DeepBike: A Deep Reinforcement Learning Based Model for Large-scale Online Bike Share Rebalancing

Zhuoli Yin

zhuoliyin@purdue.edu

Purdue University

Zhaoyu Kou

Purdue University

Hua Cai

Purdue University

Research Article

Keywords: Bike share system, large-scale, dynamic rebalancing, deep reinforcement learning

Posted Date: March 4th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3998473/v1

Additional Declarations: No competing interests reported.

DeepBike: A Deep Reinforcement Learning Based Model for Large-scale Online Bike Share Rebalancing

Zhuoli Yin[a], Zhaoyu Kou[a], Hua Cai[a,b,*]

a. School of Industrial Engineering, Purdue University, West Lafayette, IN 47907, USA

b. Environmental and Ecological Engineering, Purdue University, West Lafayette, IN 47907, USA

*Corresponding author: E-mail: huacai@purdue.edu; Phone: +1-765-494-7701.

Abstract: Bike share systems (BSSs), as a potentially environment-friendly mobility mode, are being deployed globally. To address spatially and temporally imbalanced bike and dock demands, BSS operators need to redistribute bikes among stations using a fleet of rebalancing vehicles in real-time. However, existing studies mainly generate BSS rebalancing solutions for small-scale BSSs or subsets of BSSs, while deploying small-size rebalancing fleets. How to produce online rebalancing solutions for large-scale BSS with multiple rebalancing vehicles to minimize customer loss is critical for system operation yet remains unsolved. To address this gap, we proposed a deep reinforcement learning based model — *DeepBike* — that trains deep Q-network (DQN) to learn the optimal strategy for dynamic bike share rebalancing. *DeepBike* uses real-time states of rebalancing vehicles, stations and predicted demands as inputs to output the long-term quality values of rebalancing actions of each rebalancing vehicle. Rebalancing vehicles could work asynchronously as each individually runs the DQN. We compared the performance of the proposed *DeepBike* against baseline models for dynamic bike share rebalancing based on historical trip records from Divvy BSS in Chicago, which possesses more than 500 stations and 16 rebalancing vehicles. The evaluation results show that our proposed *DeepBike* model was able to better reduce customer loss by 111.09% and 57.6% than the mixed integer programming and heuristic-based models, respectively, and increased overall net profits by 101.26% and 220.01%, respectively. The *DeepBike* model is effective for large-scale dynamic bike share rebalancing problems and has the potential to improve the operation of shared mobility systems.

Keywords: Bike share system, large-scale, dynamic rebalancing, deep reinforcement learning

1. Introduction

Rebalancing bikes to accommodate real-time customer demands is becoming critical for bike share system (BSS) operation as BSS is scaling up in major cities worldwide (Yin, Hardaway, et al., 2023). The accessibility, convenience, and low cost of BSS have attracted lots of customers. Additionally, BSS is regarded as a promising mobility solution for cities to help mitigate congestion and reduce greenhouse gas (GHG) emissions (Fishman, 2016; Shaheen et al., 2010). Due to the popularity and benefits of BSSs, BSS companies have continued to expand their existing systems or launch new ones (Fishman, 2016; Shaheen et al., 2010). Nevertheless, spatiotemporal demand heterogeneity across a city often leads to the imbalanced distribution of bike share ridership. This imbalance will then cause bikes to be overstocked or depleted in stations and impede the returning or renting of bikes, which thereby results in customer loss, and revenue decrease, as well as a GHG emission increase (Ghosh et al., 2017). Consequently, to avoid potential customer loss, the system operator needs to regularly rebalance bikes among stations to maintain a reasonable distribution across the service region (Fishman, 2014). Among various rebalancing strategies, dynamic bike share rebalancing becomes a major solution for BSS operation. This dynamic strategy is crucial because it monitors the system in real-time to maintain the stations to be well-functioning throughout the day, in contrast to static rebalancing which merely optimizes the initial distribution of bikes in a system (Gleditsch et al., 2022; Shui & Szeto, 2020; Vallez et al., 2021). Such a strategy is typically modeled as a Dynamic Bike Share Rebalancing Problem (*DBSRP*). In specific, the rebalancing is usually conducted by a fleet of automobiles, such as trucks or vans (referred to as *rebalancing vehicles* hereafter) (Citi Bike, 2023). The rebalancing process is sequential and involves two entangled decisions: (1) *routing* decisions, which choose a series of bike stations that a rebalancing vehicle is to visit, and (2) *repositioning* decisions, which determine the number of bikes to be collected from or distributed into stations decided by the *routing* decisions.

DBSRP has been widely studied in the literature, but the prior works still have major limitations in addressing the online[1] and large-scale needs of *DBSRP* (detailed discussions of the existing literature are provided in Section 2). Traditional approaches primarily solved *DBSRP* through Mixed Integer Programming (MIP) (Contardo et al., 2012; Ghosh et al., 2015, 2017; T. Wang, 2014; Zheng et al., 2021). However, they are only applicable to offline scenarios. Their

[1] We will refer to online and real-time interchangeably.

solutions rely on complete data of the entire planning horizon, but in dynamic rebalancing, customer demands for the upcoming planning horizon are unknown to the decision-makers. In recent years, to generate online rebalancing solutions, heuristic algorithms and artificial intelligence-based approaches (e.g., deep learning and reinforcement learning) have been employed (Hu et al., 2021; Li et al., 2018; Shui & Szeto, 2018). However, these algorithms were only designed for and evaluated on small-scale BSSs (typically less than 300 bike stations) or they divided the study region into small clusters and dispatched only one rebalancing vehicle in each cluster to simplify the problem (Vallez et al., 2021). Additionally, solutions from heuristic algorithms are myopic by looking forward only a few time steps (Lowalekar et al., 2017). In contrast, the size of BSS in major cities, such as Chicago and New York City in the U.S., has increased to be large-scale to cover wider urban regions, including more than 500 stations and 10 rebalancing vehicles in service (General Bikeshare Feed Specification, 2023). Thus, the scale of current online solutions no longer suffices the needs of these expanded systems.

Models that are capable of generating online and far-sighted solutions for large-scale BSSs are essential to support effective rebalancing operations. The advancement of Deep Reinforcement Learning (DRL) has demonstrated the potential opportunity to reach this goal. Integrating Q-learning with a deep neural network, DRL presents a strong representation learning capability for high-dimensional and large-scale system observations to estimate the long-term (i.e., the period covering the current time to the end of the planning horizon) benefits of decisions (de Bruin et al., 2018; Mnih et al., 2013, 2015). Once well trained, it has promising effectiveness in online sequential decision-making based on real-time observations (J. Wang & Sun, 2020). Leveraging these benefits, DRL has been widely adopted in urban mobility management, such as on-demand ride sharing dispatching (Qin et al., 2019). Thus, given the sequential decision-making nature of *DBSRP* (the formulation is detailed in Section 3), DRL could also be employed to generate online and far-sighted rebalancing solutions for this problem.

In this study, we proposed a DRL-based framework—*DeepBike*— to generate the optimal online solutions for large-scale *DBSRP*. *DeepBike* enables real-time dispatching for multiple rebalancing vehicles to rebalance bikes. In the decision-making module within the *DeepBike* framework (referred to as *agent* hereafter), we designed a Deep Q-Network (DQN), whose inputs are real-time large-scale BSS state observations, and whose outputs are the long-term quality values (Q-value) of rebalancing actions of each rebalancing vehicle. Through the

interaction with the BSS simulator, the agent iteratively learns to approximate the Q-values of rebalancing actions and selects the highest-valued action for individual rebalancing vehicles. The comparison with an MIP-based algorithm and a heuristic-based algorithm demonstrates that our algorithm outperforms these benchmarks in reducing customer loss and improving the overall net profits.

The remainder of this paper is organized as follows. Section 2 discusses the related literature, research gaps, and the main contributions of this study. Section 3 provides a full description and mathematical formulation of *DBSRP*. Section 4 discusses input data, and the *DeepBike* framework including the simulator, demand prediction, and DRL model. Additionally, it introduces the baseline algorithms and metrics that measure the performance of different algorithms. In Section 5, the model training results, and model performance comparisons are described. Last, Section 6 summarizes the findings and discusses the limitations of this study and future research directions.

2. Literature review

Existing studies have proposed various algorithms to efficiently solve *DBSRP* since the emergence of BSSs. The algorithms could mainly be classified into three groups: Mixed Integer Programming (MIP), heuristic, and artificial intelligence. Table 1 summarizes the models in existing literature and also compares them with our proposed model. As an NP-hard problem, the complexity of *DBSRP* is governed by factors such as the scale of the bike stations (i.e. the number of stations and capacity of each station), the size of the rebalancing fleet (i.e. the number of rebalancing vehicles and capacity of each vehicle), and the length of planning horizon (i.e. the number of decisions to be made over the entire planning horizon) (Lowalekar et al., 2017; Schuijbroek et al., 2017; Shui & Szeto, 2020).

Traditionally, researchers modeled *DBSRP* through MIP or linear programming (LP), treating it as a variant of the vehicle routing problem (VRP). However, these models only offer offline solutions. For example, Contardo et al. (2012) proposed a mathematical formulation of *DBSRP* for peak hours and derived the lower and upper bounds via Dantzig-Wolfe decomposition and Benders decomposition. Shu et al. (2013) addressed the *DBSRP* based on the stochastic network flow model and solved it via LP. (Ghosh et al., 2015, 2017; Zheng et al., 2021) handled *DBSRP* by an MIP model with the objective of maximizing the overall profit. To

accelerate the computation due to the NP-hard nature of the problem, they first aggregated all individual stations into different clusters and viewed each cluster as an abstract station. Then they solved the *DBSRP* at the abstract station level by Lagrangian dual decomposition. Similarly, Wang (2014) proposed an MIP formulation for *DBSRP* and solved the model through two heuristic methods and one exact approach. These traditional mathematical algorithms yield stable and exact solutions but require complete information on the considered planning horizon, generating only offline solutions. They are unable to adapt to upcoming real-time planning horizons with unknown information. Consequently, there exists a research gap in making online rebalancing decisions during the daytime to accommodate dynamically changing bike usage demands.

To facilitate online rebalancing decisions, many studies developed heuristic algorithms but their solutions are often myopic. For example, Lowalekar et al. (2017) proposed a Greedy Online Anticipatory Heuristic approach capable of offering online solutions within a short computation time (e.g., one minute). This approach picks the best rebalancing action for each rebalancing vehicle by anticipating several timeslots ahead and approximating the value of each valid action. Shui and Szeto (2018) employed a rolling horizon method to break down the planning period, solving a static bike repositioning sub-problem within each segment. Additionally, within their model, an enhanced artificial bee colony (EABC) algorithm and a route truncation heuristic were integrated to optimize the route design in each stage. Chiariotti et al. (2018) exploited historical data to predict future bike station conditions to decide the desired inventory level of each bike station. Then they selected the rebalancing actions that maximize the "survival time" of stations. Even though the aforementioned approaches could generate online rebalancing solutions, the selection of actions is myopic because they heuristically approximated the benefits of rebalancing actions by looking ahead only a few steps within the entire planning horizon, which is likely to reach the local optimum. However, *DBSRP* typically operates on an hourly basis within a day so each rebalancing action not only has an immediate effect on the BSS but also impacts subsequent system dynamics (O'Mahony & Shmoys, 2015; Li et al., 2018). For example, stations in the downtown area have high morning check-in demands for commute purposes and have symmetric high afternoon check-out demands (Zhou, 2015). Such stations could potentially achieve self-rebalancing status without the need for extra rebalancing effort. An online rebalancing decision that greedily moving bikes out of the self-rebalancing stations in the

morning might cause greater customer loss and rebalancing cost because it overlooks the path dependence in the full planning horizon and further results in supply scarcity in those stations in the afternoon (D. Chen & Sakai, 2022). Therefore, developing online as well as farsighted solutions that evaluate the payoff of rebalancing decisions over the entire planning horizon is still a gap for *DBSRP*.

Prior works also leverage the capability of artificial intelligence in sequential decision-making to solve *DBSRP* and provide online solutions (Yin, Kou, et al., 2023). However, their models together with previous works with MIP and heuristic algorithms were only designed on small-scale BSSs or a small subset of BSSs with the deployment of a small size of rebalancing fleet in BSSs (less than 300 bike stations and fewer than 5 rebalancing vehicles in the fleet) (Vallez et al., 2021). For example, Li et al. (2018) dealt with *DBSRP* through a spatial-temporal reinforcement learning framework aiming to minimize long-term customer loss. They first proposed an interdependent inner-balance clustering algorithm and then designated only one vehicle to each cluster. Nevertheless, their approach was only applicable to a subset of the Citi Bike system and overlooks spatiotemporal imbalance of customer demand among those clusters. Assigning one vehicle to each cluster is inadequate to handle this disparity. Given the heterogeneity of customer demands across different regions, such as downtown versus suburban areas, the allocation of rebalancing vehicles should be smartly and efficiently adjusted for global demands and overall system dynamics in the entire planning horizon. As such, their model is overly simplified for the operation of multi-rebalancing vehicles and large-scale stations in current BSSs. Notably, (Zhu et al., 2022) also employed reinforcement learning to tackle large-scale *DBSRP*. However, they employed a user incentivized crowd-sourcing rebalancing strategy which is different from the *DBSRP* with centralized rebalancing managed by the operators. Chen et al. (2021) applied deep learning to learn from historical solution instances and utilized the trained model for future cases. The solutions heavily rely on the quality of solution instances from which their model learned. Additionally, the model can only generate rebalancing decisions for one rebalancing vehicle, regardless of the coordination between multiple rebalancing vehicles in service. Additionally, in terms of the scale of solutions in previous studies, such as (Contardo et al., 2012; Shu et al., 2013; Kloimüllner et al., 2014; T. Wang, 2014; Regue & Recker, 2014), they proposed the *DBSRP* models for BSSs of fewer than 100 stations. Other works such as (Ghosh et al., 2015; Lowalekar et al., 2017; D. Zhang et al., 2017; Ghosh et al., 2017; Brinkmann

et al., 2020) dealt with *DBSRP* with fewer than 400 stations in the BSSs. Furthermore, in (Chiariotti et al., 2018; Ghosh et al., 2016; T. Wang, 2014), the problem was simplified by allocating a single rebalancing vehicle to the service region. However, current station-based BSSs in major cities have grown to possess more than 500 stations in their service regions. For instance, as of 2023, Citi Bike in New York City and Velib' Metropole in Paris maintained stations of 2,000 and 1,472, respectively (*General Bikeshare Feed Specification*, 2023). Moreover, current operators typically deploy multiple rebalancing vehicles in lieu of a single one to meet rebalancing needs. For example, (Sun, 2021) reported that BSSs in Chicago, Boston and Los Angeles need 16, 10 and 5 rebalancing vehicles, respectively. Therefore, existing algorithms fall short of accommodating the expanding scale of BSSs and there is a gap in developing models adequate for generating high-quality online rebalancing solutions for large-scale BSSs with multiple rebalancing vehicles.

To address the above-mentioned research gaps, we proposed a DRL-based model—*DeepBike*— to generate the optimal strategy for the dynamic rebalancing of large-scale BSS in real-time. Specifically, in the proposed framework, an agent interacts with the BSS simulator (which simulates the dynamics of the BSS) iteratively to learn the optimal policy, whose objective is to maximize the system's overall profits. We designed a Deep Q-Network (DQN) for the agent to estimate the long-term (from the current timeslot to the end of the planning horizon) benefits of the rebalancing actions, given the global system observations as network inputs. The agent selects the rebalancing action with the highest benefit for each rebalancing vehicle. Our model differs from the existing work in an important aspect that we incorporated large-scale, real-time, and farsighted rebalancing decision-making simultaneously, catering to BSS operations in large-scale cities nowadays. In the context of the existing literature, we summarize the main contributions of this study as follows:

(1) We proposed a DRL-based framework (*DeepBike*) tailored for *DBSRP.* Different from existing literature, our model focuses on optimizing large-scale *DBSRP* in real-time, without partitioning the service region into smaller clusters.

(2) We constructed a convolutional neural network (i.e., DQN) which is capable of processing large-scale system observations covering the entire service region. Once the network converges, the *DeepBike* model generates instantaneous rebalancing decisions for each rebalancing vehicle every hour in a day based on real-time observation.

(3) We trained the DQN through the Double Deep Q-Network (DDQN) algorithm, enforcing the network to approximate the long-term benefits of rebalancing decisions from the current step to the end of the planning horizon to reach the far-sighted optimum.

(4) We evaluated the *DeepBike* model on the real-world dataset of Divvy Bike Share in Chicago which has more than 500 bike stations and 16 rebalancing vehicles. The empirical results demonstrate the effectiveness of our model in improving the overall system's profits.

Table 1. Summary of the related literature on dynamic bike share rebalancing problem and comparison with this study.

Paper	Offline/ Online	# of stations	Clustering/ # of clusters	# of rebalancing vehicles	Planning horizon	Algorithm
Contardo et al. (2012)	Offline	25/50/100	Yes/9	5	2 hours	MIP
Shu et al. (2013)	Online	100	No	/	24 hours	LP
Ghosh et al. (2015)	Offline	95	Yes/25	4	7 hours	MIP
Ghosh et al. (2017)	Offline	95/305	Yes/25/30	3/5	24 hours	MIP
Zheng et al. (2021)	Offline	60/95/305	Yes/12	9/13/45	19 hours	MIP
Wang (2014)	Offline	12/24/36/48	No	1	1 hour/2 hours	MIP/Heuristic
Kloimüllner et al. (2014)	Offline	30/60/90	No	1/2/3/4/5	4 hours/8 hours	Heuristic
Regue and Recker (2014)	Offline	61	No	2	24 hours	Heuristic
Lowalekar et al. (2017)	Online	95/305	No	3/6	6 hours	MIP/Heuristic
Zhang et al. (2017)	Online	100/104/200	No	2	9 hours	Heuristic
Shui and Szeto (2018)	Online	30/60/90/120/180	No	1	6/12/18 time steps	Heuristic
Chiariotti et al. (2018)	Online	280	No	1	24 hours	Heuristic
Li et al. (2018)	Online	389	Yes/4	4	24 hours	DRL
Brinkmann et al. (2020)	Online	35/169	No	1/2/3/4	24 hours	MIP+heuristic
Chen et al. (2021)	Online	149	No	1	8 hours	DL
This study (*DeepBike*)	**Online**	**578**	**No**	**16**	**24 hours**	**DRL**

Note: MIP: Mixed Integer Programming; LP: Linear Programming; DL: Deep Learning; DRL: Deep Reinforcement Learning

3. Problem formulation

In this section, we begin by illustrating the *DBSRP* with an example. Then we describe the generic MIP formulation of *DBSRP*. Notations employed throughout this paper are detailed in Appendix Table A-1.

As demonstrated in Fig. 1, the rebalancing process involves a rebalancing vehicle, typically a truck or van, initiating its routes from a depot at the start of the planning horizon and ending its route at the same depot. The primary objective of this rebalancing vehicle is to reposition shared bikes across various stations, either by picking up or dropping off bikes. This rebalancing process ensures a reasonable inventory level of bikes at each station, thereby facilitating uninterrupted bike usage for customers. Meanwhile, customers rent and return bikes at these stations throughout the day. In the event that a station either runs out of bikes or docking space, customers fail to rent or park bikes at their visited stations. These service failures are considered as customer loss.

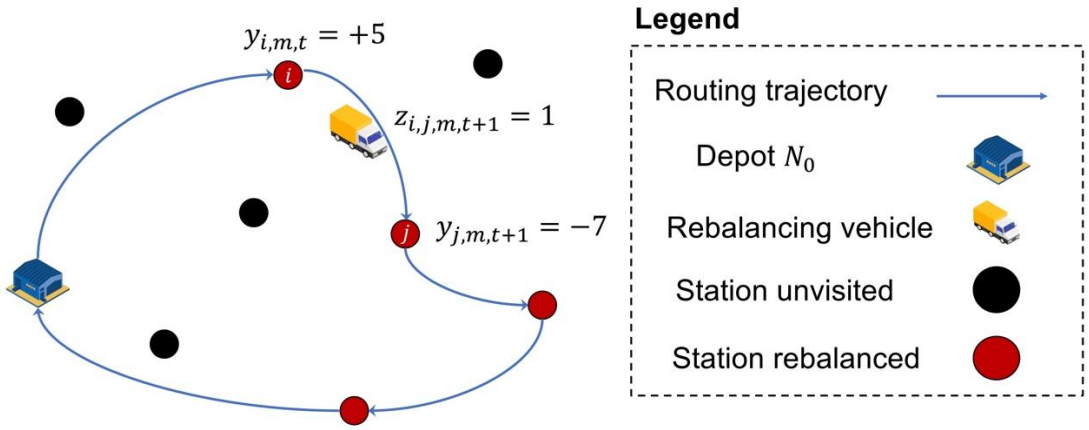

Fig. 1. An illustration of *DBSRP* studied in this work. In this example, the rebalancing vehicle m departs from the depot N_0. It routes to bike station i and moves 5 bikes into the station in timeslot t and then travels to bike station j and moves 7 bikes out from the station in timeslot $t+1$. The routing decision that vehicle m travels from station i to station j is denoted as $z_{i,j,m,t+1} = 1$, while the repositioning decisions at station i in timeslot t and at station j in timeslot $t+1$ are denoted as $y_{i,m,t}$ and $y_{j,m,t+1}$, respectively. After visiting the subsequent stations, the rebalancing vehicle returns to the depot at the end of the planning horizon.

We depict the comprehensive dynamics of *DBSRP* in Eqs. (1) – (12) based on MIP formulations proposed by (Ghosh et al., 2017; Luo et al., 2020). These formulations describe customer interaction with BSS and operational constraints involved in the rebalancing process, where decision variables involve repositioning variables $y_{i,m,t}$ and routing variables $z_{i,j,m,t}$:

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

s.t.

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

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

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

$$V_{m,t+1}^{Bike} = V_{m,t}^{Bike} + \sum_{m\in M} -y_{j,m,t} \quad \forall m \in M, t \in T \quad (5)$$

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

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

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

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

$$S_{i,0}^{Bike} = C_0^{station} \ \forall i \in N\,, \qquad V_{m,0}^{Bike} = C_0^{vehicle} \ \forall m \in M \quad (10)$$

$$0 \le S_{i,t}^{Bike} \le C_i^{station}, 0 \le V_{m,t}^{Bike} \le C_m^{vehicle} \quad \forall i \in N, m \in M, t \in T \quad (11)$$

$$z_{i,j,m,t} \in \{0,1\}, y_{i,m,t}\ is\ integer\ \forall i, j \in N, m \in M, t \in T \quad (12)$$

The objective function that is described in Eq. (1) minimizes the total costs of customer loss (the sum of the customer loss from each station i in each timeslot t, $CL_{i,t}$, the definition of which is detailed in Section 4.4) and fuel cost of rebalancing vehicles that route to target stations (the sum of the routing distance between stations i and j, $P_{i,j} \cdot z_{i,j,m,t}$). Additionally, in Eq. (1), α represents the average price paid by customers for a single bike share trip and β represents the average fuel cost per mile of vehicle use. The objective function also considers the tradeoff between the reduced customer loss (to improve the revenue from trips) and the cost of routing rebalancing vehicles that is required.

For constraints, Eqs. (2)-(3) conserve the in- and out-flows of bikes in the station. The number of bikes at the station either after being rebalanced or after the rent/return of customers should always stay within the station's capacity, $C_i^{station}$. Eq. (4) calculates the customer loss in

each station, which is the difference between the expected inventory level of the station if all customer demands could be fulfilled after being rebalanced (that is, $S_{i,t}^{Bike} + \sum_{m \in M} y_{i,m,t} + D_{i,t}$) and the actual inventory level constrained by Eq. (2) (that is, $S_{i,t+1}^{Bike}$). Eq. (5) defines the transition of bikes stocked in the rebalancing vehicles over time. The inflow and outflow of the bikes on the vehicles are the result of the repositioning decision $\sum_{m \in M} -y_{j,m,t}$. Eq. (6) enforces that the rebalancing vehicles departing from a station $\sum_{j \in N} z_{i,j,m}^{t}$ could only be the ones that arrive at this station in the previous 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. Intuitively, $\sum_{j \in N} z_{N_0,j,m,0}$ and $\sum_{i \in N} z_{i,N_o,m,|T|}$ are set to be 1 for every rebalancing vehicle m to ensure it to depart from the depot initially and return to it at the end. Eqs. (7) and (8) guarantee that, in any timeslot, no more than one vehicle is visiting the same station, and each vehicle can only visit at the most one station, respectively. These constraints avoid the risk of multiple vehicles rebalancing the same station, thereby preventing offsetting effects. Eq. (9) couples the repositioning decision and routing decision. It enforces that rebalancing vehicles can only pick up or drop off bikes in physically visited stations. That is, $y_{j,m,t}$ will be forced to be zero if no rebalancing vehicles $\sum_{i \in N} z_{i,j,m,t}$ are assigned to visit station j departing from each station i. Eq. (10) sets the initial inventory level of bikes among stations (constant $C_0^{station}$) and the initial capacity of the rebalancing vehicles (constant $C_0^{vehicle}$). Eq. (11) confines the maximum capacities of bike stations and rebalancing vehicles, denoted by the constants $C_i^{station}$ and $C_m^{vehicle}$, respectively. Eq. (12) defines that the routing decision variable $z_{i,j,m,t}$ is a binary and repositioning decision variable $y_{i,m,t}$ is an integer. The variable $z_{i,j,m,t}$, when setting to 1, represents vehicle m in the timeslot t will travel from station i to station j. When setting $y_{i,m,t}$ to an integer *K*/-*K*, it indicates rebalancing vehicle m moves *K* bikes into/out of the station i in the timeslot t.

4. Data and methods

This section introduces our proposed model—*DeepBike*—designed for the *DBSRP*, as well as the associated data used to train and evaluate the model. Specifically, Section 4.1 describes the Divvy BSS data from Chicago and the preprocessing to align the data with our framework. In Section 4.2, we detail the *DeepBike* framework based on the RL paradigm, involving the environment, agent, state, action, and reward. More specifically, Section 4.2.1

establishes a BSS simulator (i.e., the environment)[2], which replicates BSS dynamics and provides information on BSS status for the agent to make rebalancing decisions. Following this, we develop a prediction model in Section 4.2.2, which forecasts next-hour customer demands across the stations based on the BSS status information. This model offers predicted demands for the agent to assist in generating rebalancing actions. In Section 4.2.3, we devise the state space, action space, and reward function within the framework of DRL. Then, the Double-DQN algorithm and DQN architecture are introduced in Sections 4.2.4 and 4.2.5, respectively. Section 4.3 introduces baseline models for comparative analysis. Last, Section 4.4 outlines evaluation metrics for measuring *DBSRP* algorithm performance.

4.1. Case study data and preprocessing

We chose Divvy BSS in Chicago as our case study system because Divvy is a well-established station-based BSS and has been actively used since 2013. Divvy's trip records and station information were employed to build the simulator to train the *DeepBike* model. The historical trip records are publicly available, spanning from the year 2013 to the year 2023 (*Divvy System Data*, 2023). For our case study, we selected four consecutive weeks of historical trip data from Sept. 2nd, 2019 (Monday) to Sept. 29th, 2019 (Sunday). This period includes over 449,000 trip records. Specifically, we used the start and end time of a bike trip, and the trip's origin and destination station IDs from the trip records. Following this, the data from the first three weeks was employed for model training and the data of the remaining one week was reserved for evaluation purposes. The station information of Divvy in the year 2019 was collected through GBFS (General Bikeshare Feed Specification) API. The station information includes the station ID, the corresponding location (latitude & longitude), and capacity. Those station IDs are consistent with the trips' origin and destination station IDs. In addition to the above data, we also utilized Divvy's third quarter data in 2018 to train a separate demand prediction model, prior to *DeepBike* model training (detailed in SI-S2).

In the preprocessing phase, we trimmed 40 inactive stations that displayed no trip activities in September 2019. This results in 578 active physical stations across Chicago. We considered a rectangle region bounded by the locations of active stations as the service region. Additionally, following the gridding approach used by Al-Abbasi et al. (2019) and Oda and Joe-

[2] In the following content, the terms "simulator" and "environment" are used interchangeably.

Wong (2018), we divided the service region into 51×51 grids with each grid measuring $400\ m \times 700\ m$. This gridding approach aligns the service region with the input requirements of the DQN (detailed further in Section 4.2.2). Stations residing within the same grid are consolidated as one abstract station[3], leading to 451 abstract active stations in total. This results in the number of physical stations within a single abstract station ranging from zero to seven. Additionally, considering the grid size, physical stations within one abstract station are close enough to each other such that the subsequent allocation of rebalancing within an abstract station can be intuitively determined by the operators (Ghosh et al., 2017; Osorio et al., 2021). The capacity of the abstract station is therefore the cumulative capacities of all physical stations within that grid. The estimated route between two abstract stations is calculated based on the average distance between all physical station pairs within the respective grids. For each station pair, the route distance is approximated using road network distance from the Open Street Map dataset for cars in Chicago based on 578 stations' physical locations.

The physical address of the bike service warehouse of Divvy (i.e., the depot) is located at (41.89002, -87.658054) on Google Maps, which corresponds to the grid (27, 26) in this study. The rebalancing fleet of Divvy is estimated to include at least 16 automobiles in service, as reported by Sun (2021). Considering the work shift, we assumed that these vehicles operate daily from 6:00 am to 11:00 pm.

4.2. DeepBike framework

The overall framework of *DeepBike* is illustrated in Fig. 2 and each component within the framework is detailed in the subsections. A *DeepBike* agent interacts with the simulator (Section 4.2.4) through three components—states, actions and rewards (Section 4.2.1). The components are continuously stored as experiences. Based on the observed states, the *DeepBike* estimates the Q-values (long-term benefits) of actions via a Deep Q-Network (Section 4.2.2). The *DeepBike* iteratively improves its Q-value estimation by learning from the collected experiences based on the Double Deep Q-Network algorithm (Section 4.2.3).

[3] In the following context, the abstract station will be referred to as the station or grid.

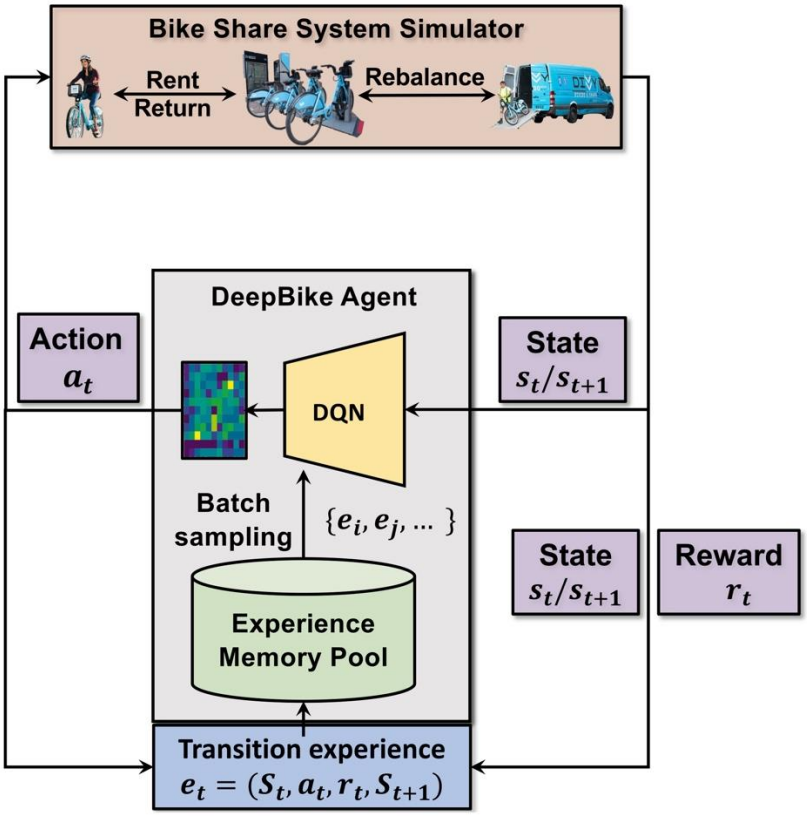

Fig. 2. Illustration of the *DeepBike* learning framework following the RL paradigm. The agent iteratively interacts with the simulator to collect experiences to learn the relationship between the input states and the long-term benefits of actions. The agent generates an action by DQN based on the observed state. The simulator executes the action and simulates the bike usage. It then sends the updated state and reward to the agent as feedback. States, actions, and rewards are continuously stored in the memory pool, from which the agent updates the parameters of its DQN based on the experiences sampled.

4.2.1. State, action, and reward

In the interactions between the *DeepBike* agent and the BSS simulator, the transition from timeslot t to $t+1$ is characterized by three fundamental components: state, action, and reward. Given a specific rebalancing policy π (i.e., the decision-making rule), each transition is formulated by the tuple $(s_t, a_t, r_t, s_{t+1}, \pi)$ (Sutton & Barto, 2018).

"State" depicts the observed status of the environment. In this study, the state is populated from the real-time information of the BSS environment on bike stations, rebalancing vehicles, and predicted future customer demands. Mathematically, we denoted the state at the beginning of timeslot t as $s_t = \left(S_{i,t}^{Bike}, S_{i,t}^{Dock}, V_{m,t}^{Bike}, V_{m,t}^{Slot}, V_{m,t}^{Loc}, D_{i,t:t+1}^{pred}\right), i \in \{1,2,\dots,N\}, m \in \{1,2,\dots,M\}$, where $S_{i,t}^{Bike}$ and $S_{i,t}^{Dock}$ are the number of parked bikes and available docks in station i of

timeslot t, respectively; $V_{m,t}^{Bike}$, $V_{m,t}^{Slot}$ and $V_{m,t}^{Loc}$ represent the number of bikes onboard, available slots, and the location of the rebalancing vehicle m in timeslot t; $D_{i,t:t+1}^{pred}$ is the forecasted customer demands across the grids in the next timeslot, which is detailed in S3 in SI).

"Action" is the decision generated based on the current state according to a specific policy, leading to the transition of the environment to a subsequent state. Here, the rebalancing actions align with the decision variables in the MIP model Section 3. Thus, the rebalancing actions generated by *DeepBike* are denoted as $a_{m,t} = (z_{i,j,m,t}, y_{j,m,t})$ for rebalancing vehicle m in timeslot t, where $z_{i,j,m,t}$ is the routing from station i to j and $y_{j,m,t}$ is the number of bikes to be collected/deposited at the target station j. To reduce the computation time, the action space for routing action $z_{i,j,m,t}$ is limited to stations reachable within ten horizontal or vertical grids from the current station, which totals 441 discrete actions. Such a reachability constraint also ensures that a rebalancing vehicle can arrive at a target station within one timeslot. Meanwhile, we aggregated the repositioning actions into seven discrete actions to reduce the computational complexity and avoid inefficient rebalancing: (1) do not move any bike in the target bike station, (2,3) move five in/out of the target station, (4,5) move ten bikes into/out of the target station, (6) move maximum feasible bikes out of the station ("greedy all out"), and (7) move maximum feasible bikes from the rebalancing vehicle into the station ("greedy all in").

"Reward" is the signal received after transitioning from the current state to the next one by taking an action. The overarching aim of an agent in DRL is to select actions that maximize the expected total future rewards until the episode (i.e. the entire planning horizon) terminates (Mnih et al., 2015; Oda & Joe-wong, 2018) as described in Eq. (13):

$$max \sum_{t'=t}^{\infty} \gamma^{t'-t} r(a_{t'}, s_{t'}) \tag{13}$$

where $r(a_{t'}, s_{t'})$ is the reward function calculating a scalar reward value given the state $s_{t'}$ and action $a_{t'}$; $\gamma \in [0, 1]$ is the time discount factor which discounts future rewards; t is the current timeslot; and t' is the future timeslot. In this study, we designed the reward function for a rebalancing vehicle m that executes rebalancing action $a_{m,t}$ in timeslot *t* given the state s_t as:

$$r(a_{m,t}, s_t) = \alpha \cdot (CL_{j,t} - \widehat{CL}_{j,t}) - \beta \cdot P_{i,j} \cdot z_{i,j,m,t} \tag{14}$$

Eq. (14) is a modification to the objective function Eq. (1) to reflect an immediate reward in current timeslot, where $(CL_{j,t} - \widehat{CL}_{j,t})$ is the delta customer loss in the target station j in

timeslot t and the $P_{i,j} \cdot z_{i,j,m,t}$ is the routing distance from station i to j. Specifically, the delta customer loss is the difference between the customer loss $CL_{j,t}$ occurring with executed rebalancing under a given policy and the baseline customer loss $\widehat{CL}_{j,t}$ (i.e., the loss if no rebalancing was performed). This modification results in a scalar reward that can range between positive and negative values, thereby better indicate what *DeepBike* is supposed to accomplish and speed up reaching optimality (Sutton & Barto, 2018). Considering that the *DeepBike* is trained offline before deploying online, baseline customer loss is always attainable following the simulation setup in this work.

4.2.2. DQN architecture

The design of a neural network is crucial in effectively estimating the long-term benefits of actions in a given state (Silver et al., 2016). Such a neural network is known as a Deep Q-Network (DQN) while the long-term benefits of actions corresponding to a state are denoted as the Q-value $Q_\pi(s_t, a_t)$. In this study, we designed a convolutional neural network architecture as the DQN, motivated by (Oda & Joe-wong, 2018; Silver et al., 2016; Zhao et al., 2021).

As presented in Fig. 3, we designed two input channels for the DQN to gather both global and local observations to capture the spatial-temporal information of the entire BSS service region. 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. The DQN output is structured as a two-dimensional 441×7 table to approximate the Q-values for different rebalancing actions. This tabular approach is designed due to the intertwined nature of routing and repositioning. Each table dimension aligns with the action spaces of repositioning and routing as described in Section 4.2.1, respectively. Thus, selecting an action from this table effectively maps to corresponding rebalancing actions in both dimensions simultaneously. Detailed descriptions of the input features and the process by which the convolutional neural network estimates the outputs are provided in the Supplementary Information (SI) Sections S-1 and S-2, respectively.

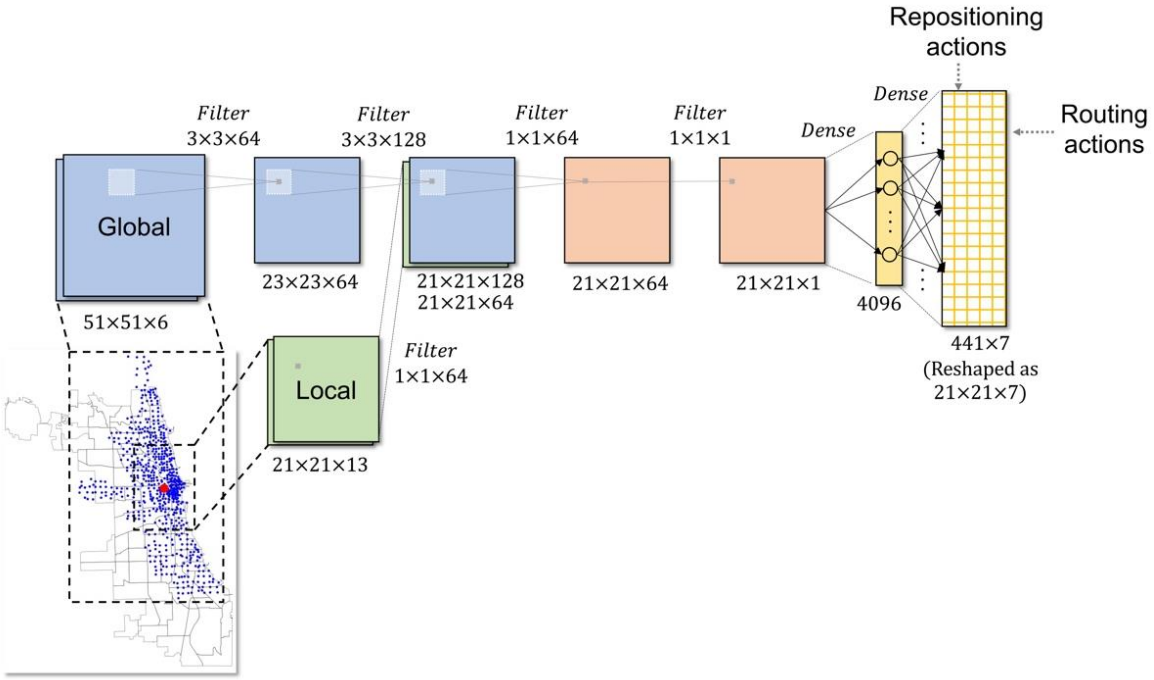

Fig. 3. Deep Q-Network architecture of *DeepBike*.

We further applied an *action mask* on the output Q-value table to eliminate infeasible actions. This *a priori* constraint excludes specific actions from selection, preventing infeasible actions from entering the experience pool and improving the exploration efficiency. Specifically, pixels in the Q-value table (i.e., individual rebalancing actions) will be masked as zero under three conditions: (1) if a grid lacks an installed station; (2) the station has already been assigned to be visited by another rebalancing vehicle (as per Eq. (7),); and (3) if available bikes parked at the intended stations or bike inventory in a rebalancing vehicle are insufficient for fulfilling the rebalancing action associated with this pixel.

4.2.3. Action selection and DQN learning

We first introduce how an action is chosen from the action space and then describe how a DQN is trained to become a better estimator. Essentially, to avoid the overestimation of Q-values, two DQNs with identical architecture (that is, a primary network with weights θ and a target network with weights θ^-) were employed when selecting actions and improving DQN.

During the interaction with the environment, the rebalancing action to be executed is determined by the primary network. For a given state s_t, this network estimates the Q-values of the rebalancing actions for individual rebalancing vehicles (Mnih et al., 2013; Silver et al., 2016). Formally, the Q-value of an action a_t in a state s_t is defined in Eq. (15) as the expected cumulative future rewards yielded from selecting the immediate action and future actions following a policy thereafter:

$$\mathcal{Q}(s_t, a_t; \theta) \equiv \mathbb{E}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_k \mid S = s_t, A = a_t, \pi\right] \quad (15)$$

where $\gamma \in [0,1]$ is a discount factor that compensates for the importance of immediate and future rewards, which was set to be 0.9. Based on Eq. (13), we always choose the highest-valued action $a_t^* = \underset{a_t}{\operatorname{argmax}}\, \mathcal{Q}^*(s_t, a_t; \theta)$ to maximize the cumulative rewards to be received, where $\mathcal{Q}^*(s_t, a_t; \theta)$ denotes the optimal Q-value (Van Hasselt et al., 2016).

Prior to the DQN being fully trained and reaching convergence, the selected action may only be sub-optimal because the Q-values can be poorly estimated. Therefore, the *exploration-exploitation* scheme is applied to trade off gathering a new observation in an unknown state-action space and making the best decision based on the current DQN (Silver, 2014). We implemented the ϵ-greedy policy to select rebalancing actions during the training stage. Specifically, in each iteration, with probability $1 - \epsilon$, we select the highest valued action from the DQN as described above (i.e., exploitation); while with probability ϵ, we choose a random action from the action space (i.e., exploration) (Mnih et al., 2013). Thus, the exploration will lead to discovering more rewarding spaces that are unseen from the existing experiences. In contrast, as the DQN learns to correctly identify the actions yielding higher rewards, exploitation will reinforce the current DQN. In this study, we set the initial ϵ to be the commonly adopted value of 0.99. It decreases linearly to 0.05 over 40,000 training timeslots and remains unchanged afterward.

As the interaction progresses, the experiences attained over time, represented as $e_{t'} = (s_{t'}, a_{t'}, r_{t'}, s_{t'+1})$, are streamlined as a replay buffer, denoted as $\mathcal{D} = \{e_1, \dots, e_{t'}, \dots, e_R\}$ (Lin, 1992). The buffer $\mathcal{D}$ retains the most recent R experiences, from which DQN will learn. In this study, we set R to 40,000.

DQN progressively improves its Q-value estimation by learning from accumulated experiences, which is parallel to ongoing interaction with the environment. We adopted the Double-Deep Q-Network (DDQN) algorithm proposed by Van Hasselt et al. (2016) to train our designed DQN. The DQN is improved (i.e., the weights are tuned) by minimizing the error between the predicted Q-value of the selected action and a target Q-value updated based on the true reward when performing this action. Specifically, in each timeslot t, we uniformly retrieve a batch of experiences at random according to a pre-determined batch size (set to 128 in this study). These experiences are used to update the weights θ_t of the primary network by minimizing the mean-squared error (MSE) and performing backpropagation. This error is expressed as $L_t(\theta_t)$ in Eq. (16):

$$L_t(\theta_t) = \mathbb{E}_{s_{t'},a_{t'}}\left[\left(y_{t'}^{DDQN} - Q(s_{t'},a_{t'};\theta_t)\right)^2\right] \quad (16)$$

In this equation, $y_{t'}^{DDQN} = r_{t'} + \gamma Q(s_{t'+1}, \underset{a_{t'+1}}{max}\, Q(s_{t'+1},a_{t'+1};\theta_t);\theta_i^-)$ represents the target Q-value for the selected action in the sampled timeslot t' while $Q(s_{t'},a_{t'};\theta_t)$ is the predicted Q-values for the selected action based on the state $s_{t'}$ by using primary network (Mnih et al., 2015). The target Q-value is calculated as the sum of the true reward ($r_{t'}$) received for the selected action $a_{t'}$ and the discounted Q-values in the subsequent state $s_{t'+1}$. Decoupling the action selection and estimation by involving two DQN networks has been proven to mitigate DQN's overestimation bias in large-scale sequential decision-making scenarios (Mnih et al., 2015). The target network's parameters θ^- stay fixed and are periodically updated from copying the primary network's parameters θ.

4.2.4. Bike share system simulator

We designed a BSS simulator with which the *DeepBike* agent could interact and collect transition experiences to improve its DQN. The simulator models the dynamics of the bike renting and return activities and bike rebalancing operation.

Fig. 4 illustrates the simulation process, adhering to constraints from Section 3. We set every 24 hours as one episode (i.e., a simulation day/a planning horizon[4]) and split each episode evenly into $|T| = 24$ timeslots with $\Delta t = 1$ hour. The simulator is re-initialized when a new simulation day starts. Adopting the Divvy initialization from Sun (2021), the number of (homogenous) rebalancing vehicles in the fleet was set to be $|M| = 16$ while each rebalancing vehicle has the capacity $C_m^{vehicle} = 20$ and departs from the depot at the beginning of the working shift. Furthermore, the inventory of both stations and rebalancing vehicles are set to be half full to simplify each re-initialization. Following the sequential decision-making scheme, we assumed that the simulator always performs the rebalancing decision first and then executes the bike usage activities within a timeslot. By executing the decision ($z_{i,j,m,t}$, $y_{j,m,t}$), the rebalancing vehicle m routes to the target station j by the shortest road network path and changes station's inventory level by moving bikes in or out. The rebalancing vehicle remains at the station until receiving the next timeslot's rebalancing decision. The execution time of rebalancing was omitted in the simulation, considering the time interval Δt is generally long enough to perform the rebalancing for workers, as adopted the same approach from Lowalekar et al. (2017) and Schuijbroek et al. (2017). The bike usage activities (renting and return) were simulated based on Divvy's historical data described in Section 4.1. The rent/return demands are regarded as fulfilled only if the originally targeted stations have sufficient available bikes/docks. In the event that customers fail to return bikes to the intended stations, it is assumed that they will find alternate docks from the nearest feasible neighbor stations that are able to accommodate their trips. If they fail to rent bikes at the intended stations, customers will exit the system unserved (i.e. using other alternative modes instead). This ensures that the number of bikes is conserved in the system (Li et al., 2018; Lowalekar et al., 2017).

[4] Terminologies across domains often overlap in meaning and are interchangeably used in this study. "step" is more commonly used in RL whereas "timeslot" is used in the VRP. Similarly, "planning horizon" is a common term in VRP while "episode" is frequently used in RL.

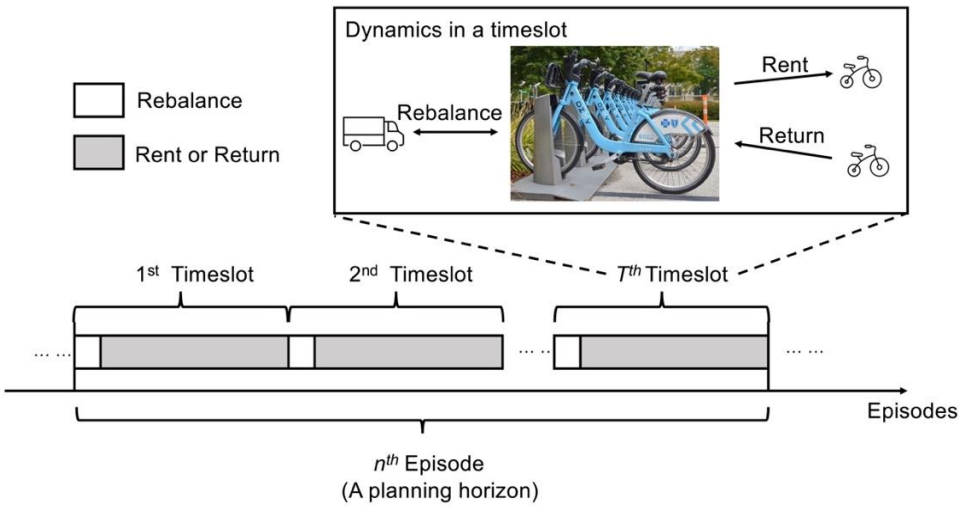

Fig. 4. The bike share system simulator. Each episode consists of $|T|$ timeslot. In each timeslot, the rebalancing actions are executed first and then the customers rent or return the bikes.

4.3. Baseline algorithms

To evaluate the performance of the proposed *DeepBike* model, we compared it against four *DBSRP* baseline algorithms. The baselines are selected because they are representative algorithms derived from either MIP or heuristics discussed in Section 2.

- *No Rebalancing* -- The operator does not execute any rebalancing in the BSS, which establishes a base scenario of BSS operation without rebalancing the bike fleet.
- *Greedy Rebalancing* -- As outlined in (Duan et al., 2018; Li et al., 2018; C. Zhang et al., 2022), in this algorithm, for a more than half-full rebalancing vehicle, it always travels to the nearest less than half-full stations and offloads the most possible bikes there until full. For a less than half-full rebalancing vehicle, it always travels to the nearest more than half-full stations and collects as many bikes as this rebalancing vehicle could.
- *Lagrangian Dual Decomposition-based MIP (LDD-MIP)* -- (Ghosh et al., 2017) exploited the Lagrangian Dual Decomposition and abstraction mechanism to solve the large-scale MIP model formulated in Section 3. To align with the action space of the *DeepBike* model defined in Section 4.2.3 for a fair comparison, Eq. (5) in Section 3 was modified as Eq. (17) such that rebalancing vehicles were constrained to route to reachable stations:

$$\sum_{j \in B_i} z_{i,j,m,t} - \sum_{j \in B_i} z_{j,i,m,t-1} = 0 \qquad \forall i \in N, m \in M, t \in T \tag{17}$$

where B_i is the set of reachable stations via a maximum of 10 horizontal or vertical moves from the current station i by a rebalancing vehicle m in timeslot t.

- *Greedy Online Anticipatory Heuristic (GOAH)* -- (Lowalekar et al., 2017) provided a heuristic approach to produce online solutions for *DBSRP* by computing policy for each individual rebalancing vehicle. Based on looking ahead to three timeslots as well as historically observed demands, this approach estimates the extra bike availability in stations in the current timeslot t. Then, it computes the weight of each valid rebalancing action based on the potential reduction in customer loss if executing this action. Finally, each rebalancing vehicle selects and executes the action that has the maximum weight.

4.4. Evaluation metrics

We used the following three metrics to measure the performance of different rebalancing methods. The metrics were defined in accordance with the terms in the objective function in Eq. (1).

- *Customer loss and customer loss reduction.* *Customer loss* is the total number of customers who failed to rent bikes at their initial departure stations or return bikes at the original arrival stations. *Customer loss reduction* is defined as the delta between customer loss with *No Rebalancing* and customer loss when using a rebalancing strategy.
- *Vehicle routing distance (VRD).* To account for the cost of conducting rebalancing, we use VRD, which is the sum of the distance that a rebalancing vehicle travels among visited stations in an episode. This is also known as vehicle-miles-traveled (VMT).
- *Overall net profit.* The net benefit of rebalancing is then measured by the overall net profit, which is the sum of the increased ride revenue earned from *customer loss reduction* and the cost (negative revenue) of operating the rebalancing vehicles. In this work, the revenue from each ride is equivalent to the average price for a trip ($3.3/trip) (Divvy, 2021). The operation cost of the rebalancing vehicles is estimated by multiplying the mileage rate (58 cents/mile in 2019, (*Home: Internal Revenue Service*, 2023)) and the routing distance of rebalancing vehicles (in miles).

5. Experiment results

In this section, we first present the training results of our proposed *DeepBike* model (Section 5.1). Then we show the empirical evaluation results of comparing the performance of

the *DeepBike* model with different rebalancing baselines, measured by the defined metrics (Section 5.2), in order to examine the effectiveness of our proposed model for *DBSRP*.

During the training phase, the *DeepBike* model was trained on the Gilbreth community cluster with 256 GB of RAM and two P100 Nvidia Tesla GPUs. The model underwent training across 6,500 episodes, totaling 156,000 training steps. The DQN reached full convergence after 168 hours of physical training time. As described in Section 4.1, three consecutive weeks of data served as training data and the subsequent one-week data was utilized to test the performance of the *DeepBike* model. Furthermore, to alleviate the uncertainty of rebalancing models, we ran the experiments ten times and calculated their average performance as well as the standard deviation.

5.1. Model training results

Fig. 5 presents the training curves of the DQN in the *DeepBike* framework. In Fig. 5(a), the average error of DQN training at each step represents the mean MSE of DQN (as defined in Eq. (16)) over the sampled batch (i.e., 128 experiences). Meanwhile, Fig. 5(b) displays the mean highest Q-value over the batch for the selected action. Both the average loss and the average maximum Q-value converged after around 60,000 steps. This observation implies that the agent has learned a rebalancing policy and is able to well approximate the long-term benefits (i.e., Q-values) of rebalancing actions.

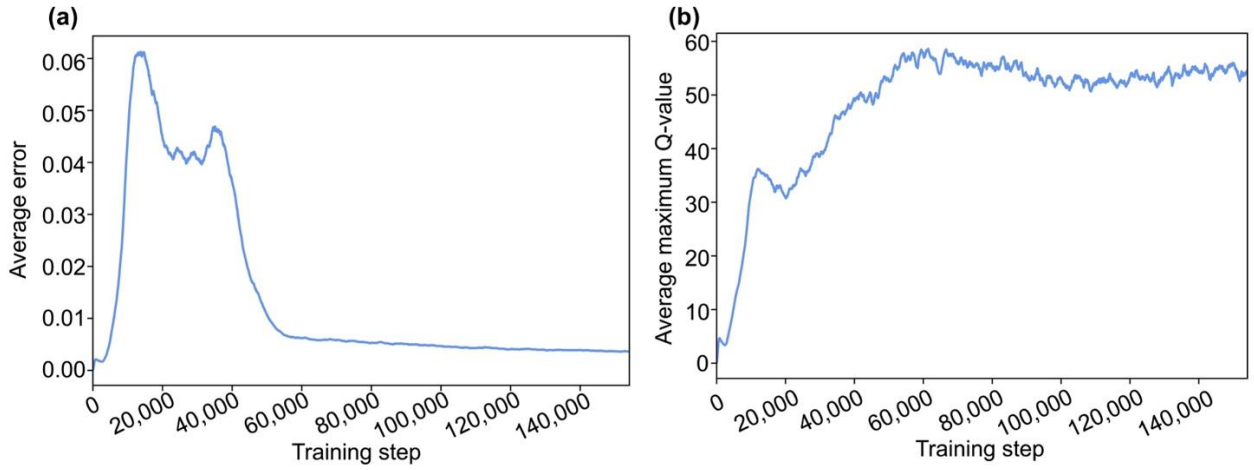

Fig. 5. Training curves of the *DeepBike* model. (a) Average error over the training steps; (b) Average maximum Q-value over the training steps.

5.2. Performance results and analysis

5.2.1. Customer loss reduction

We first present the customer loss under different rebalancing strategies, reflecting how the rebalancing strategy promotes the success of bike rentals and returns for potential customers. Fig. 6(a) illustrates that without rebalancing, the majority of daily customer losses happened in downtown Chicago, while notable customer losses can be observed in suburban regions as well. This implies the potential locations for rebalancing vehicles to target and subsequently mitigate customer loss.

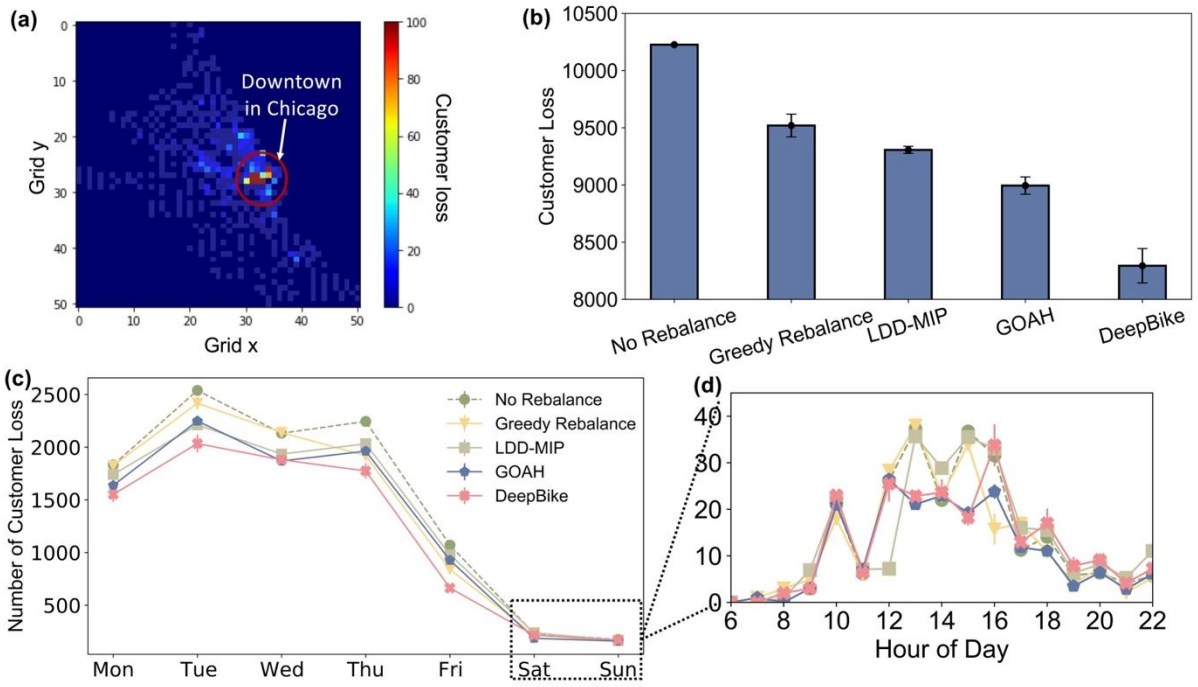

Fig. 6. Comparison of different rebalancing strategies in customer loss. (a) Average daily customer loss distribution across grids in Chicago without rebalancing. (b) Comparison of average customer loss in a week. (c) Comparison of average daily customer loss. (d) Comparison of average hourly customer loss over Saturday and Sunday.

As demonstrated in Fig. 6(b), the *DeepBike* model minimizes total customer loss over the one-week test horizon, outperforming all other strategies. Compared to the *No Rebalance* strategy, *DeepBike* achieves an average reduction of 1,942 customer losses in a week. In contrast, *GOAH*, *LDD-MIP,* and *Greedy Rebalance* yield average weekly reductions of 1,232, 920, and 707 customer losses, respectively. That is, *DeepBike* surpasses *GOAH*, *LDD-MIP*, and *Greedy Rebalance* by reducing additional customer loss by 57.6%, 111.09%, and 174.7%, respectively.

Fig. 6(c) shows that the *DeepBike* model consistently outperforms all other strategies in daily customer loss reduction throughout the weekdays. Both *DeepBike* and *GOAH* outperformed *LDD-MIP* in reducing customer loss due to their adaptability to BSS's real-time customer demands, whereas *LDD-MIP* only produces offline rebalancing solutions. *Greedy Rebalance* underperforms as it neglects upcoming demands and lacks coordination among rebalancing vehicles. However, these rebalancing strategies demonstrate no significant disparities in reducing customer loss during weekends. This can be attributed to the comparatively lower customer demands on weekends as opposed to weekdays. A closer look at the average hourly customer loss distribution on the weekend, depicted in Fig. 6(d), reveals that each rebalancing strategy excels the others only at certain times of the day, exhibiting minor overall gaps in customer loss. The customer loss without rebalancing reaches a maximum of 40 even during peak hours, implying adequate bike inventory at most stations and minimal customer loss due to low usage on weekends. As a result, all rebalancing strategies tend to yield similar rebalancing decisions on weekends, which leads to comparable customer loss reduction. *Vehicle routing distance*

Fig. 7 shows that, from the perspective of VRD, the *DeepBike* model's rebalancing vehicles rank second in routing activity. Its vehicles travel to bike stations more frequently than those of the *Greedy Rebalance* and *LDD-MIP* but not as often as the *GOAH*. Additionally, we show the representative routing trajectories under different strategies in Fig. 8 (Complete trajectories for all rebalancing vehicles are available in the SI from S-2 to S-5).

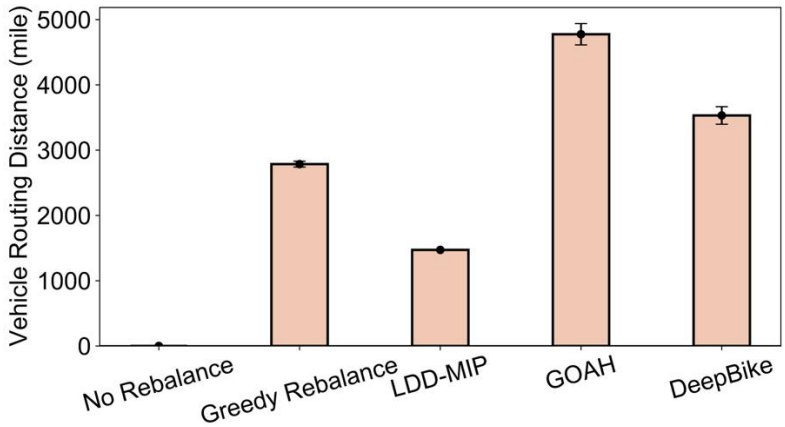

Fig. 7. Comparison of cumulative routing distance (in miles) in a week of different rebalancing strategies.

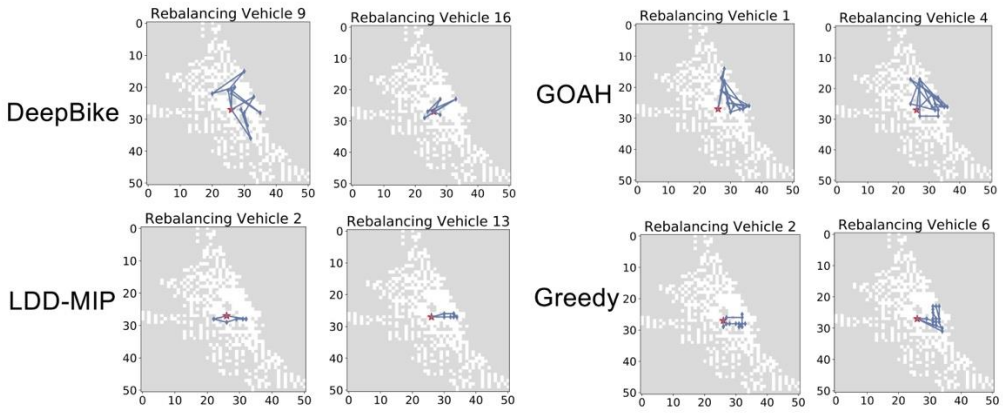

Fig. 8. Comparison of representative trajectories under different rebalancing strategies

The *DeepBike* model proactively dispatches the rebalancing vehicles not only to the downtown area but also to suburban areas to overcome customer loss. Its VRD is 137.57% and 25.45% higher than that of *LDD-MIP* and *Greedy Rebalancing*, respectively. Despite that the rebalancing vehicles are constrained to move at most ten grids either vertically or horizontally in a single timeslot in Section 4.2.3.1, the *DeepBike* model has far sight to maximize this limit. For instance, for rebalancing vehicle #15 in Fig. S-2, it opts for the maximum move in 61% (11 out of 18) timeslots. This strategy enables the vehicles to reach suburban areas over multiple timeslots. Additionally, *DeepBike* learns to coordinate the routings among rebalancing vehicles. This is observed as rebalancing vehicle #16 conservatively circulates around the downtown area contrasting with other rebalancing vehicles that are more aggressive in routing. It suggests that the *DeepBike* model has a far-sighted strategy to balance the customer loss reduction and fuel cost efficiency for the rebalancing fleet to reach long-term optimal rebalancing performance as well as cover a wider range of bike stations in the service region. Contrarily, the routing distances under the *LDD-MIP* strategy are the shortest over other strategies and rebalancing mainly concentrates on the downtown area as indicated in Fig. 7 and Fig. 8, respectively. Even though *GOAH* employs a more active rebalancing strategy as depicted in Fig. 8, it falls short of achieving a higher reduction in customer loss, as discussed earlier. Such a result of the *GOAH* echoes the limitation addressed in (Lowalekar et al., 2017) that their model did not specifically consider the long-term fuel cost in evaluating the marginal benefits of the rebalancing actions in the algorithm.

5.2.2. *Overall net profit*

Last, to comprehensively assess both the benefit and cost of rebalancing, we compare the overall net profits achieved under different rebalancing strategies shown in Fig. 9. The results show that the proposed *DeepBike* model attained the highest overall net profits compared with all other evaluated rebalancing strategies. The overall net profits represent the tradeoff between reducing customer loss and incurring fuel costs. Despite the longer routing distances from aggressive rebalancing to service both downtown and suburban areas, *DeepBike*'s strategy markedly diminishes potential customer loss and maximizes profits. This implies that incurring higher routing costs can be justified by the greater profit returns achieved through better customer demand fulfillment.

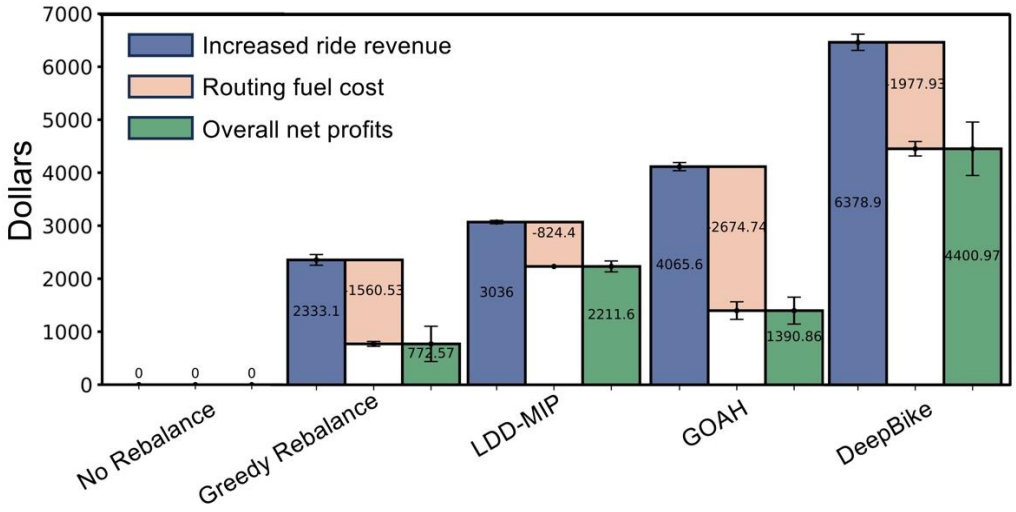

Fig. 9. Comparison of overall net profits in a week with the breakdown of increased ride revenue and routing fuel cost of different rebalancing strategies.

To sum up, our proposed *DeepBike* rebalancing strategy effectively addresses the large-scale *DBSRP*. It generates specific rebalancing decisions for each individual rebalancing vehicle based on real-time system states. Our experiments show that, by trading off between the routing distance and customer loss reduction, rebalancing decisions generated by the *DeepBike* model are beneficial for customers to improve the success of using bikes and for operators to boost the overall net profits.

6. Discussions and conclusion

In this study, we proposed a deep reinforcement learning based model—*DeepBike*—for large-scale *DBSRP*. In *DeepBike*, we designed a neural network (i.e., the DQN) to process the

real-time system states and estimate the long-term benefits (Q-values) of the rebalancing actions for each rebalancing vehicle. *DeepBike* updates its DQN parameters through collected experiences from the interaction with the BSS environment (i.e., learning from trial and error), such that DQN can accurately approximate the Q-values. The numerical results presented in Section 5 demonstrate that *DeepBike* mitigates the most customer loss and maximizes the overall net profit over the one-week test horizon in a large-scale BSS with over 500 bike stations and 16 rebalancing vehicles. We also observed that the *DeepBike* model tends to balance routing fuel cost against customer loss to achieve the highest overall net profits. The resulting customer loss reduction and overall net profit improvement are beneficiaries for both customers and BSS operators.

Our proposed model, *DeepBike*, provides an advancement over existing solutions to *DBSRP*. The complexity of *DBSRP* escalates with the expansion of BSS, marked by more bike stations and a larger fleet size of rebalancing vehicles. In online decision making, only historically observed data is available. By employing a DRL scheme, *DeepBike* trains a neural network that, once converged, can generate online rebalancing decisions for large-scale BSS, contrasting with the MIP approaches that require the complete data for the planning horizon and solve the model from scratch. Our model does not necessitate partitioning the service region; instead, it is able to take the entire region as input for the DQN. Additionally, the output of the neural network approximates the long-term benefits of the rebalancing actions till the end of the planning horizon, rather than myopically making the decisions. The model creates individualized rebalancing actions for each rebalancing vehicle, therefore it is able to adapt to changes in fleet size. It is also notable that, even though our *DeepBike* model was experimented on a station-based BSS because of 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.

Although the *DeepBike* model contributes to building a more efficient and profitable BSS that benefits both customers and operators, it has the following limitations that future works could address to further improve the model. First, the model relies on publicly available trip data from BSS operators, representing observed demands. Such data underestimates station check-in rates as it overlooks potential customers deterred by the unavailability of bikes at empty stations. A true demand estimation model would better represent the latent customer demands and

therefore refine the simulator. Second, all rebalancing vehicles share the same DQN to generate rebalancing decisions, leading to a uniform rebalancing policy across all vehicles. However, adopting a multi-agent decision-making approach, where each rebalancing vehicle operates on distinct policies, could further foster cooperation among vehicles, potentially better maximizing overall revenue. Third, this work neglected the labor cost in the objective, presuming unconditional completion of rebalancing assignments by workers alone. Building a model that accounts for not only the revenues from customers but also the varying cost schemes of conducting rebalancing tasks (such as hiring workers to fulfill all rebalancing needs versus incentivizing customers to ride bikes to a designated location) could lead to a more operationally realistic and economical representation of the BSS.

Appendix

Table A-1. List of notations.

Decision Variables	
$z_{i,j,m,t}$	Binary, the routing decision variable that is set to 1 for vehicle m in the timeslot t if traveling from station i to station j
$y_{j,m,t}$	Integer, the repositioning decision variable for vehicle m. The positive/negative denotes the number of bikes to drop off/pick up when visiting target station j in timeslot t
Intermediate Variables	
$S_{i,t}^{Bike}$	The number of bikes in the station i at the beginning of the timeslot t
$S_{i,t}^{Dock}$	The number of docks in the station i at the beginning of timeslot t
$V_{m,t}^{Bike}$	The number of bikes in the rebalancing vehicle m at the beginning of the timeslot t
$V_{m,t}^{Slot}$	The number of slots in the rebalancing vehicle m at the beginning of the timeslot t
$V_{m,t}^{Loc}$	The location of vehicle m at the beginning of timeslot t
$D_{i,t}$	The net demands of the station i in the timeslot t. The net demand equals to the value that the return demands minus the rent demands
$D_{i,t:t+n}^{pred}$	The predicted future net demands of the station i in the next n timeslots from the beginning of the timeslot t
$CL_{i,t}$	The customer loss occurred in the station i in timeslot t.
Deep Reinforcement Learning Components	
s_t	BSS environment state in timeslot t
r_t	The reward earned in timeslot t
e_t	A sample consists of S_t, a_t, r_t and S_{t+1}, which records the transition from the timeslot t to the timeslot $t+1$
a_t	Rebalancing action in timeslot t, a tuple consists of the routing decision and repositioning decision $a_t = (z_{i,j,m,t}, y_{j,m,t})$
Sets and Constants	
T	The set of discrete timeslots, $t \in \{1,2,\dots,\lvert T\rvert\}$
M	The set of rebalancing vehicles in the fleet, $m \in \{1,2,\dots,\lvert M\rvert\}$
N	The set of aggregated bike stations in BSS, $i,j \in \{1,2,\dots,\lvert N\rvert\}$

$C_m^{vehicle}$	The capacity of vehicle m – the maximum number of bikes that can be carried in the vehicle m, $C_m^{vehicle} = v_{m,t}^{Bike} + v_{m,t}^{Slot}$
α	The average price of a single trip riding the shared bike (dollar/trip)
β	The mileage rate of rebalancing vehicles (dollar/mile)
Δt	The duration of each timeslot
ϵ	The probability of selecting a random action during the training process of *DeepBike*
γ	Time discount factor
ρ	The probability of doing no rebalancing action during the initial training process

References

Brinkmann, J., Ulmer, M. W., & Mattfeld, D. C. (2020). The multi-vehicle stochastic-dynamic inventory routing problem for bike sharing systems. *Business Research*, *13*(1), 69–92. https://doi.org/10.1007/s40685-019-0100-z

Chen, D., & Sakai, K. (2022). A User-Based Bike Return Algorithm for Docked Bike Sharing Systems. *Workshop Proceedings of the 51st International Conference on Parallel Processing*, 1–8. https://doi.org/10.1145/3547276.3548443

Chen, J., Yang, Z., Shu, Y., & Cheng, P. (2020). Rebalance Bike-Sharing System With Deep Sequential Learning. *IEEE Intelligent Transportation Systems Magazine*, *January 2020*, 2–8. https://doi.org/10.1109/MITS.2019.2926252

Chiariotti, F., Pielli, C., Zanella, A., & Zorzi, M. (2018). A dynamic approach to rebalancing bike-sharing systems. *Sensors (Switzerland)*, *18*(2), 1–22. https://doi.org/10.3390/s18020512

Citi Bike. (2023). *January 2023 Monthly Report*. https://citibikenyc.com/system-data/operating-reports

Contardo, C., Morency, C., & Rousseau, L.-M. (2012). Balancing a dynamic public bike-sharing system. *Cirrelt*.

de Bruin, T., Kober, J., Tuyls, K., & Babuška, R. (2018). Integrating State Representation Learning Into Deep Reinforcement Learning. *IEEE Robotics and Automation Letters*, *3*(3), 1394–1401. https://doi.org/10.1109/LRA.2018.2800101

Divvy. (2021). *Single Ride*. https://www.divvybikes.com/pricing/single-ride

Divvy System Data. (2023, May 20). Divvy Bikes. https://ride.divvybikes.com/system-data

Duan, Y., Wu, J., & Zheng, H. (2018). A Greedy Approach for Vehicle Routing When Rebalancing Bike Sharing Systems. *2018 IEEE Global Communications Conference (GLOBECOM)*, 1–7. https://doi.org/10.1109/GLOCOM.2018.8647755

Fishman, E. (2014). *Bikeshare: Barriers, facilitators and impacts on car use*. Queensland University of Technology.

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

General Bikeshare Feed Specification. (2023, July 22). GitHub. https://github.com/MobilityData/gbfs

Ghosh, S., Trick, M., & Varakantham, P. (2016). *Robust repositioning to counter unpredictable demand in bike sharing systems*.

Ghosh, S., Varakantham, P., Adulyasak, Y., & Jaillet, P. (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*, 230–231.

Ghosh, S., Varakantham, P., Adulyasak, Y., & Jaillet, P. (2017). Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research*, *58*, 387–430. https://doi.org/10.1613/jair.5308

Gleditsch, M. D., Hagen, K., Andersson, H., Bakker, S. J., & Fagerholt, K. (2022). A column generation heuristic for the dynamic bicycle rebalancing problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2022.07.004

Home: Internal Revenue Service. (2023, May 20). Internal Revenue Service | An Official Website of the United States Government. https://www.irs.gov/

Hu, R., Zhang, Z., Ma, X., & Jin, Y. (2021). Dynamic Rebalancing Optimization for Bike-Sharing System Using Priority-Based MOEA/D Algorithm. *IEEE Access*, *9*, 27067–27084. https://doi.org/10.1109/ACCESS.2021.3058013

Kloimüllner, C., Papazek, P., Hu, B., & Raidl, G. R. (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), 73–84. https://doi.org/10.1007/978-3-662-44320-0_7

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

Lin, L.-J. (1992). *Reinforcement learning for robots using neural networks*. Carnegie Mellon University.

Lowalekar, M., Varakantham, P., Ghosh, S., Jena, S. D., & Jaillet, P. (2017). Online repositioning in bike sharing systems. *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, 200–208.

Luo, H., Zhao, F., Chen, W. Q., & Cai, H. (2020). Optimizing bike sharing systems from the life cycle greenhouse gas emissions perspective. *Transportation Research Part C: Emerging Technologies*, *117*(September 2019), 102705. https://doi.org/10.1016/j.trc.2020.102705

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*. 1–9.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Oda, T., & Joe-wong, C. (2018). MOVI : A Model-Free Approach to Dynamic Fleet Management. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2708–2716.

O'Mahony, E., & Shmoys, D. B. (2015). Data analysis and optimization for (Citi)bike sharing. *Proceedings of the National Conference on Artificial Intelligence*, *1*, 687–694.

Osorio, J., Lei, C., & Ouyang, Y. (2021). Optimal rebalancing and on-board charging of shared electric scooters. *Transportation Research Part B: Methodological*, *147*, 197–219. https://doi.org/10.1016/j.trb.2021.03.009

Qin, Z., Tang, J., & Ye, J. (2019). Deep reinforcement learning with applications in transportation. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3201–3202.

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

Schuijbroek, J., Hampshire, R. C., & van Hoeve, W. J. (2017). Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, *257*(3), 992–1004. https://doi.org/10.1016/j.ejor.2016.08.029

Shaheen, S., Guzman, S., & Zhang, H. (2010). Bikesharing in Europe, the Americas, and Asia. *Transportation Research Record*, *2143*, 159–167. https://doi.org/10.3141/2143-20

Shu, J., Chou, M. C., Liu, Q., Teo, C. P., & Wang, I. L. (2013). Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Operations Research*, *61*(6), 1346–1359. https://doi.org/10.1287/opre.2013.1215

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

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

Silver, D. (2014). Lecture 9: Exploration and Exploitation. *Computer Science Department, University of London*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. https://doi.org/10.1038/nature16961

Sun, R. (2021). *Bike Share System-Rebalancing Estimation and System Optimization* [PhD Thesis]. Purdue University Graduate School.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Vallez, C. M., Castro, M., & Contreras, D. (2021). Challenges and opportunities in dock-based bike-sharing rebalancing: A systematic review. *Sustainability*, *13*(4), 1829.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-Learning. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2094–2100. https://doi.org/10.1609/aaai.v30i1.10295

Wang, J., & Sun, L. (2020). Dynamic holding control to avoid bus bunching: A multi-agent deep reinforcement learning framework. *Transportation Research Part C: Emerging Technologies*, *116*(April), 102661. https://doi.org/10.1016/j.trc.2020.102661

Wang, T. (2014). *Solving Dynamic Repositioning Problem for Bicycle Sharing Systems: Model, Heuristics, and Decomposition*.

Yin, Z., Hardaway, K., Feng, Y., Kou, Z., & Cai, H. (2023). Understanding the demand predictability of bike share systems: A station-level analysis. *Frontiers of Engineering Management*, *10*(4), 551–565. https://doi.org/10.1007/s42524-023-0279-8

Yin, Z., Kou, Z., & Cai, H. (2023). A Deep Reinforcement Learning Model for Large-Scale Dynamic Bike Share Rebalancing with Spatial-Temporal Context. *The 12th International Workshop on Urban Computing*. http://urban-computing.com/urbcomp2023/file/UrbComp2023_paper_7.pdf

Zhang, C., Wu, F., Wang, H., Tang, B., Fan, W., & Liu, Y. (2022). A Meta-Learning Algorithm for Rebalancing the Bike-Sharing System in IoT Smart City. *IEEE Internet of Things Journal*, *9*(21), 21073–21085. https://doi.org/10.1109/JIOT.2022.3176145

Zhang, D., Yu, C., Desai, J., Lau, H. Y. K., & Srivathsan, S. (2017). A time-space network flow approach to dynamic repositioning in bicycle sharing systems. *Transportation Research Part B: Methodological*, *103*, 188–207. https://doi.org/10.1016/j.trb.2016.12.006

Zhao, J., Mao, M., & Zhao, X. (2021). A hybrid of deep reinforcement learning and local search for the vehicle routing problems. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/9141401/?casa_token=joKmbnG-H1kAAAAA:XSSUCwt_NeOCg0twxOJeS3yl3uwiz1o5te-NkOxMisQunkE8_zq3ordTklZNona4NOzpCrE12kA

Zheng, X., Tang, M., Liu, Y., Xian, Z., & Zhuo, H. H. (2021). Repositioning bikes with carrier vehicles and bike trailers in bike sharing systems. *Applied Sciences (Switzerland)*, *11*(16). https://doi.org/10.3390/app11167227

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

Zhu, H., Shou, T., Guo, R., Jiang, Z., Wang, Z., Wang, Z., Yu, Z., Zhang, W., Wang, C., & Chen, L. (2022). Redpacketbike: A graph-based demand modeling and crowd-driven station rebalancing framework for bike sharing systems. *IEEE Transactions on Mobile Computing*. https://ieeexplore.ieee.org/abstract/document/9693278/

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- DeepBikeSIfinal.docx