be charged. Such a situation is depicted in Figure 12.

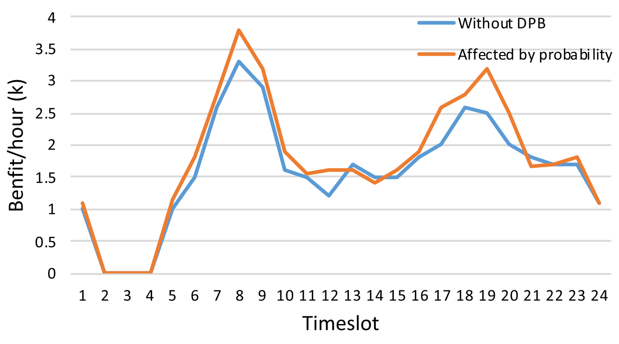

Figure 15 Benefit for parking affected by probability

5.4.4 Hierarchical mechanism

Figure 13 compared the changes in service rates with regard the HDDPG algorithm and the DDPG algorithm under different budgets. It can be clearly seen that both the algorithms can continuously improve the service rate with an increase in the budget levels. However, the service rate using the HDDPG algorithm under the same budget is always higher than that using the DDPG algorithm.

Figure 14 compared the changes in additional income brought to merchants of the HDDPG and DDPG algorithms under different budget levels. It can be observed that as the budget level increases, the additional income of the merchants will continue to decrease - decreasing gradually in the early stage and then drastically after passing a certain threshold. Therefore, it is expected that when the merchant's budget is set to this threshold, there will be a significant increase in the additional income. Moreover, it is evident that the additional revenue from the HDDPG algorithm is always higher than that from the DDPG algorithm.

5.4.5 The user selects a parking spot according to the probability

In reality, while choosing the parking options, the user does not necessarily pick the highest reward spot within 2 km. Depending on the urgency of the user's arrival at the destination, the user can choose whether to stop at the most rewarded parking spot according to a certain probability. Although not every parking action occurs at the spot with the most rewards within 2 km, employing the DPB model is still more profitable for merchants than choosing to operate without the model, as evident in Figure 15.

6 Conclusion

In summary, we propose a new method, the DPB method, to resolve the problems faced by car-sharing service providers. This method uses reinforcement learning to schedule the cars in the car-sharing system and achieves remarkable results. A series of comparative experiments show that the rewarding mechanism of the DPB method can indeed guide user behaviors through price leverage, increase user stickiness, cultivate user habits, and therefore boost the service provider's profits in the long run. In addition to inter-station scheduling, we investigated in-station scheduling. This type of scheduling can maintain the power of new energy vehicles within a reasonable range as much as possible. This is significant because only then can we guarantee that the shared car received by the users can be reused immediately.

The DPB method has its own limitations. For instance, it does not provide a solution for maintaining high cash rewards after an increase in user loyalty. Following is another issue that needs to be resolved: how to increase the service provider's revenue without degrading user experience? In future studies, we will continue to investigate the abovementioned limitations of the DPB method and attempt to achieve an improvement in user experience and increase in the service provider's revenue.

Acknowledgments This work is supported in part by National Key R&D Program of China Under Grant No. 2018YFB1004003 and the National Natural Science Foundation of China under Grant No. U1636215.

References

1. Andre, D., Russell, S.J.: State abstraction for programmable reinforcement learning agents[C]. AAAI/IAAI, pp. 119–125 (2002)
2. Cai, Q., Filos-Ratsikas, A., Tang, P., et al.: Reinforcement Mechanism Design for e-commerce[C]. Proceedings of the 2018 World Wide Web Conference. International World Wide Web Conferences Steering Committee, pp. 1339–1348 (2018)
3. Chemla, D., Meunier, F., Pradeau, T., Calvo, R.W., Yahiaoui, H.: Self-service bike sharing systems: simulation, repositioning pricing (2013)
4. Dayan, P., Hinton, G.E.: Feudal reinforcement learning[C]//Advances in neural information processing systems, pp. 271–278 (1993)
5. Dean, T., Lin, S.H.: Decomposition techniques for planning in stochastic domains[C]. IJCAI **2**, 3 (1995)
6. Dietterich, T.G.: The MAXQ Method for Hierarchical Reinforcement Learning[C]. ICML **98**, 118–126 (1998)
7. Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition[J]. J. Artif. Intell. Res. **13**, 227–303 (2000)
8. Fricker, C., Gast, N.: Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. Euro J. Transp. Logist. **5**(3), 261–291 (2016)
9. Ghosh, S., Trick, M., Varakantham, P.: Robust Repositioning to Counter Unpredictable Demand in Bike Sharing Systems. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16), pp. 3096–3102. AAAI Press. http://dl.acm.org/citation.cfm?id=3061053.3061055 (2016)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory[J]. Neural Comput. **9**(8), 1735–1780 (1997)
11. Kaelbling, L.P.: Hierarchical learning in stochastic domains: Preliminary results[C]. Proc. Tenth Int. Conf. Mach. Learn. **951**, 167–173 (1993)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
13. Lillicrap, T.P., Hunt, J.J., Pritzel, A., et al.: Continuous control with deep reinforcement learning. Comput. Sci. **8**(6), A187 (2016)
14. Li, Y., Yu, Z., Yang, Q.: Dynamic Bike Reposition: A SpatioTemporal Reinforcement Learning Approach. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1724–1733. ACM (2018)
15. Liu, J., Sun, L., Chen, W., Xiong, H.: Rebalancing Bike Sharing Systems: A Multi-source Data Smart Optimization. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1005–1014. ACM (2016)
16. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Playing Atari with deep reinforcement learning. Proceedings of Workshops at the 26th Neural Information Processing Systems 2013. Lake Tahoe, pp. 201–220 (2013)
17. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
18. Ning, W., Wenjian, Z., Xiang, L., Jing, Z.: Inter-Site-Vehicle Artificial scheduling strategy design for electric vehicle Sharing[J]. J. Tongji Univ. (Nat. Sci.) **46**(8), 1064–1071 (2018)
19. Ning, W., Yajing, S., Linhao, T., WenJian, Z.: Adaptive Scheduling Strategy in Car-sharing System Based on Feedback Dynamic Pricing. J. Transp. Syst. Eng. Inf. Technol. **18**(5), 12–17 (2018)
20. O'Mahony, E., Shmoys, D.B.: Data analysis and optimization for (citi) bike sharing. In: AAAI, pp. 687–694 (2015)
21. Pan, L., Cai, Q., Fang, Z., et al.: A Deep Reinforcement Learning Framework for Rebalancing Dockless Bike Sharing Systems[J]. arXiv:1802.04592 (2018)
22. Sergey, I., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167 (2015)
23. Silver, D., Lever, G., Hess, N., et al.: Deterministic policy gradient algorithms. Proceedings of the International Conference on Machine Learning. Beijing, pp. 387–395 (2014)
24. Sutton, R.S., Barto, AG.: Reinforcement learning: an Introduction. MIT Press, Cambridge (1998)
25. Sutton, R.S., McAllister, D.A., Singh, S.P., et al.: Policy gradient methods for reinforcement learning with function approximation. Proceedings of the Advances in Neural Information Processing Systems, Denver, pp. 1057–1063 (1999)
26. Singla, A., Santoni, M., ok, G.abor.B., Mukerji, P., Meenen, M., Krause, A.: Incentivizing users for balancing bike sharing systems. In: AAAI, pp. 723–729, Austin, Texas (2015)

27. Van Seijen, H. et al.: Hybrid reward architecture for reinforcement learning. Advances in Neural Information Processing Systems (2017)
28. Watkins, C.J.C.H.: Learning from delayed rewards. Robot. Auton. Syst. **15**(4), 233–235 (1989)

Affiliations

Changwei Ren[1] · Lixingjian An[1] · Zhanquan Gu[2] · Yuexuan Wang[1,3] · Yunjun Gao[1]

Changwei Ren
rcw@zju.edu.cn

Lixingjian An
alxj2014@zju.edu.cn

Yuexuan Wang
amywang@zju.edu.cn

Yunjun Gao
gaoyj@zju.edu.cn

[1] College of Computer Science, Zhejiang University, Hangzhou, China
[2] Cyberspace Institute of Advanced Technology (CIAT), Guangzhou University, Guangzhou, China
[3] Department of Computer Science, The University of Hong Kong, Hongkong, China