A Deep Reinforcement Learning Framework for Rebalancing Dockless Bike Sharing Systems[*]

Ling Pan,[1] Qingpeng Cai,[1] Zhixuan Fang,[2] Pingzhong Tang,[1] Longbo Huang[1]
[1]IIIS, Tsinghua University
[2]The Chinese University of Hong Kong
{pl17, cqp14}@mails.tsinghua.edu.cn, zxfang@ie.cuhk.edu.hk, {kenshin, longbohuang}@tsinghua.edu.cn

Abstract

Bike sharing provides an environment-friendly way for traveling and is booming all over the world. Yet, due to the high similarity of user travel patterns, the bike imbalance problem constantly occurs, especially for dockless bike sharing systems, causing significant impact on service quality and company revenue. Thus, it has become a critical task for bike sharing operators to resolve such imbalance efficiently. In this paper, we propose a novel deep reinforcement learning framework for incentivizing users to rebalance such systems. We model the problem as a Markov decision process and take both spatial and temporal features into consideration. We develop a novel deep reinforcement learning algorithm called Hierarchical Reinforcement Pricing (HRP), which builds upon the Deep Deterministic Policy Gradient algorithm. Different from existing methods that often ignore spatial information and rely heavily on accurate prediction, HRP captures both spatial and temporal dependencies using a divide-and-conquer structure with an embedded localized module. We conduct extensive experiments to evaluate HRP, based on a dataset from Mobike, a major Chinese dockless bike sharing company. Results show that HRP performs close to the 24-timeslot look-ahead optimization, and outperforms state-of-the-art methods in both service level and bike distribution. It also transfers well when applied to unseen areas.

Introduction

Bike sharing, especially dockless bike sharing, is booming all over the world. For example, Mobike, a Chinese bike-sharing giant, has deployed over 7 million bikes in China and abroad. Being an environment-friendly approach, bike sharing provides people with a convenient way for commuting by sharing public bikes among users, and solves the "last mile" problem (Shaheen, Guzman, and Zhang 2010). Different from traditional docked bike sharing systems (BSS), e.g., Hubway, where bikes can only be rented and returned at fixed docking stations, users can access and park sharing bikes at any valid places. This relieves users' concerns about finding empty docks when they want to use bikes, or getting into fully occupied stations when they want to return them.

However, due to similar travel patterns of most users, the rental mode of BSS leads to bike imbalance, especially during rush hours. For example, people mostly ride from home to work during morning peak hours. This results in very few bikes in residential areas, which in turn suppresses potential future demand, while subway stations and commercial areas are paralyzed due to the overwhelming number of shared bikes. This problem is further exaggerated for dockless BSS, due to unrestrained users' parking locations. This imbalance can cause severe problems not only to users and service providers, but also to cities. Therefore, it is crucial for bike sharing providers to rebalance bikes efficiently, so as to serve users well and to avoid congesting city sidewalks and causing a bike mess.

Bike rebalancing faces several challenges. First, it is a resource-constrained problem, as service providers often pose limited budgets for rebalancing the system. Naively spending the budget to increase the supply of bikes will not resolve the problem and is also not cost-efficient. Moreover, the number of bikes allowed is often capped due to regulation. Second, the problem is computationally intractable due to the large number of bikes and users. Third, the user demand is usually highly dynamic and changes both temporally and spatially. Fourth, if users are also involved in rebalancing bikes, the rebalancing strategy needs to efficiently utilize the budget and incentivize users to help, without knowing users' private costs.

There have been a considerable set of recent results on bike rebalancing, which mainly focuses on two approaches, i.e., the vehicle-based approach (O'Mahony and Shmoys 2015; Liu et al. 2016; Ghosh, Trick, and Varakantham 2016; Li, Zheng, and Yang 2018; Ghosh and Varakantham 2017) and the user-based approach (Singla et al. 2015; Chemla et al. 2013; Fricker and Gast 2016). The vehicle-based approach utilizes multiple trucks/bike-trailers to achieve repositioning by loading or unloading bikes in different regions. However, its rebalancing effect depends heavily on the accuracy of demand prediction. Also, it can be inflexible as it is hard to adjust the repositioning plan in real time to cater to the fluctuating demand. Additionally, due to the mainte-

[*]The work of Longbo Huang and Ling Pan was supported in part by the National Natural Science Foundation of China Grants 61672316, 61303195, the Tsinghua Initiative Research Grant, and the China Youth 1000-Talent Grant. Pingzhong Tang and Qingpeng Cai were supported in part by the National Natural Science Foundation of China Grant 61561146398, a China Youth 1000-talent program and an Alibaba Innovative Research program.

nance and traveling costs of trucks, as well as labor costs, the truck-based approach can deplete the limited budget rapidly. In contrast, the user-based approach offers a more economical and flexible way to rebalance the system, by offering users monetary incentives and alternative bike pick-up or drop-off locations. In this way, users are motivated to pick up or return bikes in neighboring regions rather than regions suffering from bike or dock shortage. However, existing user-based approaches often do not take the spatial information, including bike distribution and user distribution, into account in the incentivizing policy. Moreover, user related information, e.g., costs due to walking to another location, is also often unknown.

In this paper, we propose a deep reinforcement learning framework for incentivizing users to rebalance dockeless BSS, as shown in Figure 1. Specifically, we view the problem as interactions between a bike sharing service operator and the environment, and formulate the problem as a Markov decision process (MDP). In this MDP, a state consists of supply, demand, arrival, and other related information, and each action corresponds to a set of monetary incentives for each region, to incentivize users to walk to nearby locations and use bikes there. The immediate reward is the number of satisfied user requests. Our objective is to maximize the long-term service level, i.e., the total number of satisfied requests, which is of highest interest for bike sharing service providers (Lin and Yang 2011). Our approach falls under the topic of incentivizing strategies in multi-agent systems (Xue et al. 2016; Tang 2017; Cai et al. 2018).

To tackle our problem, we develop a novel deep reinforcement learning algorithm called the *hierarchical reinforcement pricing* (HRP) algorithm. The HRP algorithm builds upon the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al. 2015), using the general hierarchical reinforcement learning framework (Dietterich 2000). Our idea is to decompose the Q-value of the entire area of interest into multiple sub-Q-values of smaller regions. The decomposition enables an efficient searching for policies, as it addresses the complexity issue due to high-dimensional input space and temporal dependencies. In addition, the HRP algorithm also takes spatial dependencies into consideration and contains a localized module, in order to correct the bias in Q-value function estimation, introduced by decomposition and correlations among sub-states and sub-actions. Doing so reduces the input space and decreases the training loss. We also show that the HRP algorithm improves convergence and achieves a better performance compared with existing algorithms.

The main contributions of our paper are as follows:

- We propose a novel spatial temporal bike rebalancing framework, and model the problem as a Markov decision process (MDP) that aims at maximizing the service level.
- We propose the *hierarchical reinforcement pricing* (HRP) algorithm that decides how to pay different users at each time, to incentivize them to help rebalance the system.
- We conduct extensive experiments using Mobike's dataset. Results show that HRP drastically outperforms state-of-the-art methods. We also validate the optimality of HRP by comparing with our proposed *offline-optimal* algorithm. We further demonstrate HRP's generalization ability over different areas.

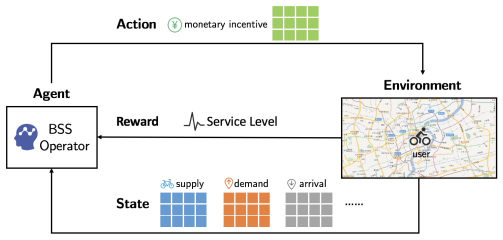


Figure 1: An overview of the deep reinforcement learning framework for rebalancing dockless bikesharing systems.

Related Work

Rebalancing Approaches. With the recent development of BSS, researchers have started to study operational issues leveraging big data (Liu et al. 2017; Yang et al. 2016; Chen et al. 2016; Li et al. 2015), among which rebalancing is one of the most important focuses. Rebalancing approaches can be classified into three categories. The first category adopts the truck-based approach which employs multiple trucks. These methods can reposition bikes either in a static (Liu et al. 2016) or dynamic (Ghosh, Trick, and Varakantham 2016) way, for docked BSS. The second category focuses on the use of bike-trailers (O'Mahony and Shmoys 2015; Ghosh and Varakantham 2017; Li, Zheng, and Yang 2018). The third category focuses on the user-based rebalancing approach (Singla et al. 2015; Chemla et al. 2013; Fricker and Gast 2016).

Reinforcement Learning. Deep Deterministic Policy Gradient algorithm (DDPG) (Lillicrap et al. 2015) builds upon the Deterministic Policy Gradient algorithm (Silver et al. 2014), using deep neural networks to approximate the action-value function for improving convergence. However, conventional reinforcement learning methods cannot scale up to problems with high-dimensional input spaces. Hierarchical reinforcement learning (Dayan and Hinton 1993) 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 sub-MDP. Thus, the sub-agent can neglect part of the state which is irrelevant to its current decision (Andre and Russell 2002; Van Seijen et al. 2017) to enable faster learning.

Problem Definition

In this section, we introduce and formalize the user-based bike rebalancing problem, i.e., by offering users monetary incentives to motivate them to help rebalance the system.

Consider an area spatially divided into n regions, i.e., $\{r_1, r_2, ..., r_n\}$. We discretize a day into T timeslots with equal length, denoted by $\mathcal{T} = \{1, 2, ..., T\}$. Let $S_i(t)$ denote the supply in region r_i at the beginning of timeslot

$t \in \mathcal{T}$, i.e., the number of available bikes. We also denote $\boldsymbol{S}(t) = (S_i(t), \forall i)$ as the vector of supply. The total user demand and bike arrival of region r_i during the timeslot t is denoted by $D_i(t)$ and $A_i(t)$ respectively. We similarly denote $\boldsymbol{D}(t) = (D_i(t), \forall i)$ and $\boldsymbol{A}(t) = (A_i(t), \forall i)$ as the vector of demand and the vector of arrival, respectively. Let $d_{ij}(t)$ denote the number of users intending to ride from region r_i to region r_j during the timeslot t.

Pricing Algorithm. At each timeslot t, for a user who cannot find an available bike in his current region r_i, a pricing algorithm $\mathcal{A}$ suggests him alternate bikes in r_i's neighboring regions, denoted by $N(r_i)$. Meanwhile, $\mathcal{A}$ also offers the user a price incentive $p_{ij}(t)$ (in the order of user arrivals), to motivate him to walk to neighboring region $r_j \in N(r_i)$ to pick up a bike b_{hj}, where b_{hj} denotes the h-th bike in r_j. $\mathcal{A}$ has a total rebalancing budget B. When the budget is completely depleted, $\mathcal{A}$ can no longer make further provision.

User Model. For each user u_k in region r_i, if there are available bikes in the current region, he takes the nearest one. Otherwise, there is a walking cost if he walks from r_i to a neighboring region r_j to pick up bikes. We denote the cost by $c_k(i,j,x)$, where x is the walking distance from his location to the bike. We assume that it has the following form:

$$c_k(i,j,x) = \begin{cases} 0 & i \text{ equals to } j \\ \alpha x^2 & r_j \text{ is a neighboring region of } r_i \\ +\infty & \text{else} \end{cases}. \quad (1)$$

This particular form of $c_k(i,j,x)$ is motivated by a survey conducted in (Singla et al. 2015), where it is shown that user cost has a convex structure. Note that this cost is private to each user and the cost function is unknown to the service provider. For a user who cannot find an available bike in his current region r_i, if he receives an offer $(p_{ij}(t), b_{hj})$ and accepts it, he obtains a utility $p_{ij}(t) - c_k(i,j,x)$. Thus, a user will choose and pick up the bike b_{hj} where he can obtain the maximum utility, and collect the price incentive $p_{ij}(t)$. If no offer leads to a nonnegative utility, the user will not accept any of them, resulting in an unsatisfied request.

Bike Dynamics. Let $x_{ijl}(t)$ denote the number of users in r_i riding to r_l by taking a bike in r_j at timeslot t. The dynamics of supply in each region r_i can be expressed as:

$$S_i(t+1) = S_i(t) - \sum_{j=1}^{n}\sum_{l=1}^{n} x_{jil}(t) + \sum_{m=1}^{n}\sum_{j=1}^{n} x_{mji}(t). \quad (2)$$

The second and third terms in Eq. (2) denote the numbers of departing and arriving bikes in region r_i. Note that there is a travel time for bikes travel among regions.

Objective. Our goal is to develop an optimal pricing algorithm $\mathcal{A}$ to incentivize users in congested regions to pick up bikes in neighboring regions, so as to maximize the service level, i.e., the total number of satisfied requests, subject to the rebalancing budget B.

Hierarchical Reinforcement Pricing Algorithm

In this section, we present our *hierarchical reinforcement pricing* (HRP) algorithm that incentivizes users to rebalance the system efficiently.

MDP Formulation

Our problem is an online learning problem, where an agent interacts with the environment. Therefore, it can be modeled as a Markov decision process (MDP) defined by a 5-tuple (S, A, Pr, R, γ), where S and A denote the set of states and actions, Pr the transition matrix, R the immediate reward and γ the discount factor. In our problem, at each timestep t, the **state** $s_t = (\boldsymbol{S}(t), \boldsymbol{D}(t-1), \boldsymbol{A}(t-1), \boldsymbol{E}(t-1), RB(t), \boldsymbol{U}(t))$. Here, $\boldsymbol{S}(t)$ is the current supply for each region while $RB(t)$ is the current remaining budget. $\boldsymbol{D}(t-1)$, $\boldsymbol{A}(t-1)$ and $\boldsymbol{E}(t-1)$ are the demand, arrival and the expense in the last timestep for each region. $\boldsymbol{U}(t)$ represents the un-service rate for each region for a fixed number of past timesteps. The bike sharing service operator takes an **action** $a_t = (p_{1t}, ..., p_{nt})$, and receives an **immediate reward** $R(s_t, a_t)$ which is the number of satisfied requests in the whole area R at timestep t. Specifically, p_{it} represents the price for region r_i at timestep t.[1] $Pr(s_{t+1}|s_t, a_t)$ represents the **transition probability** from state s_t to state s_{t+1} under action a_t. The **policy function** $\pi_\theta(s_t)$ with the parameter θ, maps the current state to a deterministic action. The overall **objective** is to find an optimal policy to maximize the **overall discounted rewards** from state s_0 following π_θ, denoted by $J_{\pi_\theta} = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R(a_k, s_k)|\pi_\theta, s_0]$, where $\gamma \in [0,1]$ denotes the **discount factor**. The Q-value of state s_t and action a_t under policy π_θ is denoted by $Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}[\sum_{k=t}^{\infty} \gamma^{k-t} R(s_k, a_k)|\pi_\theta, s_t, a_t]$. Note that J_{π_θ} is a discounted version of the targeting service level objective, and will serve as a close approximation when γ is close to 1. Indeed, our experimental results show that with $\gamma = 0.99$, our algorithm performs very close to the offline optimal of the service level objective, demonstrating the effectiveness of this approach.

The HRP Algorithm

One of the key challenges in our problem, is that it has a continuous and high-dimensional action space that increases exponentially with the number of regions and suffers from the "curse of dimensionality". To tackle this problem, we propose the HRP algorithm, which is shown in Algorithm 1. Inspired by hierarchical reinforcement learning and DDPG, the HRP algorithm, which captures both temporal and spatial dependencies, is able to address the convergence issue of existing algorithms and improve performance.

Specifically, we decompose the Q-value of the whole area into the sub-Q-value of each region. Then, the Q-value can be estimated by the additive combination of estimators of sub-Q-values according to $\sum_{j=1}^{n} Q_{\mu_j}^{j}(s_{jt}, p_{jt})$, where s_{jt}, p_{jt} denote the sub-state and sub-action of region r_j at timestep t, and μ_j corresponds to the parameter of the estimator. For each timestep, the current state depends on previous states as people pick up and return bikes dynamically. To capture the sequential relationships exhibit in states, one idea is to train each sub-agent using Long

[1]To reduce the complexity of the MDP, we employ the policy that the monetary incentives a user in region r_i receives for picking up bikes in neighboring regions are the same. Thus, we use p_{it} in place of $p_{ij}(t_k)$ for ease of notations.

Algorithm 1 The Hierarchical Reinforcement Pricing (HRP) algorithm.

```
Require: Randomly initialize weights $\theta, \mu$ for the actor network $\pi_\theta(s)$ and the critic network $Q_\mu(s,a)$.
  Initialize target actor network $\pi'$ and target critic network $Q'$ with weights $\theta' \leftarrow \theta, \mu' \leftarrow \mu$
  Initialize experience replay buffer $\mathcal{B}$ by filling with samples collected from the warm-up phase
  for episode $= 1, ..., M$ do
      Initialize a random process $\mathcal{N}$ to explore the action space, e.g. Gaussian noise, and receive initial state $s_1$
      for step $t = 1, ..., T$ do
          ▷ Explore and sample
          Select and execute action $a_t = \pi_\theta(s_t) + \mathcal{N}_t$ ($\mathcal{N}_t$ sampled from $\mathcal{N}$), observe reward $R_t$ and next state $s_{t+1}$
          Store transition $(s_t, a_t, R_t, s_{t+1})$ in experience replay buffer $\mathcal{B}$ //update experience replay buffer $\mathcal{B}$
          Sample a random minibatch of $N$ transitions $(s_i, a_i, R_i, s_{i+1})$ from $\mathcal{B}$
          ▷ Get current state-action pair's Q-value
          Compute the decomposed Q-values $Q^j_{\mu_j}(s_{ji}, p_{ji})$ for each region $r_j$
          Compute the bias-correction term $f_j(s_{ji}, NS(s_i, r_j), p_{ji})$ for each region $r_j$ by the localized module
          Compute current state-action pair's Q-value $Q_\mu(s_i, a_i)$ according to Eq. (3)
          ▷ Get next state-action pair's Q-value
          Get action for next state by actor network: $a'_{i+1} = \pi'_{\theta'}(s_{i+1})$
          Compute the decomposed Q-values $Q^j_{\mu'_j}(s_{j(i+1)}, p'_{j(i+1)})$ for each region $r_j$
          Compute the bias-correction term $f_j(s_{j(i+1)}, NS(s_{i+1}, r_j), p'_{j(i+1)})$ for each region $r_j$ by the localized module
          Compute next state-action pair's Q-value $Q'_{\mu'}(s_{i+1}, a'_{i+1})$ according to Eq. (3)
          ▷ Update
          Set $y_i = R_i + \gamma Q'_{\mu'}(s_{i+1}, a'_{i+1})$
          Update the critic by minimizing the loss according to Eq. (5)
          Update the actor using the sampled policy gradient according to Eq. (4)
          Update the target networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta', \mu' \leftarrow \tau\mu + (1-\tau)\mu'$
      end for
  end for
```

Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) unit, which can capture complex sequential dependencies. However, LSTM maintains a complex architecture and needs to train a substantial number of parameters. Note that a key challenge in hierarchical reinforcement learning is to estimate accurate sub Q-values, which leads to an accurate overall Q-value estimation. Thus, we adopt the Gated Recurrent Unit (GRU) model (Cho et al. 2014), a simplified variant of LSTM. Such a structure is more condensed and has fewer parameters, which enables a higher sample efficiency and avoids overfitting.

However, applying a direct decomposition can lead to a large bias due to the dependence of each region and its neighbors. In our case, users may pick up bikes in neighboring regions besides their current regions, resulting in the fact that actions in different regions are coupled. Therefore, we need to take this domain spatial feature into consideration. We tackle the bias of $Q_\mu(s_t, a_t)$ and $\sum_{j=1}^n Q^j_{\mu_j}(s_{jt}, p_{jt})$ by embedding the localized module to incorporate the spatial information. For each region r_j, the localized module (represented by f_j) takes not only the state s_{jt} and action p_{jt}, but also the states of its neighboring regions denoted by $NS(s_t, r_j)$ as inputs. In particular, f_j is approximated by a neural network consisting of two fully-connected layers for bias correction. Thus, we estimate $Q_\mu(s_t, a_t)$ by:

$$\sum_{j=1}^{n} Q^j_{\mu_j}(s_{jt}, p_{jt}) + f_j(s_{jt}, NS(s_t, r_j), p_{jt}). \tag{3}$$

The network architecture we propose to accelerate the search process in a large state space and a large action space is shown in Figure 2. Formally speaking, the key components of the HRP algorithm are described below:

The Actor: The actor network represents the policy π_θ parameterized by θ. It maximizes J_{π_θ} using stochastic gradient ascent. In particular, the gradient of J_{π_θ} over θ is given by:

$$\nabla_\theta J_{\pi_\theta} = \mathbb{E}_{s\sim\rho_{\pi_\theta}}[\nabla_\theta \pi_\theta(s) \nabla_a Q_\mu(s,a)|_{a=\pi_\theta(s)}], \tag{4}$$

where ρ_{π_θ} denotes the distribution of states.

The Critic: The critic network takes the state s_t and action a_t as input, and outputs the action value. Specifically, the critic approximates the action-value function $Q^{\pi_\theta}(s,a)$ by minimizing the following loss (Lillicrap et al. 2015):

$$L(Q^{\pi_\theta}) = \mathbb{E}_{s_t\sim\rho_\pi, a_t\sim\pi}[(Q_\mu(s_t, a_t) - y_t)^2], \tag{5}$$

where $y_t = R(s_t, a_t) + \gamma Q^{'}_{\mu'}(s_{t+1}, \pi'_{\theta'}(s_{t+1}))$.

Experiments

Dataset

We make use of a Mobike dataset consisting of users' trajectories from August 1st to September 1st in 2016, in the city of Shanghai. Each data record contains the following information: order ID, bike ID, user ID, start time, start location (specified by longitude and latitude), end time, end location, and trace, with a total number of $102,361$ orders. Figure 3 shows the thermodynamic diagram of the dataset,

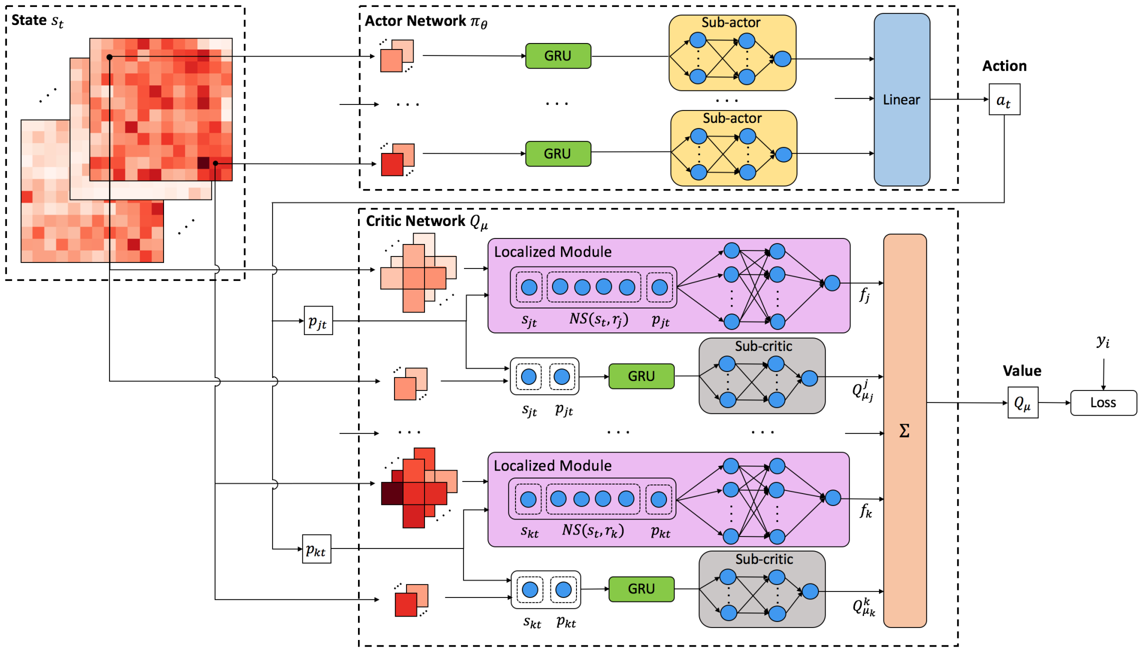
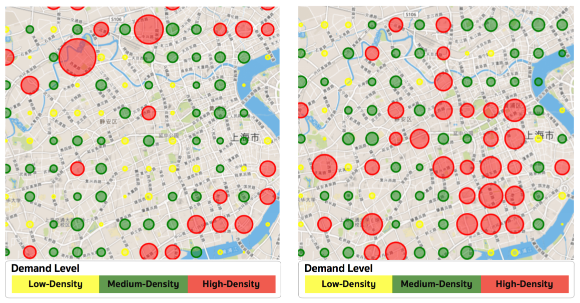


Figure 2: The actor-critic framework of the HRP algorithm.



(a) Demand in 8 a.m.-9 a.m. (b) Demand in 6 p.m.-7 p.m.

Figure 3: The temporal and spatial imbalance problem.

where different colors of circles represent different demand levels with the radius representing the number of demand. As shown, there exists a significant temporal and spatial imbalance.

Experimental Setup

According to Mobike's trajectory dataset, we obtain spatial and temporal information of users' requests and arrivals. We observe that the demand curve of different weekdays follows a very similar pattern according to the dataset, where the distribution during a day is bimodal with peaks corresponding to morning and evening rush hours. Thus, we aggregate all weekday demands to recover the total demand of a day. Each day consists of 24 timeslots, and each timeslot is 1 hour. We serve user requests every minute, where user requests are directly drawn from data while their starting locations (longitude and latitude) follow uniform distribution in their starting regions. To determine the initial distribution of bikes at the beginning of the day, we follow a common and practical way that most bike sharing operators (e.g. Mobike, Ofo) employ to redistribute bikes. The number of initial bikes in region r_i is set as the product of the total supply and the ratio of the total demand in r_i and the total demand for the whole area. Then, locations of initial bikes of each region are randomly drawn from data (from the locations of bikes in the corresponding region). Users respond to the system according to the defined user model, where the parameter α of the cost function as in Eq. (1) is selected so that the cost ranges in $[0\ RMB, 5\ RMB]$. This is chosen according to a survey conducted by (Singla et al. 2015), which shows that the cost ranges from 0 to 2 euros, and we converted it by the purchasing power of users. According to Mobike, the total number of orders in China is about 20 million, with the total supply, i.e., number of bikes, in China to be 3.65 million. Since we cannot obtain the total supply directly from data, we set the total supply as $O \times \frac{3.65}{20}$ (O is the number of orders in our system). As the initial supply affects the inherent imbalance degree in Mobike's system, we vary this number in our experiments to evaluate this effect. We use the first-order approximation to compute the number of the unobserved lost demand (O'Mahony and Shmoys 2015).

Configurations of the HRP Algorithm For the state representation, we choose the fixed number of past timesteps of $\boldsymbol{U}(t)$ to be 8. The comparison is fair as hyperparamters are the same for comparing reinforcement learning algorithms, following the configuration of (Lillicrap et al. 2015). We train the algorithm for 100 episodes in each setting, where each episode consists of $24N$ steps (N denotes the number of days). After that, 20 episodes are used for testing.

Evaluation Metric

We propose a metric called *decreased un-service ratio* for performance evaluation. This choice can better characterize the improvement of the pricing algorithm over the original system (without monetary incentives) in the un-served events, compared with the service level (Singla et al. 2015).

Definition 1. (Decreased un-service ratio) The decreased un-service number of an algorithm $\mathcal{A}$ is the difference between the number of un-service events (UN) under Mobike and that of $\mathcal{A}$, i.e., $DUN(\mathcal{A}) = UN(\text{Mobike}) - UN(\mathcal{A})$. [2] The decreased un-service ratio of $\mathcal{A}$ is defined as $DUR(\mathcal{A}) = \frac{DUN(\mathcal{A})}{UN(\text{Mobike})} \times 100\%$.

Baselines

We compare HRP with the following baseline algorithms:

- Randomized pricing algorithm (Random): assigns monetary incentives randomly under the budget constraint.
- OPT-FIX (Goldberg, Hartline, and Wright 2001): an offline algorithm to maximize the acceptance rate with full a-priori user cost information.
- DBP-UCB (Singla et al. 2015): a state-of-the-art method for user-based rebalancing which applied a multi-armed bandit framework.
- DDPG (Lillicrap et al. 2015)
- HRA (Van Seijen et al. 2017): a reinforcement learning algorithm which directly employed reward decomposition.
- Offline-optimal: a pricing algorithm which maximizes the service level under the budget constraint with known user costs in advance, serving as an upper bound for any online pricing algorithms. To reduce the complexity of the formulation, we assume a trip finishes in the same timeslot (1 hour) it begins. [3] In our comparison with the *offline-optimal*, due to computational complexity, we assume that the costs of users walking from one region to another are the same for each timeslot t, defined as $c_k(i,j,x) = c_{ij}(t)$ if r_j is the neighboring region of r_i, where $c_{ij}(t)$ is drawn independently from the empirical distribution of user costs for picking up bikes in r_j instead of their current regions r_i defined as Eq. (1) from timeslot t of the dataset. Then, the problem can be formulated as the following integer linear program: [4]

$$\max \sum_{t\in\mathcal{T}} \sum_{i,j,l\in[1,n]} x_{ijl}(t) \tag{6}$$

$$\text{s.t.} \quad x_{ijl}(t) \in Z^+, \qquad \forall i,j,l,t \tag{7}$$

$$\sum_{j\in[1,n]} x_{ijl}(t) \le d_{il}(t), \qquad \forall i,l,t \tag{8}$$

$$\sum_{i,l\in[1,n]} x_{ijl}(t) \le S_j(t), \qquad \forall j,t \tag{9}$$

$$\sum_{t\in\mathcal{T}} \sum_{i,j,l\in[1,n]} x_{ijl}(t) c_{ij}(t) \le B \tag{10}$$

$$S_i(t+1) = S_i(t) - \sum_{j,l\in[1,n]} x_{jil}(t) + \sum_{l,j\in[1,n]} x_{lji}(t), \; \forall i,t. \tag{11}$$

Constraints (8) guarantee that for any timeslot, the number of users in r_i heading for r_l is no larger than the demand from r_i to r_l. Constraints (9) mean that the number of bikes picked up in r_j does not exceed the bike supply in this region. Constraint (10) ensures that the money the service provider spends is limited by B. Constraints (11) are for the dynamics of the supply of bikes.

Performance Comparison

We first compare HRP and reinforcement learning baselines to analyze the converging issue. Next, we compare HRP with varying budget and supply to evaluate its effectiveness in a day. Then, we analyze the long-term performance. Finally, we compare HRP with the *offline-optimal* scheme and evaluate its generalization ability.

Training Loss Figure 4(a) shows the training loss under HRP, HRA, and DDPG. As expected, DDPG diverges due to high-dimensional input spaces. Notably, HRA diverges with an even larger training loss. This is due to the impractical assumption of independence among sub-MDPs, which results in a biased estimator for the Q-value function. HRP outperforms the two and is able to converge. [5]

Effect of Varying Budget Constraints Figure 4(b) shows the decreased un-service ratio of each algorithm under different budget constraints. DDPG performs similarly at different budget levels as it fails to learn to efficiently use the budget in such a large input space. HRA performs better than DDPG due to the decomposition. HRP decreases the un-service ratio by $43\% - 63\%$, and outperforms other algorithms under all budget levels. The reason that OPT-FIX and DBP-UCB underperform HRP is that they do not consider spatial information, and focus solely on optimizing the acceptance rate.

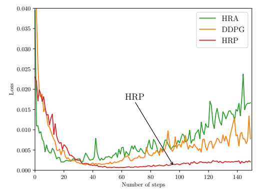
Achieving Better Bike Distribution To also analyze the rebalancing effect over a day, we measure how the distribution of bikes at the end of the day diverges from its value at the beginning of the day, using the Kullback-Leibler (KL)

[2]Note that UN(Mobike) is computed by simulating Mobike's original system.

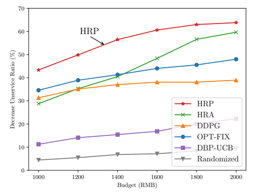
[3]Otherwise, one has to set the timeslot as 1 minute, resulting in an integer linear program with too large complexity. Please note that the model and simulation do not rely on the assumption.

[4]We solve the program with Gurobi.

[5]The running time of HRP until convergence is less than 6 minutes on an NVIDIA GTX Titan-X 12GB graphics card.

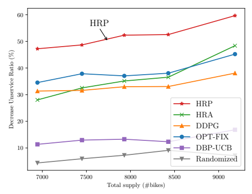


(a) Training loss.

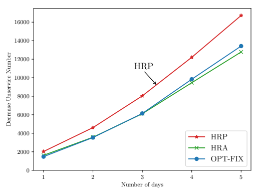


(b) Varying budget.

Figure 4: Comparison of training loss and varying budget.



(a) Varying supply.

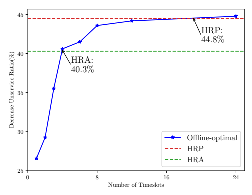


(b) Long-term.

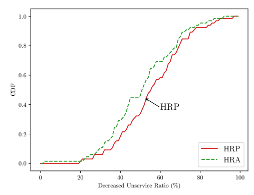
Figure 5: Comparison of varying supply and number of days.

HRP	HRA	DBP-UCB	DDPG	OPT-FIX
0.548	0.560	0.562	0.586	0.598

Table 1: KL divergence of user-based algorithms.



(a) Optimality.



(b) Generalization.

Figure 6: Optimality and generalization comparison.

divergence measure (Kullback and Leibler 1951). We simulate Mobike's original system (without monetary incentives), and obtain its KL divergence level to be 0.554. As seen in Table 1, OPT-FIX obtains a bike distribution that is most different from the initial bike distribution. The reason can be that OPT-FIX is too aggressive in maximizing the acceptance rate, which can worsen the bike distribution. HRP outperforms all existing algorithms with a KL divergence value of 0.548, even smaller than Mobike's value. This demonstrates that HRP is able to improve the bike distribution at the end of a day.

Effect of Varying Supply Besides varying the budget, it is also worth studying how HRP performs under different supply levels to evaluate its robustness. The results are shown in Figure 5(a). Intuitively, the problem is more challenging when supply is limited, which can lead to a large un-service rate. Random and DBP-UCB both perform poorly while the performance of OPT-FIX, DDPG, and HRA are almost the same. HRP performs significantly better than others, and achieves a $47\% - 60\%$ decrement in the un-service ratio, demonstrating its robustness against different total supply.

Long-Term Performance Apart from analyzing one day's effect, we also evaluate the long-term performance by varying the number of days. We compare HRP with two most competitive algorithms, HRA and OPT-FIX, from 1 day to 5 days using decreased un-service number. As shown in Figure 5(b), HRP outperforms other algorithms. The performance gap also gets larger as the number of days increases. This is because HRP can achieve a better bike distribution as discussed in the previous section, and it learns to maximize the long-term reward. Readers please refer to the supplemental material for our detailed analysis of the result.

Optimality We now provide comparison results with the *offline-optimal* scheme. To avoid computation and memory overhead, we conduct the comparison on a smaller area consisting of 3×3 regions with highest request density to evaluate how well HRP can perform in a most congesting area. We compare both HRP and HRA with the *offline-optimal*, by adjusting different values of timeslots V. The V-timeslot optimization means that for every V timeslots, we optimize the offline program with V horizons. Intuitively, a larger V leads to better performance. In this evaluation, the results are averaged over 400 independent runs of each algorithm. Figure 6a demonstrates that the performance of HRP is very close to that of 24-timeslot optimization, while HRA only performs close to 4-timeslot optimization.

Generalization Now, we investigate whether HRP can transfer well, i.e., trained with certain areas but still performs well when applied to new ones. As the dataset only consists of trajectories in Shanghai, we divide the whole area into smaller areas where each consists of 3×3 regions to generate more areas. We train HRP and HRA in a certain area and then test them on other areas. Figure 6b shows the cumulative density function (CDF) of HRP and HRA on the decreased un-service ratio over all areas. HRP can achieve a $40\% - 80\%$ un-service ratio decrement over 80% areas in testing. This demonstrates that HRP generalizes well to new areas that it has never seen before. Note that the curve of HRP is strictly to the right of HRA, showing that HRP generalizes better than HRA.

Conclusion

We propose a deep reinforcement learning framework for incentivizing users to rebalance dockless bike sharing systems. We propose a novel deep reinforcement learning algorithm called *hierarchical reinforcement pricing* (HRP). HRP maintains a divide-and-conquer structure that can handle complex temporal dependencies. It is also embedded with a localized module to better estimate the Q-value, which can capture spatial dependencies. We conduct extensive experiments for HRP based on real dataset from Mobike, a major

Chinese dockless bike sharing company. Results show that HRP outperforms state-of-the-art methods.

As for future work, one interesting extension is to deploy our algorithm in a real-world bike sharing system. To adapt to shifting environments across the year, one can first update the simulator with latest collected data, and then train HRP with the updated simulator offline every a fixed number of days. After training, one can adopt the new policy online. It is also promising to consider more factors in the user feedback model, e.g. travel distance and the time of day.

References

Andre, D., and Russell, S. J. 2002. State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, 119–125.

Cai, Q.; Filos-Ratsikas, A.; Tang, P.; and Zhang, Y. 2018. Reinforcement mechanism design for fraudulent behaviour in e-commerce. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Chemla, D.; Meunier, F.; Pradeau, T.; Calvo, R. W.; and Yahiaoui, H. 2013. Self-service bike sharing systems: simulation, repositioning, pricing.

Chen, L.; Zhang, D.; Wang, L.; Yang, D.; Ma, X.; Li, S.; Wu, Z.; Pan, G.; Nguyen, T.-M.-T.; and Jakubowicz, J. 2016. Dynamic cluster-based over-demand prediction in bike sharing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 841–852. ACM.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Dayan, P., and Hinton, G. E. 1993. Feudal reinforcement learning. In *Advances in neural information processing systems*, 271–278.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research* 13:227–303.

Fricker, C., and Gast, N. 2016. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *Euro journal on transportation and logistics* 5(3):261–291.

Ghosh, S., and Varakantham, P. 2017. Incentivizing the use of bike trailers for dynamic repositioning in bike sharing systems.

Ghosh, S.; Trick, M.; and Varakantham, P. 2016. Robust repositioning to counter unpredictable demand in bike sharing systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 3096–3102. AAAI Press.

Goldberg, A. V.; Hartline, J. D.; and Wright, A. 2001. Competitive auctions and digital goods. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, 735–744. Society for Industrial and Applied Mathematics.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics* 22(1):79–86.

Li, Y.; Zheng, Y.; Zhang, H.; and Chen, L. 2015. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 33. ACM.

Li, Y.; Zheng, Y.; and Yang, Q. 2018. Dynamic bike reposition: A spatio-temporal reinforcement learning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1724–1733. ACM.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, J.-R., and Yang, T.-H. 2011. Strategic design of public bicycle sharing systems with service level constraints. *Transportation research part E: logistics and transportation review* 47(2):284–294.

Liu, J.; Sun, L.; Chen, W.; and Xiong, H. 2016. 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*, 1005–1014. ACM.

Liu, J.; Sun, L.; Li, Q.; Ming, J.; Liu, Y.; and Xiong, H. 2017. Functional zone based hierarchical demand prediction for bike system expansion. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 957–966. ACM.

O'Mahony, E., and Shmoys, D. B. 2015. Data analysis and optimization for (citi) bike sharing. In *AAAI*, 687–694.

Shaheen, S.; Guzman, S.; and Zhang, H. 2010. Bikesharing in europe, the americas, and asia: past, present, and future. *Transportation Research Record: Journal of the Transportation Research Board* (2143):159–167.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 387–395.

Singla, A.; Santoni, M.; Bartók, G.; Mukerji, P.; Meenen, M.; and Krause, A. 2015. Incentivizing users for balancing bike sharing systems. In *AAAI*, 723–729.

Tang, P. 2017. Reinforcement mechanism design. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 5146–5150.

Van Seijen, H.; Fatemi, M.; Romoff, J.; Laroche, R.; Barnes, T.; and Tsang, J. 2017. Hybrid reward architecture for reinforcement learning. In *Advances in Neural Information Processing Systems*, 5392–5402.

Xue, Y.; Davies, I.; Fink, D.; Wood, C.; and Gomes, C. P. 2016. Avicaching: A two stage game for bias reduction in citizen science. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 776–785. International Foundation for Autonomous Agents and Multiagent Systems.

Yang, Z.; Hu, J.; Shu, Y.; Cheng, P.; Chen, J.; and Moscibroda, T. 2016. Mobility modeling and prediction in bike-sharing systems. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 165–178. ACM.