# A Deep Reinforcement Learning Model for Large-Scale Dynamic Bike Share Rebalancing with Spatial-Temporal Context

Zhuoli Yin
School of Industrial Engineering
Purdue University
West Lafayette, IN, USA
zhuoliyin@purdue.edu

Zhaoyu Kou
School of Industrial Engineering
Purdue University
West Lafayette, IN, USA
kouz@purdue.edu

Hua Cai†
School of Industrial Engineering,
Environmental and Ecological Engineering
Purdue University
West Lafayette, IN, USA
huacai@purdue.edu

## ABSTRACT

In cities with top ridership, bike-sharing systems (BSSs) have expanded to over 500 stations, necessitating dynamic rebalancing of bikes among stations to accommodate timely imbalanced customer demands. BSS operators redistribute bikes among stations by employing a fleet of rebalancing vehicles. Traditional mixed-integer programming and heuristic approaches often generate offline or shortsighted solutions. Moreover, current approaches based on reinforcement learning and deep learning, while beneficial for urban mobility operations, are designed primarily for small-scale BSSs or partitioned sub-BSSs, deploying only small rebalancing fleets. How to produce online rebalancing solutions for large-scale BSS with multiple rebalancing vehicles is significant for current BSS operations yet remains still unclear. To fill the gap, we proposed a deep reinforcement learning based model to learn the optimal policy for dynamic bike share rebalancing. We designed a deep Q-network (DQN) with inputs reflecting real-time spatial-temporal system observations and outputs corresponding to intertwined rebalancing actions (repositioning and routing) for individual vehicles. Each rebalancing vehicle operates asynchronously, independently solving the DQN. Using Divvy Bike's historical data from Chicago — a system including over 500 stations and 16 rebalancing vehicles in the fleet— the experiments demonstrate that our model generates effective rebalancing solutions on a large-scale BSS against baselines, facilitating the operation of shared mobility systems in large cities.

## CCS CONCEPTS

• **Applied computing** → *Transportation.*

## KEYWORDS

bike share, dynamic rebalancing, spatial-temporal, deep reinforcement learning

## 1 INTRODUCTION

Bike-sharing systems (BSSs) offer users accessibility, convenience, and low-cost mobility, promoting multimodal trips and short-distance travel as alternatives to private cars, thus aiding in congestion mitigation and greenhouse gas reduction [9,26]. In response to growing demand, BSS companies are extending existing networks and launching new systems in global cities [9,21,26]. The distribution of bike share trips across the service region is often imbalanced due to spatial-temporal demand differences, which could precipitate customer dissatisfaction and potential revenue losses. These imbalances may also lead to increased greenhouse gas emissions if customers turn to more energy-intensive modes of transportation [11]. Therefore, regularly rebalancing bikes among stations is imperative to maintain a reasonable distribution across the service region [8]. With increasing BSS scale, dynamic rebalancing becomes a key operational solution [29,31]. Such a strategy is typically modeled as a Dynamic Bike Share Rebalancing Problem (*DBSRP*). *DBSRP* aims to dispatch rebalancing vehicles to certain bike stations and then pick-up or drop-off a certain number of bikes in those target stations to resolve the real-time varying demands [28]. Specifically, the process includes routing decisions, which determine the sequence of station visits, and repositioning decisions, indicating the number of bikes to be moved at each station. The rebalancing is usually conducted by a fleet of automobiles (referred to as rebalancing vehicles hereafter). Rebalancing vehicles carry a particular quantity of bikes and travel among bike stations sequentially. It should be noted that routing and repositioning decisions are entangled, given that rebalancing vehicles can only move bikes at stations they physically visit.

Existing works primarily addressed *DBSRP* through mixed-integer programming (MIP) or heuristic approaches. Given *DBSRP*'s complexity as an NP-hard problem, traditional MIP-based methods utilized decomposition or heuristic techniques for feasible solutions [5,10,12,33,36]. Yet, these solutions primarily cater to offline scenarios, given their need for full data for the entire planning horizon. Several studies also proposed heuristic or rolling horizon-based algorithms to support rebalancing decisions, offering online solutions for BSSs [4,28]. But these methods are typically shortsighted, only approximating the selection of rebalancing actions by foreseeing a few steps within the whole planning horizon [37]. Meanwhile, these algorithms are often developed and tested for small-scale BSSs, or they segment the research area into clusters, each serviced by a single rebalancing vehicle [5,14,25,27,33]. For instance, [15] only focused on the principal region of the entire BSS in the New York City and assigned a single rebalancing vehicle for each clustered sub-

region. However, as BSSs continue growing in scale, the computational complexity of *DBSRP* models increases correspondingly, and these algorithms fail to capture the unbalanced rebalancing needs globally and are not equipped to handle large-scale operations. Consequently, they are now insufficient for generating online rebalancing solutions with high-quality and far-sight.

Recent advancements in Deep Reinforcement Learning (DRL) methods have demonstrated the potential to solve sequential decision-making problems. However, existing DRL algorithms, as applied in urban mobility operations, exhibit limitations that hinder their direct application in large-scale *DBSRP*. DRL-based frameworks from [1,22], which were designed for ride-sharing systems, only allow isolated decisions and overlook the interdependencies between multiple concurrent decisions. Such interdependencies are necessary for *DBSRP* which has both repositioning and routing. [15] utilized a spatial-temporal reinforcement learning approach to *DBSRP*, but their model only focused on the principal system of Citi Bike in NYC and split it into subclusters. They only assigned a single vehicle to each cluster, failing to account for heterogeneous customer demands in a large-scale city. Meanwhile, [3] proposed a hybrid approach integrating DRL with MIP optimizer for freight delivery problem, but it is confined to static decision-making and unsuitable for online problems.

In this work, we proposed a DRL-based model to generate the optimal strategy for dynamic rebalancing of large-scale BSS in real-time. Our agent interacts iteratively with a BSS simulator, utilizing a Deep Q-Network (DQN) to estimate the long-term Q-values of rebalancing actions, whose objective is to maximize the system's overall profits. The agent selects the rebalancing action with the highest Q-value for each rebalancing vehicle. Different from the existing work, our model incorporated large-scale, real-time, and farsighted rebalancing decision-making. The main contributions of this study can be summarized as follows:

(1) We developed a DRL-based framework for *DBSRP*, capable of real-time optimization for large-scale *DBSRP* to enhance overall profits without necessitating service region partitioning.

(2) Contrasting with existing RL applications in urban mobility system operations, we constructed a convolutional neural network for DQN whose outputs are tailored to interdependent rebalancing actions.

(3) We evaluated the model using real-world data from Chicago's Divvy, featuring over 500 stations and 16 rebalancing vehicles. The experiments demonstrated its effectiveness in improving overall system profits.

## 2 PROBLEM STATEMENT

In this work, we aim to find the optimal solution for *DBSRP*. The notations used throughout this work are defined in A.1. *DBSRP* is a variant of the dynamic vehicle routing problem with split deliveries [2,24]. This involves multiple rebalancing vehicles that are routed to serve real-time customer demands originating from bike stations. These bike stations are not restricted to be visited only once during the operation and they can be re-visited by rebalancing vehicles as needed. Adopting from [12,17], this problem can be formulated as follows:

$$\min_{z,y} \sum_{i\in N,t\in T} \alpha \cdot L_{i,t} + \sum_{i,j\in N,m\in M,t\in T} \beta \cdot P_{i,j} \cdot z_{i,j,m,t} \quad (1)$$

s.t.

$$S_{j,t+1}^{Bike} = min\left\{max\left\{0, S_{j,t}^{Bike} + \sum_{m\in M} y_{j,m,t} + D_{j,t}\right\}, C_j^{station}\right\} \quad \forall j \in N, t \in T \quad (2)$$

$$0 \le S_{j,t}^{Bike} + \sum_{m\in M} y_{j,m,t} \le C_i^{station} \quad \forall j \in N, t \in T \quad (3)$$

$$L_{j,t} = \left| S_{j,t+1}^{Bike} - (S_{j,t}^{Bike} + \sum_{m\in M} y_{j,m,t} + D_{j,t}) \right| \quad \forall j \in N, t \in T \quad (4)$$

$$V_{m,t+1}^{Bike} = min\left\{max\left\{0, V_{m,t}^{Bike} + \sum_{m\in M} -y_{j,m,t}\right\}, C_m^{vehicle}\right\} \quad \forall m \in M, t \in T \quad (5)$$

$$\sum_{j\in N} z_{i,j,m,t} - \sum_{j\in N} z_{j,i,m,t-1} = 0 \quad \forall i \in N, m \in M, t \in T \quad (6)$$

$$\sum_{j\in N, m\in M} z_{i,j,m,t} \le 1 \quad \forall i \in N, t \in T \quad (7)$$

$$\sum_{i,j\in N} z_{i,j,m,t} \le 1 \quad \forall m \in M, t \in T \quad (8)$$

$$-C_m^{vehicle} \cdot \sum_{i\in N} z_{i,j,m,t} \le y_{j,m,t} \le C_m^{vehicle} \cdot \sum_{i\in N} z_{i,j,m,t} \quad \forall j \in N, m \in M, t \in T \quad (9)$$

$$S_{i,0}^{station} = \frac{1}{2} C_i^{station} \ \forall i \in N, \quad V_{m,0}^{vehicle} = \frac{1}{2} C_m^{vehicle} \ \forall m \in M \quad (10)$$

$$-C_m^{vehicle} \le y_{i,m,t} \le C_m^{vehicle} \ \forall i \in N, m \in M, t \in T \quad (11)$$

$$0 \le S_{i,t}^{Bike} \le C_i^{station} \ \forall i \in N, m \in M, t \in T \quad (12)$$

$$0 \le V_{m,t}^{Bike} \le C_m^{vehicle} \quad \forall i \in N, m \in M, t \in T \quad (13)$$

$$z_{i,j,m,t} \in \{0,1\} \quad (14)$$

The objective function (1) minimizes the total costs of losing customers and fuel costs incurred by rebalancing vehicles that route to targe stations, in which $\alpha$ represents the average cost paid by customers for a single bike share trip and $\beta$ represents the average fuel cost per mile of vehicles. For constraint (2) and (3), the in- and out-flows of bikes in the station are conserved. Also, the bike quantity in the station either after rebalancing or after the rent/return of customers is within station capacity. Constraint (4) records the customer loss in each station, which is the difference between the expected inventory level of the station and the actual inventory level. Constraint (5) ensures that the number of bikes in and out of the vehicles is conserved. Additionally, the number of bikes in the rebalancing vehicle after visiting the station is within the vehicle capacity. For constraint (6), it enforces that rebalancing vehicles arriving at and departing from the target stations are preserved. The rebalancing vehicles that depart from a station $\sum_{j\in N} z_{i,j,m}^t$ could only be the ones that arrive at this station in the last timeslot $\sum_{j\in N} z_{j,i,m}^{t-1}$. Note that staying in a station could be viewed as moving out of the station and moving into it again in the same time slot. $\sum z_{N_0,i,m}^0$ and $\sum z_{i,N_0,m}^T$ are set to be 1 since every rebalancing vehicle must initially depart from the depot and finally return to it. Constraints (7) and (8) guarantee the requirement that, in any timeslot, at most one vehicle is visiting the same station and one vehicle can be present at most one station. These constraints avoid the risk of multiple vehicles rebalancing the same station, thereby preventing offsetting effects. Constraint (9) couples the repositioning decision and routing decision, enforcing that rebalancing vehicles can only pick-up or

drop-off bikes in the visited stations. Constraint (10) sets the initial capacity of bike stations and rebalancing vehicles to be half full. Constraints (11)-(13) enforce that the number of repositioned bikes does not violate the physical capacities. Additionally, the number of bikes parked in stations and carried by rebalancing vehicles should not violate their maximum capacities. Constraint (14) defines the routing decision $z^t_{s,s',v}$ as binary variables.

# 3 DATA AND METHODS

This section details our methods designed for the *DBSRP*. Section 3.1 introduces the Divvy BSS data from Chicago, employed for training and testing *DBSRP* algorithms. In Section 3.2, we outline the framework developed based on the RL paradigm, involving the environment, agent (RL algorithm and DQN), state, action, and reward. Section 3.3 introduces baseline models for comparative analysis, while Section 3.4 defines evaluation metrics for measuring *DBSRP* algorithm performance.

## 3.1 Case Study Data and Preprocessing

This study utilized the publicly accessible Divvy dataset, which encompasses the station-based BSS data in Chicago. Each trip record incorporates the origin and destination station ID along with the start and end time of each trip. The station information of Divvy was collected through GBFS (General Bikeshare Feed Specification) API. The station information includes the station id, station name, location (latitude & longitude), and capacity. The service region containing 578 active stations in Chicago was divided into 51×51 grids. Each grid in this study has a size of 400m×700m [22]. 578 active Divvy stations were thereby assembled into 451 abstract stations. According to Divvy's system operator, the rebalancing vehicle fleet of Divvy has 16 automobiles in service and the bike depot of Divvy was built in (41.890082, -87.658458), which corresponds to the grid (27, 26) in this study.

## 3.2 Deep Reinforcement Learning Framework

Here, we present the formulation of the model-free deep reinforcement learning framework which aims to learn the optimal dynamic rebalancing policy.

*3.2.1 Simulator.* A simulator serves as the environment, with which the agent interacts. The dynamics of a simulator include initialization, bike rent/return process, and rebalancing activities. The simulation process is detailed in Figure 1. Initially, stations and rebalancing vehicles are assumed to be half full and rebalancing vehicles depart from the depot at the beginning of the episode. Then, in each timeslot, the simulator executes the rebalancing actions generated by the agent to modify the bike inventory of visited stations. Following that, customers are emulated to rent or return bikes according to historical demands.

*3.2.2 State space.* In this study, the state variable $s_t$ reflects the real-time BSS environment observation at the beginning of timeslot $t$. We formulated the state variables $s_t = (S^{Bike}_{i,t}, S^{Dock}_{i,t}, V^{Bike}_{m,t}, V^{Slot}_{m,t}, V^{Loc}_{m,t}), i \in \{1,2,\dots,N\}, m \in \{1,2,\dots,M\}$, which

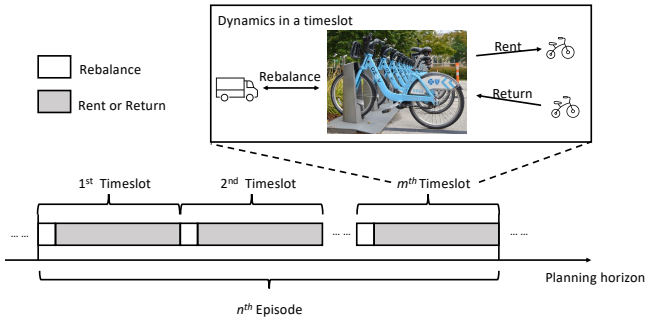


**Figure 1: BSS simulator. It executes bike rent/return and rebalancing sequentially.**

corresponds to the spatial-temporal information of bike stations, rebalancing vehicles, respectively.

*3.2.3 Action space.* Corresponding to the action variables defined in Section 2, the rebalancing action $a_{m,t} = (z_{i,j,m,t}, y_{j,m,t})$ is two-dimensional for rebalancing vehicle $m$ in the timeslot $t$. To reduce the computation time, the action space for routing action $z_{i,j,m,t}$ consists of grids reachable via a maximum of 10 horizontal or vertical moves from the current grid (each move is a grid). Meanwhile, the action space for repositioning action $y_{j,m,t}$ is comprised of seven discrete actions: do not move any bike in the target bike station, moving five or ten bikes into/out of the target station, moving maximum feasible bikes out of the station ("greedy all out"), and moving maximum feasible bikes from the rebalancing vehicle into the station ("greedy all in"). Such an action space is tailored for large-scale BSS, because rebalancing a large number of bikes in bike stations that have high customer demands is always a necessity during the day; also, customer demands are at a low level in late night or early morning so that few bike station needs rebalancing.

*3.2.4 Reward function.* The agent aims to take rebalancing actions that maximize the expected total future rewards until the episode terminates. We designed the reward function for each rebalancing vehicle $m$ that executes rebalancing action $a_{m,t}$ in the state $s_t$, by modifying the objective function (1) described in Section 2:

$$r(a_{m,t}, s_t) = \alpha \cdot \Delta L_{j,t} - \beta \cdot P_{i,j} \quad (15)$$

where $\Delta L_{j,t}$ is consequential customer loss reduction in the visited station $j$ in timeslot $t$. The customer loss reduction is the customer loss generated in the stations *without* any rebalancing during the simulation *minus* the customer loss generated in the station *with* the executed rebalancing under a given policy.

*3.2.5 Double Deep Q-Network (DDQN) algorithm.* To train the DQN network that learns the policy, we used the DDQN algorithm with experience replay as proposed in [32]. DDQN algorithm was applied in this work because it has been proven to achieve much better performance in large-scale sequential decision-making problems and can help reduce the overestimation of DQN.

To select a rebalancing action to be executed, the agent estimates the Q-value of each rebalancing action $Q_\pi(s_t, a_t)$ for individual

rebalancing vehicles given the environment state $s_t$ in timeslot $t$. Under the policy $\pi$, the Q-value of the action $a_t$ is the cumulative future rewards when choosing that action in a certain state and following the optimal policy thereafter:

$$Q_\pi(s_t, a_t) \equiv \mathbb{E}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_k \mid S = s_t, A = a_t, \pi\right] \quad (16)$$

where $\gamma \in [0,1]$ is a discount factor that compensates the importance of immediate and future rewards. The optimal Q-value of an action $a_t$ in a state $s_t$ is therefore $Q^*(s_t, a_t) \equiv \max_\pi Q_\pi(s_t, a_t)$. On top of that, an optimal rebalancing policy can be derived from the optimal Q-values by choosing the highest-valued action $a_t^* = \underset{a_t}{\operatorname{argmax}} Q^*(s_t, a_t)$ in each state $s_t$ [32]. Such long-term optimal Q-value function is iteratively updated by using the *Bellman equation* [18]:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t\right] \quad (17)$$

In practice, a neural network with weights $\theta$, which is a non-linear function approximator, is usually utilized to estimate the Q-value $Q(s, a; \theta) \approx Q^*(s, a)$ [19,30]. We refer to such a neural network as a Deep Q-Network (DQN), whose architecture will be introduced in Section 3.2.6. In this paper, the inputs of the DQN are the instantaneous states of the BSS environment at the beginning of timeslot $t$, while the outputs are Q-values of all rebalancing actions in the defined action space for an individual rebalancing vehicle.

To improve the policy $\pi$, the weights $\theta_i$ of a DQN is updated by minimizing the mean-squared error (MSE) loss functions $L_i(\theta_i)$ occurred at each iteration $i$:

$$L_i(\theta_i) = \mathbb{E}_{s_t, a_t}\left[\left(y_i - Q(s_t, a_t; \theta_i)\right)^2\right] \quad (18)$$

where $y_i = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) | s_t, a_t\right]$ is the target value for the DQN in iteration $i$ [20]. The DQN with parameters $\theta_{i-1}$ used to estimate the Q-values are from the previous iteration $i-1$. In this work, based on the DDQN algorithm, we employed two homogenous DQNs to approximate the Q-values and denoted their weights as $\theta$ and $\theta^-$, respectively. The target $y_i$ is therefore modified as

$$y_i^{DDQN} = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma Q(s_{t+1}, \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}); \theta_{i-1}^-) | s_t, a_t\right] \quad (18)$$

which reduces the overestimations by decomposing the max operator in the target $y_i$ into action selection

$$a_{t+1}^* = \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) \quad (20)$$

and action evaluation $Q(s_{t+1}, a_{t+1}^*; \theta_{i-1}^-)$ [32]. The parameters $\theta^-$ are periodically copied from parameters $\theta$. The agent always chooses the action that maximizes the Q-value based on the outputs of the DQN with parameters $\theta$.

*3.2.5 DQN architecture.* We designed an architecture for the DQN, shown in Figure 2, to estimate the Q-values of actions while capable of capturing spatial-temporal information of the large-scale BSS states. Adopted from [23,30,35], we designed two input channels for the DQN, representing two categories of observations. These channels gather both global and local observations from BSS states, taking into account their spatial-temporal characteristics. Specifically, the global observation is viewed from the entire BSS system's aspect, while the local observation is viewed from each rebalancing vehicle's aspect centering the rebalancing vehicle in the plane. Meanwhile, the output was structured as a table in order to approximate the Q-values of two entangled rebalancing actions.

The description of feature planes is summarized in Table 1. Global observations (Table 1(A)) have 5 stacked feature planes with size $51 \times 51$ and each cell in the plane corresponds to a feature of a grid in the entire service region; local observations (Table 1(B)) have 13 stacked feature planes with size $21 \times 21$ and each cell corresponds to a feature of a grid within action space. Two threads of inputs were passed through the convolutional layers and fully connected layers to nonlinearly estimate the Q-values. Specifically, first, the global observation inputs were applied average pooling with the pool size of 2, to down-sampling feature planes, which reduces the complexity of the neural networks [13]. Then the first and second hidden layers convolved the filters of size $3 \times 3$ with the dimensionality of 64 and 128, respectively. Both layers were applied exponential linear unit (ELU) activation. Likewise, the local observation inputs entered the convolutional layers, which have 64 filters of size $1 \times 1$ and ELU activation. Convolved global and local observation inputs were stacked and then entered two convolutional layers with 64 and 1 filters of size $1 \times 1$. The final output layer is composed of 3087 units, configured into a two-dimensional Q-value table of size $441 \times 7$, where each dimension corresponds to the action space size. In this way, in the output table, choosing one Q-value indicates the repositioning and routing correspondingly.

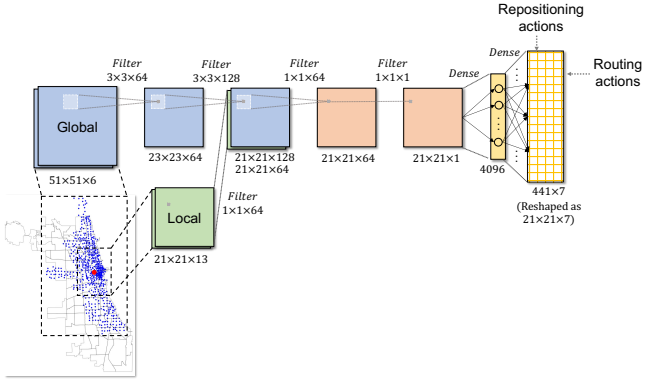


**Figure 2: DQN architecture. The red dot represents a rebalancing vehicle.**

In our study, we utilized an *action mask* on the output Q-value table to eliminate infeasible actions. This *a priori* constraint excludes specific Q-values from selection, preventing infeasible actions from entering the experience pool. Specifically, if either the available bikes or rebalancing vehicle slots are insufficient for all repositioning actions, the corresponding Q-value table pixel is masked as 0. Similarly, if a grid lacks an installed station or if a station is already assigned to another rebalancing vehicle in the current timeslot, its corresponding pixel is also masked as 0.

**Table 1: Representations of the system in the input layer of the neural network.**

(A) The global observation inputs with plane size $51 \times 51$ depict the spatial observation from global system perspective.

| Input | Feature | Number of planes | Description |
|---|---|---|---|
| Global | Bikes in rebalancing vehicles | 1 | Number of bikes carried by a rebalancing vehicle in this grid; otherwise filled with 0 |
| | Slot in rebalancing vehicles | 1 | Number of slots in a rebalancing vehicle in this grid; otherwise filled with 0 |
| | Future rebalancing vehicles | 1 | Number of available rebalancing vehicles assigned to visit in this grid in the next time slot |
| | Bikes in stations | 1 | Number of bikes parked in the station in this grid; otherwise filled with 0 |
| | Docks in stations | 1 | Number of docks in the station in this grid; otherwise filled with 0 |

(B) The local observation inputs with plane size $21 \times 21$ depict the spatial and temporal observation from rebalancing vehicle perspective.

| Input | Feature | Number of planes | Description |
|---|---|---|---|
| Local | Day of month | 2 | Constant planes filled with sine/cosine of day of the month |
| | Day of week | 2 | Constant planes filled with sine/cosine of day of the week |
| | Minute of day | 2 | Constant planes filled with sine/cosine of minute of the day |
| | Position | 1 | The center is filled with 1 to indicate the position of this rebalancing vehicle, otherwise filled with 0 |
| | Global coordinates | 2 | Constant planes filled with normalized horizontal/vertical coordinates of this rebalancing vehicle from the global aspect |
| | Local coordinates | 2 | Constant planes filled with normalized horizontal/vertical coordinates of this rebalancing vehicle from the local aspect |
| | Distance | 1 | Distance from other grids to the center |
| | Legal map | 1 | Filled with 1 if this grid has installed bike station and has not been assigned to visit by other rebalancing vehicles, otherwise filled with 0 |

## 3.3 Baselines

To evaluate the performance of the proposed model, we compared it against four *DBSRP* baseline algorithms.

(1) **No Rebalancing**. The operator does not execute any rebalancing in BSS.

(2) **Greedy Rebalancing**. As outlined in [7,15,34], in this algorithm, for a near-full rebalancing vehicle, it always travels to the nearest near-empty stations and offloads the most possible bikes there. For a near-empty rebalancing vehicle, it always travels to the nearest near-full stations and collects as many bikes as possible.

(3) **Lagrangian Dual Decomposition-based MIP (LDD-MIP)**. As proposed by [12], they exploited LDD and abstraction mechanisms to solve the MIP formulated in Section 2.

(4) **Greedy Online Anticipatory Heuristic (GOAH)**. Developed by [16], they provided a heuristic approach to produce online solutions for *DBSRP* by looking ahead three timeslots and computing policy for each individual rebalancing vehicle.

## 3.4 Evaluation Metrics

We used the following three metrics to measure the performance of different rebalancing methods:

(1) Customer loss. It is the number of customers who failed to rent bikes at their initial departure stations or return bikes at the original arrival stations. In addition to that, we also defined customer loss reduction, which is the difference between customer loss with **No Rebalancing** and customer loss with running another rebalancing strategy.

(2) Vehicle routing distance (VRD). It is the sum of the distance that a rebalancing vehicle travels among the visited stations in an episode. Vehicle routing distance is also known as vehicle-miles-traveled (VMT).

(3) Improved profit. It is the additional ride profit earned from customer loss reduction subtracts the routing cost of the rebalancing vehicles. In this work, the profit of each ride is assumed to be the cost of each single trip ($3.3 [6]). Additionally, given that the internal combustion engine rebalancing vehicle consumes fuel when routing to target stations, the routing cost of the rebalancing vehicles in this work is estimated as the mileage rate (58cents/mile in 2019 [38]) multiplies the routing distance of rebalancing vehicles.

# 4 EXPERIMENTS AND RESULTS

## 4.1 Experiment setup

The DRL model was trained on the Gilbreth community cluster with 256 GB of RAM and two P100 Nvidia Tesla GPUs at Purdue University. Our experiment used historical data of a four-week period from September 2, 2019 (Monday) to September 29, 2019 (Sunday). The first three weeks' data serve as the basis for model

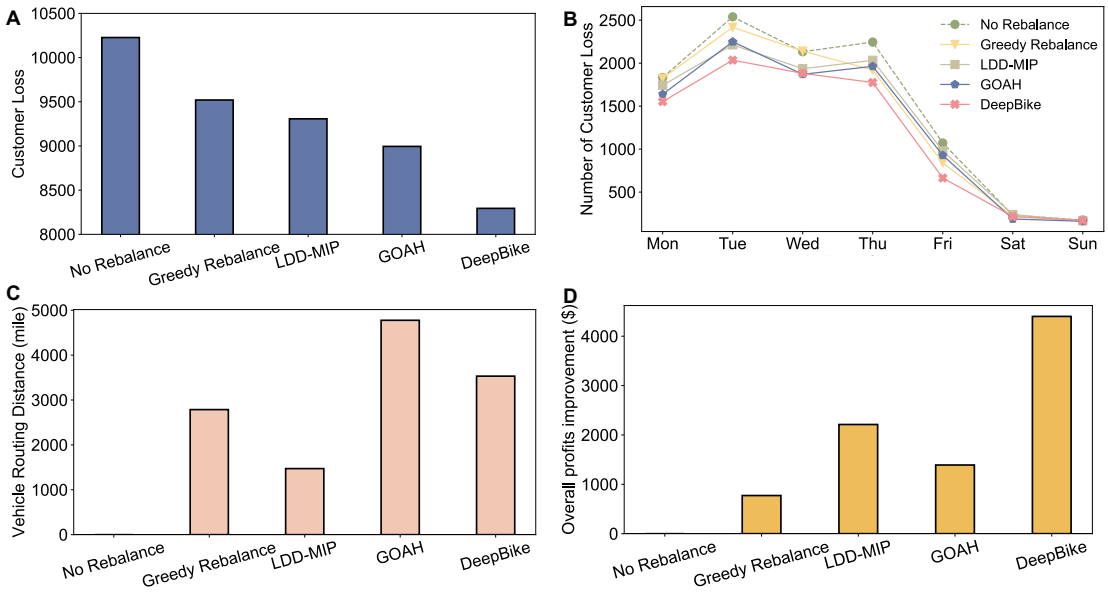


**Figure 3: Comparison of different rebalancing strategies in terms of different metrics. (A) Comparison of average weekly customer loss; (B) Comparison of average daily customer loss; (C) Comparison of average weekly vehicle routing distance; (D) Comparison of average profits improvement.**

training, while the final week's data is allocated for testing. The selected dataset incorporates over 449,000 trip records within Chicago. We trained the model for 6,500 episodes (156,000 training steps in total) with a physical training time of 168 hours until the model converged. For the DDQN algorithm, the learning rate was set as 0.0001. The initial 40,000 steps were set for exploration, over which the exploration rate decreases from 1 to 0.05. The replay buffer keeps the most recent 40,000 steps as experiences and we set the time discount rate $\gamma$ to be 0.9. To alleviate the uncertainty of rebalancing models, we ran the experiments ten times and evaluated their average performance.

## 4.2 Empirical results

Figure 3(A) demonstrates that our model minimizes total customer loss over a one-week test horizon, outperforming all other strategies. Compared to **No Rebalance**, our model leads to an average reduction of 1,942 customers loss in a week. In contrast, **GOAH**, **LDD-MIP** and **Greedy Rebalance** result in average reductions of 1,232, 920, and 707 customer loss, respectively. Consequently, the model in this work surpasses **GOAH**, **LDD-MIP**, and **Greedy Rebalance** by lowering additional customer loss by 57.6%, 111.09%, and 174.68% respectively. The daily customer loss reduction, as shown in Figure 3(B), is most significantly reduced by our model during weekdays among all evaluated strategies. Both our model and **GOAH** outperformed **LDD-MIP** in reducing customer loss due to their adaptability to BSS's real-time customer demands, whereas **LDD-MIP** produces only offline rebalancing solutions. **Greedy Rebalance** underperforms as it neglects upcoming demands and lacks coordination between rebalancing vehicles. Notably, there is no significant difference in customer loss reduction among rebalancing strategies on weekends. This is because the absolute customer demands on weekends are much smaller than those on weekdays. Most of bike stations can be thereby self-rebalanced without the need of rebalancing.

The VRDs in Figure 3(C) indicate that our model is more active in traveling to bike stations to rebalance bikes, in comparison to other baselines. It can be seen that the rebalancing under **LDD-MIP** has the shortest routing distances over other strategies. In contrast, our model aggressively dispatches the rebalancing vehicles, whose VRD is 137.57% and 25.45% higher than that of **LDD-MIP** and **Greedy Rebalancing**, respectively.

In Figure 3(D), the proposed DRL model improves the highest overall profits among rebalancing strategies, which is 101.26% more than **LDD-MIP** model, 220.01% more than **GOAH** model and 476.13% more than **Greedy Rebalancing**.

The overall profit improvement is the tradeoff between the gain from reducing customer loss and the cost incurred from rebalancing vehicle routing. Even though executing an aggressive rebalancing strategy will lead to a higher routing distance, the rebalancing solutions generated by **DeepBike** in this work can significantly contribute to reducing potential customer loss and in turn reaching the highest profits.

# 5 CONCLUSION

In this study, we proposed a DRL-based rebalancing model for large-scale BSSs, capable of producing timely rebalancing solutions. We designed a DQN to capture the real-time large-scale spatial-temporal system states without partitioning the service region and to estimate the Q-values of the rebalancing actions of each single rebalancing vehicle. Our proposed model learns to

update its DQN by interacting with the BSS environment iteratively through trial and error. The numerical results presented above demonstrate that, in a large-scale BSS which has more than 500 bike stations and 16 rebalancing vehicles, our model was able to reduce the most customer loss and achieve the highest overall profit improvement in a one-week testing horizon. The customer loss reduction and overall profit improvement are beneficiaries for the customer and operator. The model in this work learns to trade off the routing fuel cost to increase the overall profits and customer loss reduction.

The scheme of DRL facilitates large-scale and online solutions, distinct from existing algorithms. The designed CNN-based neural network captures the real-time large-scale dynamics of the entire BSS as input and estimates the long-term benefits of rebalancing decisions. Additionally, in contrast to MIP models, converged neural network is able to generate online rebalancing decisions without the need of solving the model from scratch. It is also notable that, even though our model was experimented on a station-based BSS because of the data availability, our model is transferable to free-floating BSS because zoning the service region into $n \times n$ grids and aggregating free-floating bikes parked within neighbor grids as abstract stations are still applicable to align with the inputs of DQN.

We suggest several opportunities for future research. The publicly available trip records published by BSS operators are only observed demands, which underestimates the bike check-in rate in stations. A true demand estimation model would benefit the demand simulation which will in turn improve the decision-making policy. Additionally, developing a multi-agent reinforcement learning algorithm that takes the collaboration of rebalancing vehicles into account for *DBSRP* could better learn the optimal rebalancing policy.

# APPENDIX

## A.1 List of notations

| | |
|---|---|
| $\pi$ | The rebalancing policy |
| $T$ | The total number of time slots in one episode |
| $\Delta t$ | The length of each time slot |
| $t$ | Current time index |
| $M$ | The number of rebalancing vehicles in the fleet, $m \in \{1,2,\dots,M\}$ |
| $N$ | The number of aggregated bike stations in BSS, $i, j \in \{1,2,\dots,N\}$ |
| $s$ | BSS environment state |
| $S_{i,t}^{Bike}$ | The number of bikes in the station $i$ at the beginning of time slot $t$ |
| $S_{i,t}^{Dock}$ | The number of docks in the station $i$ at the beginning of time slot $t$ |
| $V_{m,t}^{Bike}$ | The number of bikes in the rebalancing vehicle $m$ at the beginning of time slot $t$ |
| $V_{m,t}^{Slot}$ | The number of slots in the rebalancing vehicle $m$ at the beginning of time slot $t$ |
| $V_{m,t}^{Loc}$ | The location of vehicle $m$ at the beginning of time slot $t$ |
| $D_{i,t}$ | The net demands of the station $i$ in the time slot $t$. The net demand equals to the return demand minus the rent demand. |
| $D_{i,t:t+n}^{pred}$ | The predicted future net demands of the station $i$ in the next $n$ time slots from the beginning of the time slot $t$ |
| $C_i^{station}$ | The capacity of maximum parking bikes of the station $i$, $C_i^{station} = s_{i,t}^{Bike} + s_{i,t}^{Dock}$ |
| $C_m^{vehicle}$ | The capacity of maximum carrying bikes of the vehicle $m$, $C_m^{vehicle} = v_{m,t}^{Bike} + v_{m,t}^{Slot}$ |
| $L_{i,t}$ | The customer loss occurred in the station $i$ in time slot $t$ |
| $P_{i,j}$ | The routing distance between station $i$ and station j |
| $\alpha$ | The price of a single trip riding the shared bike (dollar) |
| $\beta$ | The mileage rate (dollar per mile) |
| $z_{i,j,m,t}$ | Binary, the routing decision variable that sets to 1 for vehicle $m$ in the time slot $t$ if traveling from station $i$ to station $j$ |
| $y_{j,m,t}$ | Integer, the repositioning decision variable for vehicle $m$, denoting the number of bikes to pick-up/drop-off when visiting target station $j$ in time slot $t$ |
| $r_t$ | The reward earned in time slot $t$ |
| $\mathcal{O}_t$ | A sample consists of $S_t, a_t, r_t$ and $S_{t+1}$, which records the transition from the time slot $t$ to the time slot $t+1$ |
| $a$ | Rebalancing action, a tuple consists of the routing decision and repositioning decision $a = (z_{i,j,m,t}, y_{j,m,t})$ |
| $\epsilon$ | The probability of selecting a random action during the training process of DRL |
| $\gamma$ | Time discount factor |
| $\rho$ | The probability of doing no rebalancing action during the initial training process |

# REFERENCES


[1] Abubakr O. Al-Abbasi, Arnob Ghosh, and Vaneet Aggarwal. 2019. DeepPool: Distributed Model-Free Algorithm for Ride-Sharing Using Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems* 20, 12 (2019), 4714–4727. DOI:https://doi.org/10.1109/TITS.2019.2931830

[2] C. Archetti and M. G. Speranza. 2012. Vehicle routing problems with split deliveries. *International Transactions in Operational Research* 19, 1–2 (2012), 3–22. DOI:https://doi.org/10.1111/j.1475-3995.2011.00811.x

[3] Jiayu Chen, Abhishek K Umrawal, Tian Lan, and Vaneet Aggarwal. 2021. DeepFreight: A Model-free Deep-reinforcement-learning-based Algorithm for Multi-transfer Freight Delivery. *arXiv preprint arXiv:2103.03450* (2021).

[4] Federico Chiariotti, Chiara Pielli, Andrea Zanella, and Michele Zorzi. 2018. A dynamic approach to rebalancing bike-sharing systems. *Sensors (Switzerland)* 18, 2 (2018), 1–22. DOI:https://doi.org/10.3390/s18020512

[5] Claudio Contardo, Catherine Morency, and Louis-Martin Rousseau. 2012. Balancing a dynamic public bike-sharing system. *Cirrelt* (2012).

[6] Divvy. 2021. Single Ride. Retrieved from https://www.divvybikes.com/pricing/single-ride

[7] Yubin Duan, Jie Wu, and Huanyang Zheng. 2018. A Greedy Approach for Vehicle Routing When Rebalancing Bike Sharing Systems. In *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, Abu Dhabi, United Arab Emirates, 1–7. DOI:https://doi.org/10.1109/GLOCOM.2018.8647755

[8] Elliot Fishman. 2014. Bikeshare: Barriers, facilitators and impacts on car use.

[9] Elliot Fishman. 2016. Bikeshare: A review of recent literature. *Transport Reviews* 36, 1 (2016), 92–113.

[10] Supriyo Ghosh, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. 2015. Dynamic redeployment to counter congestion or starvation in vehicle sharing systems. *Proceedings of the 8th Annual Symposium on Combinatorial Search, SoCS 2015* 2015-Janua, (2015), 230–231.

[11] Supriyo Ghosh, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. 2017. Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research* 58, (2017), 387–430. DOI:https://doi.org/10.1613/jair.5308

[12] Supriyo Ghosh, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. 2017. Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research* 58, (2017), 387–430. DOI:https://doi.org/10.1613/jair.5308

[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. MIT press Cambridge.

[14] Christian Kloimüllner, Petrina Papazek, Bin Hu, and Günther R. Raidl. 2014. Balancing bicycle sharing systems: An approach for the dynamic case. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8600, June (2014), 73–84. DOI:https://doi.org/10.1007/978-3-662-44320-0_7

[15] Yexin Li, Yu Zheng, and Qiang Yang. 2018. Dynamic bike reposition: A spatio-temporal reinforcement learning approach. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), 1724–1733. DOI:https://doi.org/10.1145/3219819.3220110

[16] Meghna Lowalekar, Pradeep Varakantham, Supriyo Ghosh, Sanjay Dominik Jena, and Patrick Jaillet. 2017. Online repositioning in bike sharing systems. *Proceedings International Conference on Automated Planning and Scheduling, ICAPS* (2017), 200–208.

[17] Hao Luo, Fu Zhao, Wei Qiang Chen, and Hua Cai. 2020. Optimizing bike sharing systems from the life cycle greenhouse gas emissions perspective. *Transportation Research Part C: Emerging Technologies* 117, September 2019 (2020), 102705. DOI:https://doi.org/10.1016/j.trc.2020.102705

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (2013), 1–9.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (2013), 1–9.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533. DOI:https://doi.org/10.1038/nature14236

[21] National Association of City Transportation Officials. 2021. Shared Micromobility in the U.S.: 2019.

[22] Takuma Oda and Carlee Joe-Wong. 2018. MOVI: A Model-Free Approach to Dynamic Fleet Management. *Proceedings - IEEE INFOCOM* 2018-April, (2018), 2708–2716. DOI:https://doi.org/10.1109/INFOCOM.2018.8485988

[23] Takuma Oda and Carlee Joe-wong. 2018. MOVI : A Model-Free Approach to Dynamic Fleet Management. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications* (2018), 2708–2716.

[24] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L. Medaglia. 2013. A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225, 1 (February 2013), 1–11. DOI:https://doi.org/10.1016/j.ejor.2012.08.015

[25] Robert Regue and Will Recker. 2014. Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem. *Transportation Research Part E: Logistics and Transportation Review* 72, (2014), 192–209. DOI:https://doi.org/10.1016/j.tre.2014.10.005

[26] Susan Shaheen, Stacey Guzman, and Hua Zhang. 2010. Bikesharing in Europe, the Americas, and Asia. *Transportation Research Record* 2143 (2010), 159–167. DOI:https://doi.org/10.3141/2143-20

[27] Jia Shu, Mabel C. Chou, Qizhang Liu, Chung Piaw Teo, and I. Lin Wang. 2013. Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Operations Research* 61, 6 (2013), 1346–1359. DOI:https://doi.org/10.1287/opre.2013.1215

[28] C. S. Shui and W. Y. Szeto. 2018. Dynamic green bike repositioning problem – A hybrid rolling horizon artificial bee colony algorithm approach. *Transportation Research Part D: Transport and Environment* 60, (2018), 119–136. DOI:https://doi.org/10.1016/j.trd.2017.06.023

[29] C. S. Shui and W. Y. Szeto. 2020. A review of bicycle-sharing service planning problems. *Transportation Research Part C: Emerging Technologies* 117, April 2019 (2020), 102648. DOI:https://doi.org/10.1016/j.trc.2020.102648

[30] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489. DOI:https://doi.org/10.1038/nature16961

[31] Carlos M. Vallez, Mario Castro, and David Contreras. 2021. Challenges and opportunities in dock-based bike-sharing rebalancing: A systematic review. *Sustainability (Switzerland)* 13, 4 (2021), 1–26. DOI:https://doi.org/10.3390/su13041829

[32] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-Learning. *30th AAAI Conference on Artificial Intelligence, AAAI 2016* (2016), 2094–2100. DOI:https://doi.org/10.1609/aaai.v30i1.10295

[33] Tan Wang. 2014. Solving Dynamic Repositioning Problem for Bicycle Sharing Systems: Model, Heuristics, and Decomposition. (2014).

[34] Cong Zhang, Fan Wu, He Wang, Bihua Tang, Wenhao Fan, and Yuanan Liu. 2022. A Meta-Learning Algorithm for Rebalancing the Bike-Sharing System in IoT Smart City. *IEEE Internet of Things Journal* 9, 21 (November 2022), 21073–21085. DOI:https://doi.org/10.1109/JIOT.2022.3176145

[35] J Zhao, M Mao, and X Zhao. 2021. A hybrid of deep reinforcement learning and local search for the vehicle routing problems. *ieeexplore.ieee.org* (2021). Retrieved March 16, 2023 from https://ieeexplore.ieee.org/abstract/document/9141401/?casa_token=joKmbnG-H1kAAAAA:XSSUCwt_NeOCg0twxOJeS3yl3uwiz1o5te-NkOxMisQunkE8_zq3ordTklZNona4NOzpCrE12kA

[36] Xinghua Zheng, Ming Tang, Yuechang Liu, Zhengzheng Xian, and Hankz Hankui Zhuo. 2021. Repositioning bikes with carrier vehicles and bike trailers in bike sharing systems. *Applied Sciences (Switzerland)* 11, 16 (2021). DOI:https://doi.org/10.3390/app11167227

[37] Xiaolu Zhou. 2015. Understanding spatiotemporal patterns of biking behavior by analyzing massive bike sharing data in Chicago. *PLoS ONE* 10, 10 (2015), 1–20. DOI:https://doi.org/10.1371/journal.pone.0137922

[38] 2023. Home: Internal Revenue Service. *Internal Revenue Service | An official website of the United States government*. Retrieved from https://www.irs.gov/