# DYNAMIC BIKE SHARING REBALANCING: A HYBRID FRAMEWORK BASED ON DEEP REINFORCEMENT LEARNING AND MIXED INTEGER PROGRAMMING

by

**Zhuoli Yin**

**A Thesis**

*Submitted to the Faculty of Purdue University*

*In Partial Fulfillment of the Requirements for the degree of*

**Master of Science in Industrial Engineering**

School of Industrial Engineering

West Lafayette, Indiana

August 2021

## THE PURDUE UNIVERSITY GRADUATE SCHOOL
## STATEMENT OF COMMITTEE APPROVAL

**Dr. Hua Cai, Chair**
School of Industrial Engineering

**Dr. Vaneet Aggarwal**
School of Industrial Engineering

**Dr. Satish V. Ukkusuri**
Lyles School of Civil Engineering

**Approved by:**
Dr. Abhijit Deshmukh

*Dedicated to my beloved parents*

# ACKNOWLEDGMENTS

I would like to express my deepest gratitude and respect to Dr. Hua Cai, who has provided firm support to my research, study, and future career. I would express my thanks to Dr. Vaneet Aggarwal and Dr. Satish V. Ukkusuri for giving helpful advice in my research work.

I am also grateful to the colleagues with whom I have worked in the uSMART lab, especially Dr. Zhaoyu Kou, Hao Luo, Zequn Li, and Runhua Sun, for their kind guidance and help. I would thank my friends at Purdue University and in China, who have provided (remote) accompanies.

I cannot think about anyone but my beloved parents, Weifang Yin and Qunying Cheng, to express my wholehearted gratitude for their consistent supports in my 24 years of life.

During my master's study, COVID-19 pandemic changed everyone's daily life. The study and work have to be remote and online. Even though, we, human, have suffered from the pandemic severely, we always stand together firmly to fight against it. People who fight for the pandemic will always be remembered.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

Bike sharing systems (BSSs), as an emerging mobility mode, have been widely deployed in major cities around the world and viewed as an environment-friendly transportation mode to serve the first/last-mile trips. Because the bike usage demands are spatially and temporally unbalanced, bike stations may become empty or full at different times and in different regions, which will cause inconvenience to customers to rent or return bikes and result in potential customer loss. To establish a BSS that caters to the bike demands and decrease customer loss, the system operator needs to efficiently rebalance the system. Existing studies mainly developed the rebalancing policies based on the mixed integer programming (MIP) algorithms, which provide centralized solutions from the perspective of the entire system. However, as the real-world BSS is often large-scale and with dynamic demands, the rebalancing policy needs to be generated efficiently and be scalable for the real-time BSS operation. This study proposes a hybrid rebalancing policy to improve the efficiency of BSS. A model-free deep reinforcement learning (DRL) framework – DeepBike – is firstly proposed to learn the optimal rebalancing policy for station-based BSS, which makes distributed online decisions for each vehicle without coordinating with the others. Then this study adopts a Mixed Integer Programming algorithm and integrates it with DeepBike to develop the hybrid policy, leveraging the "online" solutions of the DeepBike and the "near-exact" solutions of the MIP. A large-scale real-world BSS simulator is built to compare the performance of different policies based on the historical trip records from Divvy (the BSS in Chicago). The results show that, while keeping the scalability and efficiency, the hybrid policy improve the profit by +29.9% and +39.8% compared to the DeepBike and the MIP respectively. This finding supports the real-world implementation of the DRL-MIP-based hybrid model to produce online rebalancing policy dynamically for real-time rebalancing operation of large-scale BSS with multiple vehicles.

# 1. INTRODUCTION

As an emerging shared mobility mode, BSSs have been deployed in many cities around the world. BSS has been viewed as an environment-friendly transportation mode to serve the first/last-mile trips (DeMaio, 2009). Figure 1.1 shows the worldwide major bike sharing services. In the U.S., bike sharing programs have been launched in many major cities, such as Pronto in Seattle, Metro in Los Angeles, Divvy in Chicago, and Citi in New York (Kou & Cai, 2019). BSSs' convenience to substitute the first- and last-mile commuting trips attracts lots of users (DeMaio, 2009). Additionally, BSSs have the potential to provide environmental-, social-, and transportation-related benefits (Shaheen et al., 2010). For example, from the perspective of environmental benefit, New York's Citi Bike sharing program contributed to 5417 tons of greenhouse gas emission reduction in 2016 (Kou et al., 2020). BSSs mainly have two forms: station-based and dock-less. Station-based BSSs have physical infrastructures with limited number of docks to store bikes. The sites of stations are pre-determined and deployed in a city. In this type of system, customers can pick-up bikes from the installed stations, ride to the target stations and drop-off bikes there. On the other hand, in dock-less systems, customers can pick-up and drop-off bikes from/to almost any places that are within the service region.

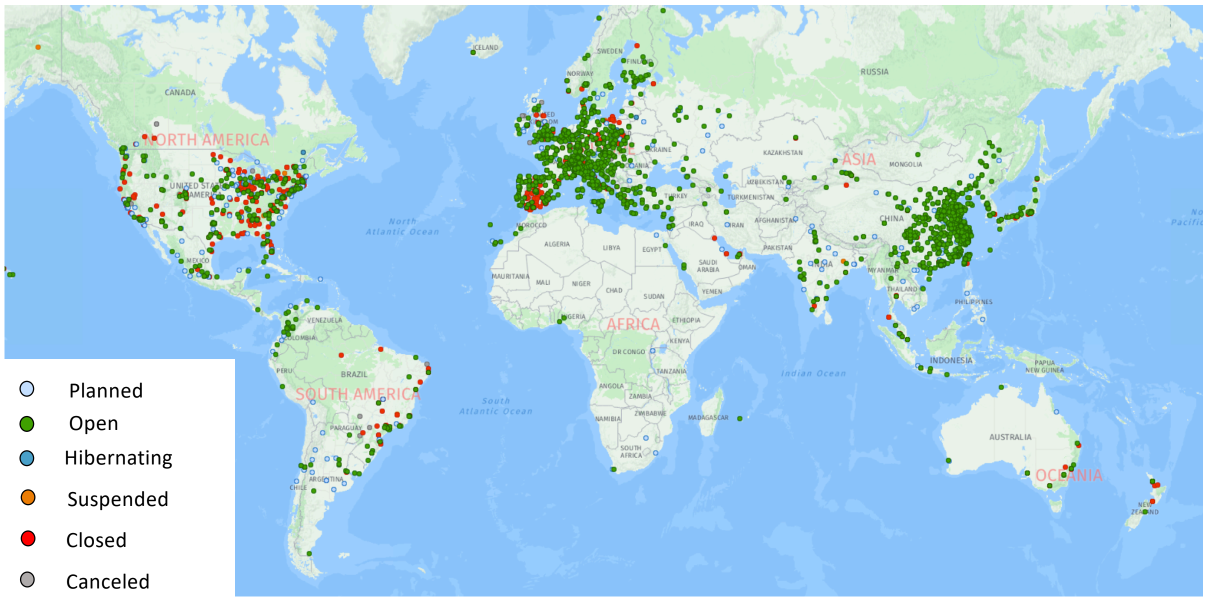


Figure 1.1. The distribution map of bike sharing systems in the world (Meddin & DeMaio, 2020)

The spatiotemporal unbalanced bike usage demands can lead to potential customer loss in BSSs. The rent and return of bikes are pertinent to personal trip purposes, resulting bike usage to occur at different time (temporal) and different places (spatial). For instance, commuters may rent bikes from their home and return bikes to the stations near their work places in the early morning. Because most of the companies locate in the center of a city, these commuting trips add lots of bikes into the stations in the downtown area in the morning (Xing et al., 2020). Figure 1.2 shows an example of such spatial and temporal unbalance. The subfigure (a) presents the net hourly demand patterns of two stations from Monday Sept. 9 to Sunday Sept. 16, 2019. Thesubfigure indicates the specific locations of the two stations. The station located at Millennium Park has highly fluctuated daily net demands compared with the station in Lake Shore Dr & Ohio St., with significant bike inflow and outflow during the morning and evening peak hours, respectively. Such imbalanced bike movements in a BSS often cause the congestion or starvation of bikes in stations (Ghosh et al., 2017). A station is considered as congested when it is (close to) full and has the risks that customers might fail to return bikes, while a station is in starvation when it is (close to) empty and has the risks that customers may not be able to find a bike. As indicated by (Fricker & Gast, 2016), stations in starvation or congestion may cause potential customer loss if the customers fail to rent or return bikes, resulting not only revenue loss and but also increased carbon emissions when customers have to use more energy intensive transportation modes, such as private cars, to make the trips (Ghosh et al., 2017).

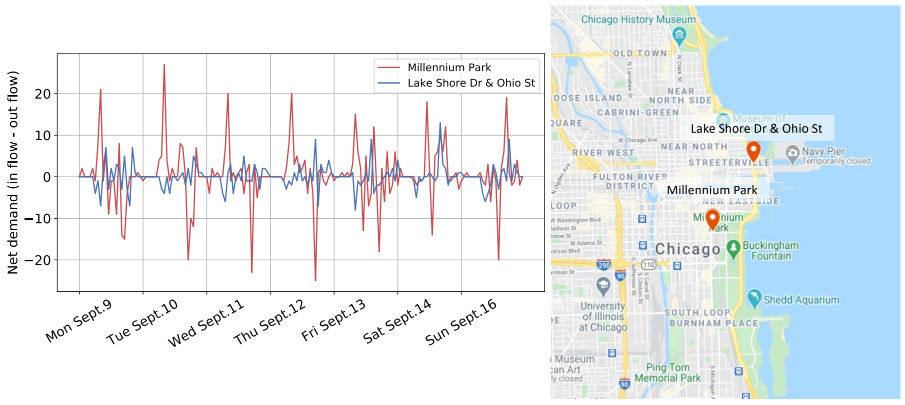


Figure 1.2. Example of unbalanced bike use demands. (a) The hourly net demands (*return demands minus rent demands*) of two different *Divvy* stations. (b) The locations of the two example stations

To avoid such customer loss and establish an efficient and reliable BSS, the system operator needs to rebalance the bikes among different stations. A well-developed rebalancing policy can redistribute the bike and dock resources to locations where they are needed (Chemla et al., 2013). This rebalancing is often conducted using automobiles, such as trucks, vans, and trailers, which travel among the stations and move bikes in or out from/to the stations (O'Mahony & Shmoys, 2015). Specifically, the rebalancing policy involves (1) routing decisions of the vehicles, which choose the sequences of bike stations to be visited, and (2) repositioning decisions of the bikes, which determine the number of bikes to be removed or delivered from/into each visited station. Note that these two decisions are highly associated because the vehicles can only move bikes into or out from the stations that the vehicles physically present.

Bike rebalancing includes two main types: static rebalancing and dynamic rebalancing. Static rebalancing rebalances the stations to the target inventory level during the night, regardless of the movement of users when conducting the rebalancing. Dynamic rebalancing executes rebalancing actions throughout the day, considering the dynamic of bike rentals and returns. Because of its importance to a BSS's efficiency, bike rebalancing has been widely studied in existing literature. Researchers seek to find an (near-) optimal rebalancing policy for large-scale BSS within a reasonable computation time, due to the NP-hard nature of bike rebalancing problem (Ghosh et al., 2017; Schuijbroek et al., 2017). The static rebalancing mainly focuses on two separate steps: (1) determining the target bike stock level of each station and (2) designing routes of vehicles traversing the stations. As the static scenario assumes that the hire and return of the bikes are negligible during rebalancing, the determined service levels of each station are usually the estimated desired level in the following long-time span (Raviv et al., 2013; Schuijbroek et al., 2017). Thus, in order to meet such service level, vehicles move to deliver bikes to the stations aiming to minimize the total movement distance. For large-scale systems, the studies mainly cluster the entire system into smaller regions in order to reduce the complexity and assign one vehicle to serve each of the cluster. Static rebalancing policy provides an efficient way to get rebalancing solutions. However, this rebalancing policy chiefly prepares for the peak hours and is inflexible to adjust the real-time plan for the long-term variant demands. Besides, the clustering algorithm treats the demands in the downtown areas and suburban areas equal, which may fail to cater to the heavy demands of rebalancing in certain regions (needs more than one vehicle to rebalance stations to the desired inventory levels). On the other hand, dynamic rebalancing policy offers a more effective way to redistribute the bikes. It continuously utilizes the current and past system status and carries out rebalancing actions frequently throughout the day. The major studies formulated and solved the problem by mixed integer programming. To obtain the solution for large-scale system, they applied decomposition algorithms to reduce the computation time. These model-based approaches provide good solutions particularly in offline learning scenarios. However, with the increasing of the BSS sizes, the computation time will expand exponentially. Also, it is hard for model-based approach to capture the uncertainties in the system.

To address the aforementioned research gaps, this work proposes a hybrid approach that leverages a model-free DRL framework – DeepBike – and an MIP model to dynamically dispatch a fleet of vehicle to rebalance a BSS. Specifically, in the proposed DeepBike algorithm, an agent interacts with the BSS simulator, with the aim to learn the optimal rebalancing policy and minimize the system's customer loss and routing fuel cost. Additionally, the agent uses Double Deep Q-network (DDQN) to output distributed rebalancing decision for each vehicle based on a shared neural network. In this decentralized manner, the agent is scalable to dispatch different number of vehicles in online decision-making setting. In order to produce highly associated decisions, this study reshapes the output of the neural network into a table in which the row and the column correspond to the routing and repositioning index respectively. Then a MIP algorithm is adapted, which provides more stable and exact rebalancing solutions in DBSRP in the offline setting. To leverage the advantages of both DeepBike and MIP, these two approaches are further integrated as a hybrid policy to improve the overall system performance.

The structure of the rest of the thesis is organized as below. Chapter 2 reviews the state-of-the-art works relating to this research. The definition of the dynamic bike sharing rebalancing problem is described in Chapter 3.  Chapter 4 explains the proposed methods including the data preprocessing, system simulator, rebalancing algorithms (DeepBike and MIP), benchmark algorithms, and evaluation metrics. The evaluation results and sensitive analysis are presented in Chapter 5. Finally, Chapter 6 concludes this work and discusses the limitations and future work.

# 2. LITERATURE REVIEW

Traditional optimization algorithms, such as MIP, have been extensively implemented to solve Bike Sharing Rebalancing Problems (BSRPs) and improve the service level, in both static and dynamic cases. The purpose of Static Bike Sharing Rebalancing Problem (SBSRP) is to prepare initial status of stations for the upcoming day. Dynamic Bike Sharing Rebalancing Problem (DBSRP) is to constantly adjust the inventory of the stations to a certain level for the upcoming demands when customers are actively using bikes. Chapter 2.1 discusses the contributions of existing work in solving these two scenarios and identifies the research gap of needing models to generate online rebalancing decision for real-time BSS operation.

Because reinforcement learning (RL) has offered promising results in solving sequential decision-making problems by modelling the problems as Markov Decision Processes (MDPs), several works have implemented RL methods in the operation of transportation systems (discussed in Chapter 2.2). The existing research has shown good performance of RL in real-time ride sharing, ride pooling, and freight delivery. However, the associated routing and repositioning decisions are a challenge for RL in solving BSRPs. Even though the previous work used RL to solve BSRPs with one vehicle in each service region, deriving a RL-based model that is scalable for different number of vehicles working in the entire system and is adaptable to highly changing BSS is needed.

## 2.1 System rebalancing

In the case of SBSRP, most of existing papers are based on optimization algorithms, such as MIP. Chemla et al. (2013) proposed an exact model of SBSRP with only one capacitated vehicle in the system and solved the problem based on a branch-and-cut algorithm. The effectiveness of the algorithm was proven on the set of instances with 100 stations. Owing to the NP-hard nature of BSRP, however, the problem is more complex when incorporating multi-vehicles in a large BSS. To solve SBSRP with multi-vehicles, (Raviv et al., 2013) and (Dell'Amico et al., 2014) presented different MIP formulations based on different assumptions and offered feasible solutions. Their approaches pre-fixed the desired service levels of the stations and focused on the routing problems. Schuijbroek et al. (2017) handled both key aspects for SBSRP together: determining the target inventory bounds of the stations with stochastic demands and solving multi-vehicles routes. They proposed a cluster-first route-second heuristic algorithm, so that each vehicle finds the routing solution separately in a cluster. Similarly, O'Mahony & Shmoys (2015) computed the lower bounds of the demands for bikes in the stations using a data-driven way and allocated the bike resources among stations to ensure the desired fill levels. Then they solved the overnight and mid-rush rebalancing problems in the static manner for multi-vehicles.

Though the static approaches prepare the desired inventory level of stations for the next few hours, its contribution to the future demand will fade out and the stations will become imbalanced during the day. Hence, it is natural to investigate the DBSRP policy for the purpose of improving the satisfaction of customers, i.e., to have the right bikes in the right stations at the right time (Lowalekar et al., 2017). This study also focuses on the DBSRP as it provides flexibility for operator to rebalance the system. Contardo et al. (2012) proposed a mathematical formulation of DSBRP during peak hours and derived the lower and upper bounds via Dantzig-Wolfe decomposition and Benders decomposition. However, there still exists a large gap between the lower and upper bounds. Ghosh et al. (2017) handled DSBRP in an offline method by considering the conjunction of vehicles' routing and multi-step future expected demands, with the aim to maximize the overall profit. To reduce the computation time, they first aggregated the all individual stations into different clusters viewing each cluster as an abstract station, solved the rebalancing problems of abstract stations by Lagrangian decomposition, and then extracted the solutions for the original DSBRP. While it is always hard to get a high-quality real-time solution for DSBRP in the large-scale system within short computation time, Lowalekar et al. (2017) provided an online approach to solve the DSBRP by MIP. They used Lagrangian dual decomposition and Greedy online anticipatory heuristic method to speed up the solution generation. However, their model did not incorporate the cost of rebalancing (e.g., fuel cost) in the objective function and the model neglected the actual travel time so that the movement from one station to another is always achievable. As discussed, the key question to be answered in DSBRP is to determine which station should the vehicle head for priorly, because the vehicle resources are limited and vehicles cannot reach all stations within the given time window. To obtain such prioritization rules, Legros (2019) proved the existence of the optimal inventory level which varies at different time step at each station via a dynamic programming approach. To choose the next target station, the paper determines the prioritization of stations by an one-step policy improvement method based on the relative value function of the system. Chiariotti et al., (2018) developed a model to periodically scan the BSS and employ the rebalancing decisions to prevent the stations from becoming completely full or empty. Each station can be reached from other stations.

Most of the work mentioned above aims to minimize the lost demand or maximize the overall revenue with the fixed bike fleet size and rebalancing frequency. Shu et al. (2013) observed the needs of using bikes to replace the short trips by estimating the user trip demand as a Poisson distribution. Based on the stochastic network flow model and linear programming model, they examined the effects in the utilization of bikes from the view of deployment and rebalancing of bikes, together with the number of docks installed in the stations. Luo et al. (2019) investigated the issue of bike fleet size and rebalancing frequency in dock-less BSS from the perspective of life cycle greenhouse gas (GHG) emission rate. With the goal of minimizing GHG emissions, they studied the tradeoff between reducing the number of bikes and increasing the frequency of rebalancing, which could provide operation insights for system rebalancing problems.

The existing model-based approaches aforementioned provide the (near-) exact solutions for BSS rebalancing, but may not have the flexibility to capture the uncertainties when executed online in real-time BSS operation. The policies for SBSRBP only redistribute the bikes in few time windows and cannot improve the BSS in a long period. On the other hand, the policies for DBSRBP need long computation time to get a high-quality solution. But in online settings with large number of stations and vehicles, the time limit for getting the solution is short so the operator may not be able to get a high quality solution (Lowalekar et al., 2017). Also, the optimization algorithms derive the solutions from the perspective of the entire system, so that each vehicle is coordinated with other vehicles when executing the rebalancing decisions. This makes the algorithms unscalable to produce effective solutions for different size of vehicle fleet. Thus, how to generate distributed online rebalancing actions for each vehicle while considering the long-term revenue is critical to improve the BSS in a long period.

## 2.2 Reinforcement learning

Model-based approaches, such as MIP, have been proven to provide good solutions and help improve the performance in classical operation research problems, such as inventory control and vehicle routing problems (Golden et al., 1998; Laporte, 1992) which can be viewed as the basis of BSRP. However, the pre-determined constraints in the model-based approaches inherently limit the performance (Kaelbling et al., 1996), especially in highly changing scenarios with the stochastic components such as real-time traffic condition, weather, and events.

RL methods are often regarded as a potential solution to enable an agent to adapt to changes in real time to an unpredictable environment (Degris et al., 2012). The goal of reinforcement learning is to learn good policies for sequential decision-making problems via optimizing a cumulative future reward signal (Sutton & Barto, 2018). In the category of reinforcement learning algorithms, Q-learning is one of the most popular value-based methods (Cristopher John Cornish Hellaby Watkins, n.d.). Ever since Mnih et al. (2013, 2015) combined Q-learning with deep neural network as Deep Reinforcement Learning (DRL), which was kick-started by the Deep Q-Networks algorithm (DQN, Hessel et al., 2018), it provides the flexible function approximation and was tested to reach human-level performance on many games. Further, to overcome the overestimations of DQN and to yield more accurate value estimations, Van Hasselt et al. (2016) built Double DQN algorithm based on Double Q-learning algorithm (Hasselt, 2010) and improved the performance of high dimension decision-making problems.

As DRL has shown promising effectiveness in sequential decision-making problems, it has been widely applied in intelligent transportation systems (ITSs), such as online ride order dispatching on on-demand ride-sharing platforms, route planning, and traffic signal control (Qin et al., 2019). Specifically, DRL provides a model-free approach to find optimal decisions based on the Q-value estimation instead of modeling the system. Q-value (or action-value) is defined as the expected discounted reward for executing an action at given system state and following policy (Christopher J C H Watkins & Dayan, 1992). Besides, the power of deep neural network helps extract the representation of the input tensors and tackle uncertainties in the system. Oda & Joe-Wong (2018) and Al-Abbasi et al. (2019) proposed a DRL-based framework to learn the optimal vehicle dispatching policy in a large-scale ride-sharing system. They used DQN for each single vehicle to produce its own dispatching decision, which could adapt quickly to the real-time spatial and temporal variance in the dynamic environments.  Li et al. (2018) dealt with

DBSRP by a spatial-temporal reinforcement learning framework aiming to minimize the long-term customer loss. To reduce the complexity of the system, they first proposed an interdependent inner-balance clustering algorithm and then designated vehicles to each cluster to rebalancing bikes. Chen et al. (2021) utilized the DRL to solve multi-transfer freight delivery problem. They integrated DRL algorithm with MIP optimizer as a hybrid approach, which shows the superior scalability and improved performance compared with MIP or DRL solution alone.

The existing studies have demonstrated some application examples of DRL in ITSs and provide foundations and motivations to solve DBSRP. Oda & Joe-Wong (2018) and Al-Abbasi et al. (2019) offer the motivation to build a DRL-based framework with convolutional neural network to tackle DBSRP. Since the real-time BSS operation is also quite uncertain and large-scale, the representations of the systems can also be constructed as the inputs of the DQN to capture the relationship between the system states and possible rebalancing decisions. However, since the policy should output highly associated repositioning and routing actions for DBSRP, the action space of DBSRP is more complex than that in the ride sharing systems. Li et al. (2018) has solved the DBSRP by DRL. However, their work only considered minimizing the customer loss without accounting for rebalancing costs. Also, they pre-clustered the service region into some smaller regions and dispatched one vehicle for each cluster. This simplification essentially converted the problem into a single vehicle rebalancing problem and how to generate rebalancing actions for multiple vehicles has not been tackled. Chen et al., (2021) integrates the DRL-based and MIP-based algorithms to further improve the performance of the solution. They selected the policy to be followed in each time step based on the defined metric "efficiency". However, the freight delivery problem in their work corresponds to SBSRP. The DBSRP is more complex because each rebalancing action usually has long-term effect. Also, the real-time BSS cannot provide full historical information to the MIP model to get an online solution. Therefore, for DBSRP, a rebalancing model which leverages the benefits of the DRL-based model and the MIP-based model to dynamically makes decentralized highly associated rebalancing decisions for the BSS with multiple-vehicles is critically needed.

To address this gap, this research aims to develop a hybrid strategy that integrates the DRL-based and MIP-based algorithms. For the DRL-based algorithm, a model-free framework called DeepBike is proposed, which learns to dynamically deploy vehicle fleet to rebalance BSS for multiple steps and minimize the defined objectives. By using neural network, each vehicle makes its own decisions without coordinating with other vehicles, so as to ensure the scalability of the algorithm. For the MIP-based algorithm, an MIP formulation for DBSRP is adapted from (Ghosh et al., 2017), which considers demands of multiple steps and applied Lagrangian dual decomposition and abstraction mechanisms to reduce complexity. Then it is integrated with the DeepBike model as a hybrid approach to reach a better performance of the system. A large-scale BSS simulator is constructed based on the real-world bike sharing trip data in Divvy, Chicago to verify the effectiveness of the proposed DRL-MIP-based hybrid approach in solving DBSRP compared with benchmark rebalancing algorithms. In the context of the existing literatures, the main contribution of this study is to propose a hybrid rebalancing strategy that could improve the profit of the BSS for multiple steps, by generating online rebalancing decisions for the real-time changing system with multiple vehicles.

# 3. PROBLEM DEFINITION

This Chapter formally introduces the definition of DBSRP. The details of notations used throughout this study are listed in Table 3.1.

The objective of the DBSRP is to maximize the net revenues that can be earned through rebalancing the system by dispatching vehicles to specific stations and drop-off/pick-up a certain number of bikes at the assigned stations to reduce customer loss while minimizing rebalancing costs. Each reduced customer loss contributes to the revenue of the bike sharing company while the vehicles consume fuel and incur cost when routing to the target stations. Therefore, instead of purely minimizing the customer loss, this work aims to maximize the profits by weighing the tradeoffs between minimizing the customer loss and minimizing the routing fuel cost.

Table 3.1. The notations used in the problem definition and formulation.

| Variables | Description |
|---|---|
| $s_t$ | The states of the station at the beginning of time slot $t$, where for every time slot $t$, the state of each station $s_{s,t} = \left(s_{s,t}^{Bike}, s_{s,t}^{Dock}\right) \in s_t$. . |
| $s_{s,t}^{Bike}$ | The number of bikes at station $s$ at the beginning of time slot $t$ |
| $s_{s,t}^{Dock}$ | The number of docks at station $s$ at the beginning of time slot $t$ |
| $v_t$ | The states of the vehicles at the beginning of time slot $t$, where for every time slot $t$, each vehicle $v_{v,t} = (v_{v,t}^{Loc}, v_{v,t}^{Bike}, v_{v,t}^{Slot}) \in v_t$. |
| $v_{v,t}^{Bike}$ | The number of bikes in vehicle $v$ at the beginning of time slot $t$ |
| $v_{v,t}^{Slot}$ | The number of slots that could store bikes in vehicle $v$ at the beginning of time slot $t$ |
| $v_{v,t}^{Loc}$ | The location of vehicle $v$ at the beginning of time slot $t$ |
| $F_{t:t+T}$ | The expected future demands in the next time slots $t+1, \dots, t+T$ |
| $\pi$ | The rebalancing policy |
| $F_t$ | The net demands of the stations in the time slot $t$. The net demand equals to the return demand minus the rent demand. |
| $S_t$ | The external observation of the BSS at the beginning of time slot $t$. |
| $a_t$ | The action taken in time slot $t$, which includes both repositioning and routing. |
| $r_t$ | The reward earned in time slot $t$ |
| $O_t$ | 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 $t+1$ |
| $z_{s,s',v}^{t}$ | The routing decision variable for vehicle $v$ in time slot $t$ that decides to travel from station $s$ to station $s'$ |
| $y_{s',v}^{t}$ | The repositioning decision variable, denotes that number of bikes to pick-up/drop-off for vehicle $v$ that visits station $s'$ in time slot $t$ |
| $M$ | The number of vehicles in the rebalancing fleet |
| $N$ | The number of bike stations in the system |

Table 3.1 continued

| | |
|---|---|
| $T$ | The total number of time slots in each episode |
| $\Delta t$ | The length of each time slot |
| $\gamma$ | Time discount factor |
| $C_s^{station}$ | The capacity of the station $s$ |
| $C_v^{vehicle}$ | The capacity of the vehicle $v$ |
| $P_{s,s'}$ | The estimated distance between station $s$ and station $s'$ |
| $\beta$ | The discounted mileage rate (cents per mile) |
| $\epsilon$ | The probability of selecting a random action during the training process of DeepBike |
| $\rho$ | The probability of selecting the MIP policy in the proposed hybrid strategy |

The dynamics of the BSS are constrained by the variables that describes the system states and rebalancing decisions. In a BSS, we have $N$ bike stations located in different urban regions and $M$ vehicles to be dispatched in the fleet. Specifically, each bike station $s$ and vehicle $v$ has limited physical capacity to store bikes. The capacities of the stations and vehicles are denoted as $C_s^{station}$ and $C_v^{vehicle}$, respectively. Each day is viewed as an episode and each episode is evenly divided into $T$ time slots with time length $\Delta t$. The current time slot is indexed as $t.$ During time slot $t-1,$ bikes are picked up or dropped off by users at station $s$, and the net demand at the station $s$ is defined as $F_{s,t-1} = return\ demand - rent\ demand$. Likewise, the future demands of stations in the next 1 time slot are defined as $F_{t:t+1} = (F_{1,t:t+1}, F_{2,t:t+1}, \dots, F_{N,t:t+1})$ which can be estimated by developing a demand prediction model. After bike renting and returning in the time slot $t-1$, the number of available bikes and docks in each station are therefore updated to $s_{s,t} = \left(s_{s,t}^{Bike}, s_{s,t}^{Dock}\right)$ at the beginning of the next timeslot $t$. The number of bikes in station $s$ (inventory level) should be always less than or equal to the capacity $C_s^{station}$. In time slot $t$, the location of vehicle $v$ is marked as $v_{v,t}^{Loc}$ and the vehicle has available bikes $v_{m,t}^{Bike}$ and available slots $v_{m,t}^{Slot}$ that could store bikes $v_{m,t}^{Slot}$Therefore, the states of each vehicle can be combined as $v_{v,t} = (v_{v,t}^{Loc}, v_{v,t}^{Bike}, v_{v,t}^{Slot})$. The external observation of the BSS is thus $S_t = (s_t, v_t, F_{t:t+T})$.

The rebalancing decisions are made for vehicles based on a specific policy $\pi$. Based on the external observation $S_t$ in each time slot $t$ and the historical observations (if needed), the routing decision $z_{s,s',v}^{t}$ is assigned to vehicle $v$, which dispatches vehicle $v$ from its original station $s$ to a target station $s'$. Meanwhile, the vehicle is designated to deliver or remove a group of bikes $y_{s',v}^{t}$ to/from station $s'$, which modifies the inventory level of bike station $s'$. Two associated decisions at each timeslot $t$ for vehicle $v$ can be combined as a rebalancing action $a_v^t = \left(z_{s,s',v}^t, y_{s',v}^t\right)$. Given any timeslot, the vehicle can only route to the region that has installed stations and can only reposit bikes jointly considering the physical capacities and the current inventory level of both stations and vehicles.

The following Chapter 4 introduces the proposed method to solve DBSRP. In Chapter 4, a simulator is firstly built to emulate the dynamics of the BSS. Then a Double Deep Q-Network (DDQN) framework is developed that learns to choose the optimal rebalancing actions. Also, an MIP formulation is adapted and finally, DDQN and the MIP are combined as a hybrid approach.

# 4. DATA AND METHOD

This Chapter describes the details of the proposed method to solve DBSRP. Figure 4.1 presents the development pipeline of methods in this Chapter. The historical trip records and station information of the Divvy BSS in Chicago were first preprocessed (Chapter 4.1) and then used as inputs for the developed BSS simulator to train and compare the proposed models with the state-of-the-art benchmarks (Chapter 4.2). Additionally, a demand prediction model based on the historical data was constructed in order to predict future net demands in stations (Chapter 4.3). The DRL-based model—DeepBike—was designed based on DDQN algorithm (Chapter 4.4). The DeepBike improved its policy, i.e., DQN, by learning from feedback signals given the specific system observations. Note that the DeepBike is an online learning model because it repeatedly interacts with the simulator and collects observations of BSS. Once the neural network in DeepBike converges after training, the DeepBike could be implemented to generate online rebalancing decisions for each single vehicle in real-time BSS operation. Furthermore, an MIP algorithm was adapted from Ghosh et al., (2017) as a benchmark to solve the DBSRP (Chapter 4.5). The MIP algorithm was then integrated with DeepBike as a hybrid strategy to further improve the service level of the BSS (Chapter 4.6). Finally, three evaluation metrics were evaluated to measure the performance of different algorithms (Chapter 4.7).

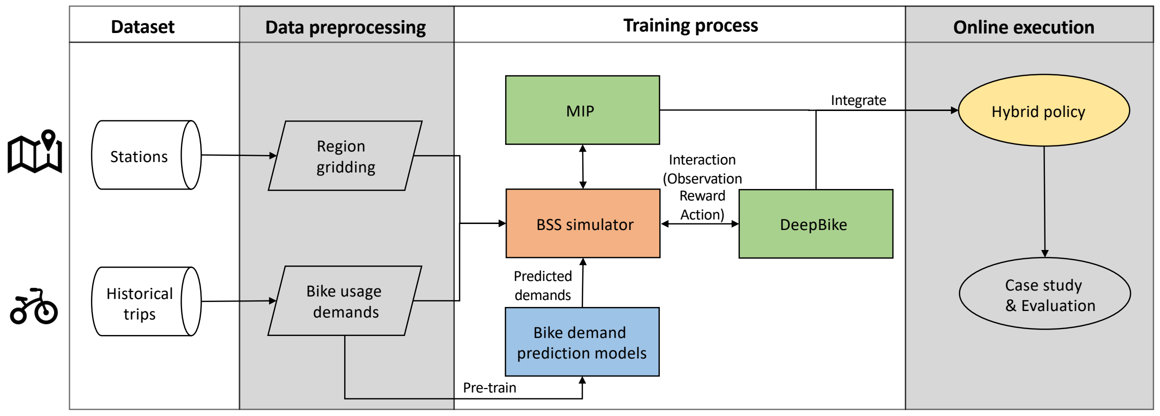


Figure 4.1. The framework of the method development from raw data to the hybrid policy

## 4.1 Data preprocessing

The historical trip data was preprocessed for further study. The real-world trip dataset of Divvy in Chicago in September 2019 were utilized. Specifically, the features of each record in the trip dataset, including pick-up time, drop-off time, start station and end station, were exploited to simulate a bike trip. The records of inactive stations that displayed no demand in the entire day were trimmed. Then the service region with active stations was divided into $51 \times 51$ grids, and each grid has a size of $400 \times 700\ m$ (Oda & Joe-Wong, 2018). The system features of the adjacent stations, such as $s_{s,t}^{Bike}$ and $s_{s,t}^{Dock}$ of each station in the same grid were aggregated and such grid is defined as abstract station. Under this gridding mechanism, the capacity of the abstract station is the sum of capacities of all aggregated stations; the demands in the abstract stations are the sum of demands of all aggregated stations; the routing distance between two abstract stations is the average routing distance of all the stations in two aggregated stations. Specifically in this study, 578 active *Divvy* stations in Chicago were assembled into 451 abstract stations. In the following context, the abstract station will be referred to as station.

## 4.2 Bike-sharing system simulator

To train and evaluate the proposed models, a BSS simulator based on real-world data was developed. The goal of the simulator is to represent the dynamics of the BSS, which includes the vehicle rebalancing, bike rent, and bike return process (Li et al., 2018). For the DeepBike algorithm, an agent is defined as a component in the algorithm to make decisions. The agent learns the rebalancing policy by iteratively interacting with the simulator. During the training process, the rebalancing actions were executed in the simulator and these actions modified the status of the bike stations accordingly, which impacts the future rent and return of bikes.

The simulator kept tracking each trip from its origin station to the destination station to emulate each bike usage demand. The first three weeks of the data were used as training data to train the DeepBike agent and the last week of the data were used for evaluation.

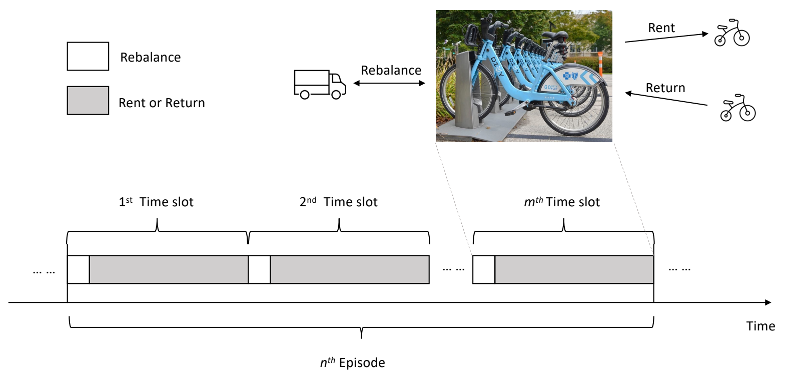


Figure 4.2. The BSS simulation process in one episode with multiple time slots

The details of the simulation process are presented in Figure 4.2.. Each day was viewed as one episode, and was split into 24 time slots so that the length of each time slot was 1 hour. The number of vehicles in the fleet was set to $M = 10$ while each vehicle has the capacity $C_v^{vehicle} = 20$. At the beginning of each episode, the status of stations and vehicles were initialized by setting the inventory levels of them as half of capacities, so that the total number of bikes in the system of Divvy was set to 5,000 (Kou et al., 2020). The initial vehicle locations at the beginning of the episode are randomly selected from the station locations. During each time slot, the simulator first executes rebalancing decisions of all dispatched vehicles which moves bike in or out of the target stations. Similar to existing literature Ghosh et al., (2017), the bike moving process was simplified as an immediate action, ignoring the time needed to complete the rebalancing tasks (i.e. bike pick up, transportation, and drop-off). Also, this study assumes that each vehicle is allowed to visit at most one station in each time slot while each station can also only host at most one vehicle, in order to avoid the conflicts of two vehicles rebalance the same stations. Then, stations update their inventory levels based on the rent and return of bikes that happen in the current time slot . For a customer who wants to rent a bike at a station, if the station has available bikes, the rent demand is successfully fulfilled. Otherwise, one customer loss is recorded at this station, and the customer will find an alternative bike from the top-k nearest neighbor stations that have available bikes (Li et al., 2018a), which is the closed k stations to the target stations. This maintains the total number of bikes in the system throughout the episode. Because the users who fail to start or end the trips at the originally intended stations indicate risks of potential customer loss, the rebalancing policy should ensure that more users succeed to rent or return bikes at their originally intended stations. Similarly, when a customer returns a bike at the destination station, the demand can be satisfied if there are available docks at the station. If not, the customer will have to reroute to one of the alternative top-k nearest neighbor stations that have available docks. At the beginning of the next time slot, new rebalancing action for each vehicle is generated based on the updated system observation at the end of the previous time slot.

The trajectory distance of each movement from one station to another was estimated using Open Street Map dataset based on the road network covering the system region. For each movement of the vehicle, the distance between two stations is the shortest path for cars on the road network during the day with the assumption that vehicles always follow such path to travel to their target stations.

## 4.3 Demand prediction

The demand prediction model was built to estimate the future net demands in each station, which would be appended to the system states as the external observation of the BSS. The demand prediction was implemented using a convolutional neural network to estimate the future bike usage demand in each grid. Same to the gridding mechanism in Chapter 4.1, the output of the network is a $51 \times 51$ image, where each pixel represents the expected net demand number in the grid for the next time slot. System states obtained from the simulator (inventory level of the stations and time information of the current time slot) were input into the network as the input samples. The time information includes the sine and cosine of day of month, day of week, and the hour of day. The heat maps of observed demand distribution in the next time slot made up the true label of the network output. The Divvy's historical trip data in 2018 Q3 (three months) were used to train the prediction model. The first three weeks of each month were used as training dataset and the last week of each month were used as validation dataset.

The architecture of the prediction model was presented in Figure 4.3, which was adapted from Oda & Joe-Wong, (2018). The first two layers convolve the filters of size $3 \times 3$ with the dimensionality of 64 and 128, respectively. Then another two layers convolve the filters with 64 and 1 filters of size $1 \times 1$. At last, the previous layers connect to two dense layers with 4096 and 2601 units. The final layer is reshaped to a size $51 \times 51$ tensor as a heat map to indicate the demand distribution of each cell in the next 1 hour. Note that all layers are activated by rectified linear unit (ReLU) activation functions.

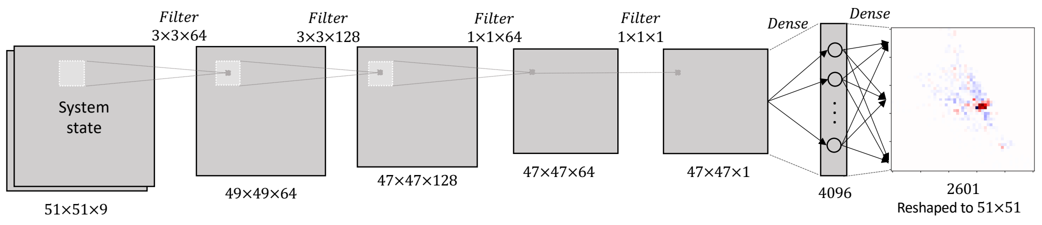


Figure 4.3. Neural network architecture for demand prediction

As comparison, this study employed historical average (HA) as a baseline to compare the prediction performance. In HA algorithm, the average values of the historical demands in each station were considered as the predicted results. Moreover, the study used root mean square error (RMSE) as the evaluation metric.

## 4.4 DeepBike framework

This Chapter introduces the model-free DRL framework—DeepBike. An agent was modelled in the DeepBike framework to learn the optimal rebalance policy. In the framework, the agent interacts with the BSS simulator and learns the optimal bike sharing rebalancing policy with trial and error. At each time slot $t$, the agent sequentially dispatches each idle vehicle to a station (routing action) and decides the number of bikes to be moved into or out of the corresponding station (repositioning action). Then, the simulator executes the rebalancing actions and bike usage demands. At the end of each time slot, the simulator outputs the system observations and rewards to the agent. The agent captures system observation to generate the rebalancing actions for the next time slot $t + 1$. Under the framework of DRL, this study implemented a single-agent multi-entities mechanism, where an agent consecutively generates distributed rebalancing action for each vehicle entity based on a shared DQN. Therefore, the DeepBike model outputs the rebalancing decision for each vehicle individually considering the coordination with the other vehicles, so that the algorithm can be extended to the scenario with changing BSS and different number of vehicles in the fleet. The complete agent learning framework where the agent interacts with the simulator was presented in Figure 4.4..

In the subchapters below, the basic elements (state, action, and reward) in the RL model are explained in Chapter 4.4.1, 4.4.2, and 4.4.3, respectively. Then, Chapter 4.4.4 introduces the DDQN algorithm with experience replay, in which the agent iteratively sampled the memory from the experience replay pool and updated its neural network based on the memory. Chapter 4.4.5 discusses the neural network architecture that was implemented in the DeepBike model. The neural network structure captured the features of the system states and output the Q-values of actions. The Q-value is defined as the action's quality value of collecting future rewards.

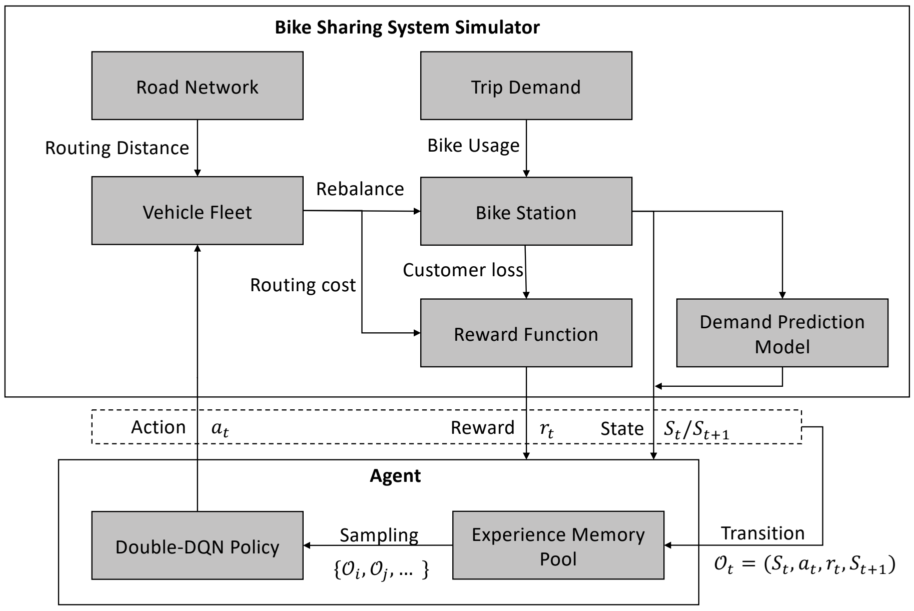


Figure 4.4. Interaction between the agent in the DeepBike model and BSS environment for learning process

### 4.4.1 State

The state variables represent the BSS environment status, which is the external observation of the system $S_t$ described in Chapter 3. Specifically, at the end of time slot $t$, the state $S_t$ consists of the states of all grids, all vehicles, and the expected demands, so that $S_t = (s_t, v_t, x_{t:t+1})$. The environment sends the state $S_t$ to the agent at the beginning of the time slot $t+1$ . The agent thus generates decisions for the real-time BSS operation fully based on the state variables. The state variables are DQN's input layers and then the agent selects the maximum Q-value of the actions based on the approximation of the DQN in the output layer. Such mapping from the states to the Q-values will be introduced in Chapter 4.4.5.

### 4.4.2 Action

The action variable $a_v^t = (z_{s,s',v}^t, y_{v,s'}^t)$ is the two-dimension rebalancing action for vehicle $v$ in the slot $t$, which consists of two components: (1) routing $z_{s,s',v}^t$ and (2) repositioning $y_{v,s'}^t$. The routing denotes which station the vehicle should visit in time slot $t$. In this study, the vehicle was limited to move at most 10 grids vertically or horizontally. Thus the action space of the routing was designed as all the active stations within the reachable grids. The action space of repositioning is the number of bikes to be moved. Also, this study added two extra options: “move the maximum feasible bikes out of the station to the vehicle” (“all out”) and “move the maximum feasible bikes from the vehicle into the station” (“all in”) to the action space based on the observation that the net demand is high particularly in downtown area and moving a large batch of bikes at one time is needed. In this study, the action space of repositioning is designed as “all in”, “all out”, moving five bikes into/out of the station, moving ten bikes into/out of the station and moving zero bikes. Note that the rebalancing action is subject to the physical constraints of the vehicles and stations. For routing, the vehicle only decides to move to a zone if this zone has an active station. For repositioning, the number of moved bike depends on the inventory level of the vehicle and the stations. For instance, if the vehicle is empty, only actions moving bikes out of the station are feasible; while if the vehicle is full, only actions of loading bikes into the station can be taken.

### 4.4.3 Reward

The reward that the vehicle received at the end of each episode is an evaluation of the action. The reward is the number calculated by the stochastic functions of the state of the environment and the action taken (Sutton & Barto, 2018). The number of the reward for vehicle $v$ at the end of time slot $t$ is derived from the function of the state of the environment $S_t$ and the action $a_v^t$ that is taken by vehicle $v$. To reach the objective described in Chapter 3, the agent aims to take rebalancing actions that maximize the expected total future rewards under the specified reward function:

$$\sum_{t=k}^{\infty} \gamma^{t-k} r(a_t, s_t)$$

Where $\gamma \in [0, 1]$ is the time discount factor (Oda & Joe-Wong, 2018).

The design of the reward function $r(a_t, s_t)$ should contain the performance metrics: customer loss and routing cost, as stated in Chapter 3. A customer loss is a customer who fails to rent a bike from the bike station he/she departs because of the empty bike station; or a customer who cannot return the bike to the station he/she arrives because the station is full. Reducing lost customers contributes to increasing the revenue when the customers successfully complete the trips and pays the rent fee. The routing cost is from the gasoline consumed when vehicles travel from one station to another, which depends on the routing distance. Therefore, for vehicle $v$ that moves from station $s$ to station $s'$ and executes the action $a_v^t$ in time slot $t$, the reward function is designed as:

$$r_v^t = r(a_v^t, S_t) = \Delta l_s^t - \beta \cdot P_{s,s'} \quad (1)$$

Where $\Delta l_t^s$ is the reduced customer loss which is equal to the customer loss generated at the station $s'$ at time slot $t$ when the simulator runs without rebalancing *minus* the customer loss occurred at the target station $s'$ at time slot $t$ when executing rebalancing. $\beta$ is the discounted mileage rate to calculate the fuel cost and $P_{s,s'}$ is the distance between station $s$ and station $s'$. Thus, $\beta \cdot P_{s,s'}$ is the normalized fuel cost of vehicle $v$ when executing the rebalancing action $a_v^t$. Because the reward function does not explicitly express the relationship between $a_t$ and $r_t$, DeepBike uses DQN to approximate the specific relationship.

### 4.4.4 DDQN algorithm

In DeepBike approach, the agent utilized the DDQN algorithm with experience replay to dynamically produce rebalancing decisions. The DDQN algorithm is believed to help overcome the overestimation of the Q learning (Van Hasselt et al., 2016). At each time slot, the agent received the observation from the environment and selected an action with the highest Q-value from the action space. At the beginning of the time slot, the action was sent to the simulator from the agent and was executed by the rebalancing fleet to adjust the inventory of the stations. At the end of the time slot, the simulator outputs the system *observation* and a feedback *–reward—* to the agent. By interacting with the simulator, the agent learns good rebalancing policy for DBSRP by optimizing a cumulative future reward signal. Under each specific state, the agent learns to estimate the optimal value of each action for each vehicle $v$, which is defined as the maximum expected sum of achievable rewards (Al-Abbasi et al., 2019). Therefore, in time slot $t$, the Q-value of an action $a$ in a system state $S$ under a given policy $\pi$ is described in equation (2).

$$Q_\pi(S,a) \equiv \mathbb{E}\left[\sum_{k=t}^{\infty} \gamma^{k-t} r_{k,v} \mid S_v^t = S, a_v^t = a, \pi_t\right] \tag{2}$$

And the optimal action-value function is therefore $Q^*(s,a) \equiv \max_\pi Q_\pi(s,a)$. In Equation (2), $\gamma \in [0,1]$ is a discount factor that compensates the importance of the immediate and future reward. For example, if $\gamma$ is small, the agent is expected to maximize the immediate reward. From Equation (2), an optimal policy could be derived based on the optimal value by choosing the maximum action-value in each state: $a_{t,n}^* = \operatorname{argmax}_{a_{t,n}} Q^*(s,a)$. Thus, to obtain the long-term optimal action-value function, the basic idea behind the value-based reinforcement learning is to use Bellman equation as an iterative update (Equation (3)).

$$Q^*(S_t,a_t) = \mathbb{E}_{S_{t+1}}\left[r_t + \gamma \max_{a_{t+1}} Q^*(S_{t+1},a_{t+1}) | S_t, a_t\right] \tag{3}$$

In deep reinforcement learning models, it is common to use neural network as a nonlinear function approximator to estimate the action-value function $Q(s,a;\theta) \approx Q^*(s,a)$. And a neural network in the reinforcement learning model with weights $\theta$ was referred to as a Q-network.

For the agent with untrained neural network, it has no knowledge of the problem to be solved, therefore it is important to tradeoff between exploring the state space and exploiting currently trained neural network to make decisions (Mnih et al., 2013; Thrun, 1992). Epsilon-Greedy policy was implemented for the agent to choose its action during the training. Under the given state, the agent decided its action by selecting the highest value in the output of the Q-network with probability $1-\epsilon$, while with probability $\epsilon$, it chooses a random action (Mnih et al., 2013). Specifically in this study, the initial $\epsilon$ was set to 0.99. It decreased linearly to 0.05 over $20000$ training steps and kept unchanged afterwards. The state of the system would be modified by the vehicle's rebalancing during the training comparing with running no rebalancing actions. When the agent has little knowledge (the neural network that has not been well-trained) in the system, the state of the system modified by suboptimal rebalancing actions might add difficulty for the agent to learn the optimal policy. A parameter $\alpha$ was introduced in order to alleviate such effect and make the agent receive the original states of the system. It is the probability of keeping a vehicle to stay in its original station and move no bikes in the current time slot. Specifically, $\alpha$ linearly decreases from 0.8 to 0 over the first 20000 training steps. In other words, only 20% of the vehicles will perform rebalancing actions at the beginning of the training process.

The agent extracted the representation of the system states by neural network and improved its policy by iteratively updating the network. $\theta_i$ and $\theta_i^-$ denoted the weights of two neural networks in the context of DDQN (Van Hasselt et al., 2016). Under the mechanism of experience replay, the agent iteratively interacted with the simulator and stored training samples $(S_t, a_t, r_t, S_{t+1})$, which records the transition from the old state $S_t$ to the new state $S_{t+1}$ after taking the action $a_t$, into the experience replay pool. In each learning iteration, the agent randomly selected a batch of samples from the pool to train its neural network. This breaks the temporal correlation and avoids the frequent utilization of a single experience when training the agent (Lin, 1992). In this study, the neural network was trained by sampling a mini batch of 128 samples from the experience replay pool and updating weights $\theta_i$ at each time step $i$ with the aim to minimize the mean-squared error (MSE) loss function below (Van Hasselt et al., 2016):

$$L_i(\theta_i) = \mathbb{E}_{s_t,a_t,r_t,s_{t+1}}\left[\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_i^-) - Q(s_t, a_t; \theta_i)\right)^2\right] \tag{4}$$

Where the agent replaces the optimal Q-values $y_t$ that is predicted by network with the estimated target values $r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_i^-)$ as defined in Equation 31.

### 4.4.5 DQN architecture

A specific architecture of neural network was built to map the BSS observations to the Q-values of the actions, which is demonstrated in Figure 4.5. This architecture implemented the convolutional neural networks and fully connected neural network to approximate the Q-values of the two-dimension actions: repositioning and routing.

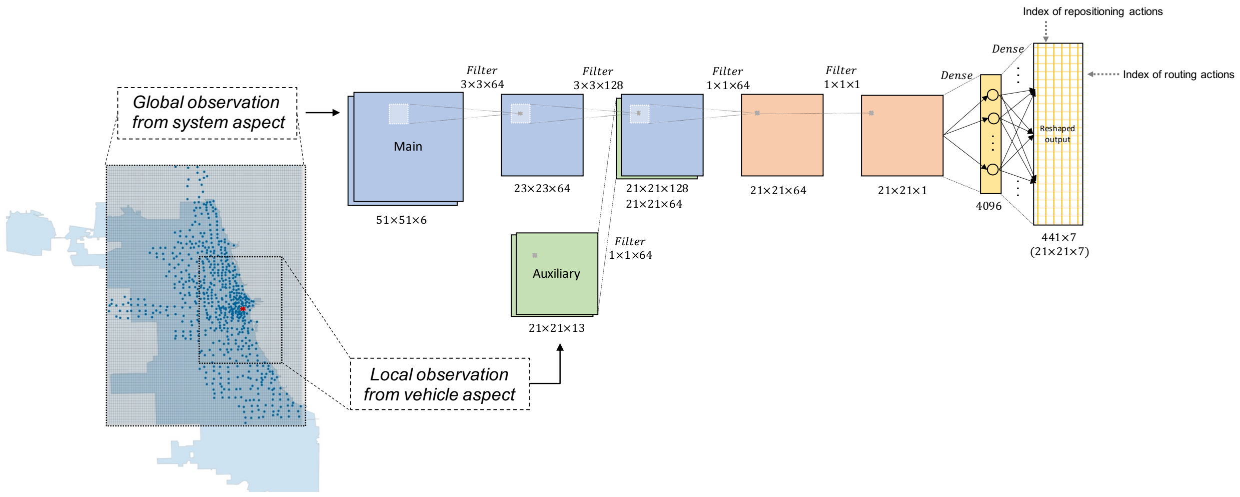


Figure 4.5. Deep Q-Network architecture of DeepBike

The spatial and temporal input features were fed into the input layers of the neural network, shown in Table 4.1. These features were generated fully based on the system observation $S_t$. Recall that this work divided the area into $51 \times 51$ cells. The vehicles were constrained to move maximally 10 vertical cells (right or left) and 10 horizontal cells (up or down). This led to a $21 \times 21$ map (441 cells in total) so that a vehicle could move to any of the 440 cells to visit the station there or it could stay at its current cell. The constraints of moving distance ensured that vehicles could reach their target stations within a short time. In order to capture environment information of more perspectives, this work designed two channels of input features: global and local. The global features were the observation from the system aspect, which is noted as the main input. The stacked planes of the main inputs in Table 4.1(a) were the heat maps with size $51 \times 51$ and each grid in the plane denotes the value of one of the input features from Table 4.1, for example, a grid could represent the number of bikes in the this grid if there is a station. Specifically, the features included the number of bikes and slots in vehicles in the cells, the number of bikes and docks in stations in the cells, the expected positions of vehicles in the next one hour, and the predicted net demands in stations in the next one hour. The local features were the date and time information and the observation from the vehicle's perspective when putting the vehicle in the center of the plane, which was noted as the auxiliary input. The stacked planes of the auxiliary inputs in Table 4.1(b) were the heat maps with size $21 \times 21$, in order to match the moving constraints of the vehicles aforementioned. The auxiliary planes were the day of the month, day of the week, hour of the day, the position of the vehicle(binary), the coordination of the vehicle, the coordination of the surrounding cells, the distance from the vehicle to other cells, and the action feasibility of each cell.

Table 4.1. System representations of the input layer in the neural network.

(a) The main inputs depict the global system observation.

| Input | Feature | Number of planes | Plane size |
|---|---|---|---|
| Main | Bikes in vehicle | 1 | $51 \times 51$ |
| | Space in vehicle | 1 | $51 \times 51$ |
| | Expected number of available vehicles in each grid in the next hour | 1 | $51 \times 51$ |
| | Bikes in stations in each grid | 1 | $51 \times 51$ |
| | Docks in stations in each grid | 1 | $51 \times 51$ |
| | Predicted number of net demand in each grid in the next hour | 1 | $51 \times 51$ |

(b) The auxiliary inputs depict the time information and local system observation.

| Input | Feature description | Number of planes | Plane size |
|---|---|---|---|
| Auxiliary | Day of month | 2 | $21 \times 21$ |
| | Day of week | 2 | $21 \times 21$ |
| | Minute of day | 2 | $21 \times 21$ |
| | Current position of this vehicle (Binary) | 1 | $21 \times 21$ |
| | Globally normalized coordinates of this vehicle | 2 | $21 \times 21$ |
| | Locally normalized coordinates of this vehicle | 2 | $21 \times 21$ |
| | Normalized distance from other cells to the center position (vehicle) | 1 | $21 \times 21$ |
| | Whether moving to this cell is legal | 1 | $21 \times 21$ |

Two channels of inputs were sequentially passed through convolutional and dense layers to nonlinearly estimate the Q-values of different actions. To reduce the complexity of the neural network, the main inputs were first applied average pooling and the pool size is 2 (Goodfellow et al., 2016). 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 auxiliary inputs entered the convolutional layers, which has 64 filters of size $1 \times 1$ and ELU activation. Foregoing convolved inputs were stacked and then they entered another two convolutional layers with 64 and 1 filters of size $1 \times 1$. Finally, the output of the convolutional layers was pushed into two dense layers with 4096 and 3087 units, respectively. The output of the dense layer with 3087 units was reshaped into a two-dimension tensor with size $441 \times 7$, in order to match the size of the routing and repositioning action space, respectively.

The output of the DQN was constructed as a two-dimension tensor, which corresponds to the approximated Q-values of the actions. As mentioned in Chapter 4.4.2, the agent produces two branches of actions for one vehicle and these branches of action were inherently highly correlated. Practically, the vehicles can drop off or pick-up bikes at a station (execute repositioning action) only if the station is visited (execute routing action). Therefore, as mentioned before, the output tensor was reshaped to a two-dimension table to represent the coupling between the two actions. For every selected maximum Q-value in the output, the row index in the table refers to the index in the routing action space with size of 441, while the column index in the table refers to the index in the repositioning action space with the size of 7.

Since the state space ($\approx 10^{15988}$ in 24 time slots) and action space ($\approx 7 \times 10^{4}$ for each vehicle in 24 time slots) was significantly large, the training process requires a long time. This work applied *action mask* to prune the unavailable actions under each system state, which will result in faster training convergence and will improve the learning efficiency (Ye et al., 2019). As the plane of legal map was added into the auxiliary inputs, the output was multiplied by the legal map to mask the unavailable actions. For routing legal map, if the grid contains stations and has not been assigned rebalancing action, the corresponding pixel in the legal map would be denoted as 1; if the grid has no installed station, the pixel would be denoted as 0 (unavailable routing action). For repositioning legal map, only if the number of bikes and slots of the vehicle could satisfy the repositioning action, the corresponding pixel in the legal map would be denoted as 1, otherwise the pixel would be denoted as 0 (unavailable repositioning action).

In order to train the neural network in the DeepBike model, the bike usage records of three consecutive weeks from Monday 09-02-2019 to Sunday 09-22-2019 were used. Namely, each training epoch contains 21 episodes, and each episode represents one day's data. For each episode, one day was defined to start from 0 a.m. to 0 a.m. (next day). Each episode's first time slot was run with "no rebalancing" strategy to obtain the initial BSS status for the agent. During the training, the BSS status was refreshed at the beginning of each episode, so that the capacities of stations and vehicles were all reset to half. In this way, the input training data for each epoch loop could be shuffled, ensuring the scalability of agent to execute the rebalancing policy at each single day independently. The neural network was trained for over 150k time steps while keeping the most recent 40k experiences in the replay buffer. Also, the first 20k steps was set as the exploration steps.

## 4.5 Mixed Integer Programming formulation and other baselines

This research used the MIP formulation first as a benchmark algorithm and then integrated it with DeepBike as a hybrid approach with the aim to further improve the effectiveness of the rebalancing policy. This research also compared the proposed approaches with three additional benchmarks: no rebalancing, random rebalancing, and greedy rebalancing.

### 4.5.1 Mixed integer linear programming

An MIP approach that exploits Lagrangian dual decomposition and abstraction mechanisms was adopted from (Ghosh et al., 2017) to solve DBSRP. Based on the notations defined in Table 3.1, the decision and intermediate variables in the MIP approach were described in Table 4.2., and the objective function and constraints were formulated in Equations (5) – (16). This work replaced the satisfied number of customers in the objective function with the number of customer loss, and added associated constraints to calculate customer loss. Additionally, in this MIP model, instead of modelling the demand by bike flows, this work used net demands as aforementioned in Chapter 3.

Table 4.2. Decision and intermediate variables and constants in the model

| Variable type | Variable | Description |
|---|---|---|
| Decision | $y_{s,v}^{t}$ | The number of bikes moved into/out of the station $s$ by vehicle $v$ in time slot $t$ |
| | $z_{s,s',v}^{t}$ | Binary variable that is equal to 1 if vehicle $v$ moves from station $s$ to $s'$ in time slot $t$ |
| Intermediate | $x_s^t$ | Intermediate inventory level of the station $s$ in time slot $t$ |
| | $d_s^{station,t}$ | The number of bikes parked at station $s$ at the beginning time slot $t$ |
| | $d_v^{vehicles,t}$ | The number of bikes carried by vehicle $v$ in time slot $t$ |
| | $l_s^t$ | The customer loss occurred at station $s$ in time slot $t$ |
| Constant | $P_{s,s'}$ | The estimated distance between station $s$ and station $s'$ |
| | $\beta$ | The discounted mileage rate (cents per mile) |
| | $C_s^{station}$ | The capacity of station $s$ |
| | $C_v^{vehicle}$ | The capacity of vehicle $v$ |
| | $\sigma_{v,s}^{t}.$ | $\sigma_{v,s}^{0}$ is set to 1 if vehicle $v$ is in station $s$, which sets the initial location of the vehicle $v$ if $t = 0$. $\sigma_{v,s}^{t}$ is set to 0 if $t > 0$ |
| | $F_s^t$ | The net demands of station $s$ in the time slot $t$ |

$$\min_{z,y^+,y^-} \sum_{t,s} l_s^t + \sum_{t,v,s,s'} \beta \cdot P_{s,s'} \cdot z_{s,s',v}^t \tag{5}$$

s.t.

$$d_s^{station,t} + \sum_{v \in V} y_{s,v}^t + F_s^t = x_s^t \quad \forall s,t \tag{6}$$

$$d_s^{station,t+1} = min\{max\{0, x_s^t\}, C_s^{station}\} \quad \forall s,t \tag{7}$$

$$l_s^t = \left| d_s^{station,t+1} - x_s^t \right| \quad \forall s,t \tag{8}$$

$$d_v^{vehicles,t} + \sum_{s \in S} -y_{s,v}^t = d_v^{vehicles,t} \quad \forall v,t \tag{9}$$

$$\sum_{s' \in S} z_{s,s',v}^t - \sum_{s' \in S} z_{s',s,v}^{t-1} = \sigma_{v,s}^t \quad \forall s,v,t \tag{10}$$

$$\sum_{s' \in S, v \in V} z_{s,s',v}^t \leq 1 \quad \forall s,t \tag{11}$$

$$\sum_{s' \in S, s \in S} z_{s,s',v}^t \leq 1 \quad \forall v,t \tag{12}$$

$$-C_v^{vehicle} \cdot \sum_{s' \in S} z_{s,s',v}^t \leq y_{s',v}^t \leq C_v^{vehicle} \cdot \sum_{s' \in S} z_{s,s',v}^t \quad \forall s,t,v \tag{13}$$

$$d_s^{station,0} = \frac{1}{2} C_s^{station} \ \forall s, d_v^{vehicles,0} = \frac{1}{2} C_v^{vehicle} \ \forall v \tag{14}$$

$$0 \leq x_s^t, -C_v^{vehicle} \leq y_{s,v}^t \leq C_v^{vehicle}, 0 \leq d_s^{station,t} \leq C_s^{station}, 0 \leq d_v^{vehicles,t} \leq C_v^{vehicle} \tag{15}$$

$$z_{s,s',v}^t \in \{0,1\} \tag{16}$$

The specific details of the objective function and constraints were described below. For objective function in Equation (5), it represents the trade-off between the customer loss and fuel consumption. Similar to the reward function of DDQN model, the first term in the objective function is the sum of the customer loss and the second term is the sum of the normalized dollar value of the gasoline cost (mileage rate/single trip cost of using the bike). For constraints (6) and (7), the flows of bikes moved in and out of the stations by vehicles and customers are conserved. The number of bikes in the stations is always less than the capacity and cannot be negative. The constraint (8) indicates the lost customer at each station, which is the difference between the expected inventory level of the station and the actual inventory level. Similar to constraints (6) and (7), constraint (9) ensures the number of bikes in and out of the vehicles are conserved. For constraint above , it enforces the flows of vehicles arriving and departing from the stations are preserved. Specifically, the number of vehicles moving out of the station $\sum_{s' \in S} z_{s,s',v}^t$ is equal to the number of vehicles previously moving into the stations $\sum_{s' \in S} z_{s',s,v}^{t-1}$ plus the number of vehicles already in that station $\sigma_{v,s}^{t}$. Constraint (11) and (12) guarantee the requirement that at any time step at most one vehicle can visit one station and one vehicle can be present at most one station, respectively. Constraint (13) couples the repositioning and routing, enforcing that vehicles can only pick-up or drop-off bikes at the visited station. Constraint above confines the initial inventory levels of the stations and vehicles as half capacities, which is consistent with the setup in the BSS simulator in Chapter 4.2. Constraint (15) enforces that the number of repositioned bikes and the inventory levels of the stations and vehicles do not violate the physical capacities at all times. Constraint (16) defines the $z_{s,s',v}^{t}$ as binary variables.

### 4.5.2 Baselines

Except for the DeepBike and the MIP approaches aforementioned, another three methods (no rebalancing, greedy rebalancing, and random rebalancing) were used to solve DBSRP and compare the performances of them with our proposed hybrid strategy.

***No rebalancing***

For No Rebalancing strategy, the BSS does not execute any rebalancing actions.

***Greedy rebalancing***

The Greedy Rebalancing (GR) strategy produces rebalancing decisions that follows three principles. First, for a vehicle full of bikes, it always travels to the nearest and most deficient stations (the station that has the most available docks) and drop-off the most possible bikes there (fill the station to full if there are enough bikes in the vehicle). The exact number of bikes unloaded depends on the available docks at the station and available bikes in the vehicle. Second, for an empty vehicle, it always goes to the nearest and most congested (the station that has the most available bikes) stations and pick-up the most possible bikes there (discharge the station to empty if there are enough slots in the vehicle). The moved bikes depend on the available bikes of the station and available capacity in the vehicle. At last, except for full and empty vehicles, other vehicles route to the nearest and the most deficient or congested stations (absolute value of the number of bikes minus the number of docks) and pick-up or drop-off the most possible bikes there, considering the inventory level of the station and vehicle.

***Random rebalancing***

The Random Rebalancing strategy was defined that a vehicle randomly selects a station and randomly drop-off or pick-up a group of bikes considering the current number of bikes and docks in the vehicle and station.

## 4.6 The hybrid strategy: DeepBike + MIP

A hybrid strategy that combines model-free reinforcement learning algorithms and MIP algorithms has the potential to leverage the benefits of both methods (Chen et al., 2021). Therefore, this work designed the hybrid strategy that uses both the proposed DeepBike model and the modified MIP approach in making rebalancing decisions. As a learning-based model-free method, the DeepBike could generate online decisions within short computation time and adapt to the changing environment. The MIP approach is model-based and can provide (near-) exact solution from the problem.

To execute the hybrid strategy, the vehicle fleet was split into two parts, with one controlled by the DeepBike policy and the other controlled by the MIP approach. Specifically, as shown in Algorithm 1, at the beginning of each episode, each vehicle chooses to execute MIP rebalancing policy in this episode with probability $\rho$ if a uniform random number is less than or equal to $\rho$, else it follows DeepBike rebalancing policy to generate decision with probability $1 - \rho$. In this way, the selection of the policies among vehicles will be diverse, but each vehicle only follows one policy (either MIP or DeepBike) so that the actions of each vehicle is consistent with one policy in each episode. In this work, $\rho$ was set to 0.2.

```
Algorithm 1: The hybrid strategy
Initialize the probability $\rho$ of choosing MIP policy;
for each episode do
    for each vehicle $v$ in the fleet do
        $n \leftarrow$ a uniform random number between 0 and 1;
        if $n \leq \rho$ then
            Choose the MIP policy;
        else
            Choose the DeepBike policy;
        end
    end
    for $t = 0$ to $T$ do
        Send the system observation $S_t$ to the agent;
        Generate rebalancing actions $a_t$ according to the policy;
        Execute rebalancing actions $a_t$ ;
        Rent/Return of bikes;
        Update the system observation;
    end
end
```

## 4.7 Evaluation metrics

The following three metrics were used to evaluate the performance of different rebalancing strategies.

(1) Customer loss is defined as the number of customers who failed to rent the bikes at the primary departure stations or return the bikes at the original arrival stations.

(2) Vehicle routing distance (VRD) is also known as vehicle-miles-traveled (VMT). The total VRD of the vehicle is the sum of the distance that the vehicle travels among the visited targeted stations.

(3) Full-empty rate is the proportion of stations which are completely full or empty. The number of stations being full or empty in timeslot $t$ is denoted as $f_t$. Thus, the full-empty rate is $\frac{f_t}{N}$, where $N$ is the total number of stations in the system.

(4) Improved profit is the profit of reduced customer loss compared to no rebalancing *strategy* minus the routing cost of the rebalancing fleet. This work assumes that once a lost trip is avoided, the customers will always rent the bikes to finish their planned trips. Thus, the reduced customer loss is the improved number of customers who successfully complete their trip. The average profit of each reduced customer loss is

therefore the cost of each single trip (\$3.3, Divvy, (2021)). The routing cost is the mileage rate (58cents/mile in 2019, IRS, n.d. ) multiples the routing distance.

(5) Greenhouse Gas (GHG) emission is one of the metrics to evaluate the environmental impact of the BSS (Luo et al., 2019). The users might switch to other transportation modes that generate higher GHG emissions (such as taxis) if users fail to rent a bike (Kou et al., 2020). The rebalance strategy could improve the usage of bikes by redistributing the bikes to the proper stations to meet the demand of renting and returning bikes, and therefore help improve the system's GHG emission reduction. However, bike rebalancing is another source of GHG emission as the operators usually use automobile to serve the demands of rebalancing (Luo et al., 2019). The average emission reduction per trip in Divvy, Chicago is 0.52210 kg CO2-eq (Kou et al., 2020) . Thus, each avoided customer loss (lost trip) could help reduce 0.52210 kg CO2-eq GHG. Furthermore, the vehicle emission rate is 2.13 kg CO2-eq/(t·km) while each bike weights about 20kg (Luo et al., 2019, 2020), which depends on the number of bikes carried by a vehicle and the distance the vehicle travels. Therefore, a vehicle carrying 1 ton of bikes and moving 1 km would produce around 2.13 kg CO2-eq GHG emission.

# 5. RESULTS

The numerical evaluations used one-week bike trip records from Monday 09-23-2019 to Sunday 09-29-2019 to compare the performance results of the proposed algorithms with aforementioned benchmarks. The effectiveness of the demand prediction model is shown in Chapter 5.1. The training results of the DeepBike model are presented in Chapter 5.2. Finally, the performance results and the sensitive analysis are described in Chapter 5.3 and 5.4, respectively. Chapter 5.5 conducts the analysis to evaluate why the hybrid strategy could outperform other strategies, such as DeepBike and MIP approaches.

## 5.1 Demand prediction model

The RSME for validation dataset of our CNN model and HA model are 0.9207 and 1.4527, respectively, which illustrates the effectiveness of the prediction model. Figure 5.1 provides an example of the predicted and ground truth bike usage demand heat map.

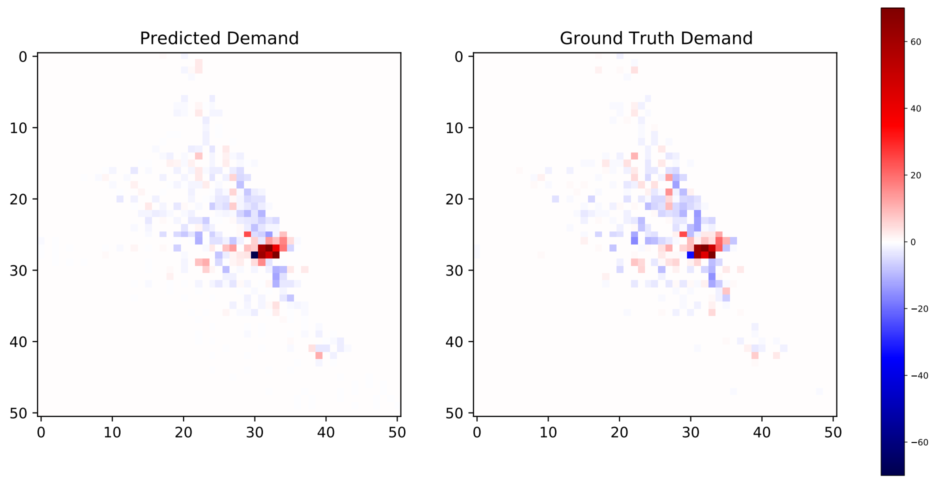
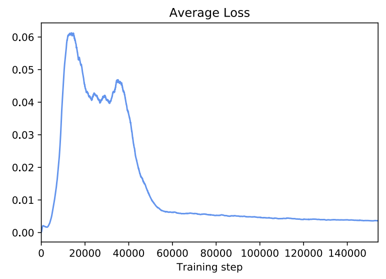


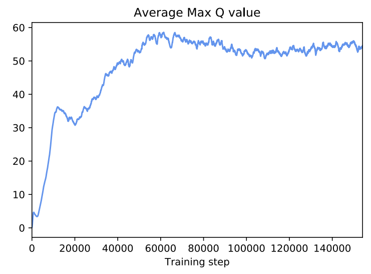
Figure 5.1. An example of the predicted demand and the actual demand heat maps at 8 a.m. on July 25, 2018

## 5.2 Training results

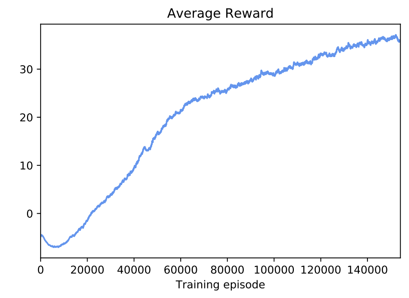
The training curves keep tracking the training process of the neural network in DeepBike model.  The curve show the convergence of the neural network and the increasing reward collected by the agent. Figure 5.2 (a) shows the curve of the average loss $L_i(\theta_i)$ of the neural network which was defined in Equation 32. The aim of training the neural network is to minimize the MSE between the $r_t + \gamma \max\limits_{a_{t+1}} Q(\mathrm{s}_{t+1}, \mathrm{a}_{t+1}; \theta_i^-)$ and $Q(s_t, a_t; \theta_i)$ . The value of the average loss in Figure 5.2 (a) starts to reduce after 10k steps and converges to 0.005 with the increasing of the steps thereafter. Figure 5.2 (b) indicates the average maximum Q-value of the output of the neural network. As the agent always selects the highest Q-value from the output, the result shows that the max Q-value increases for the first 60k steps and reaches 60. Then the Q-value reduces, as the vehicle weighs the fuel cost and reduced customer loss. Figure 5.2 (a) and (b) indicate that the Q-values of the actions predicted by the neural network converge to the optimal value. Figure 5.2 (c) shows the average collected reward of each vehicle in one episode. The reward reduces at the beginning because the BSS environment is unknown to the agent and the agent explores the state-action space to collect higher reward. As the agent continues to improve the policy and collect more reward, the mean reward increases, and the policy could produce better rebalancing decisions.



(a) Average loss curve



(b) Average Max Q-value curve



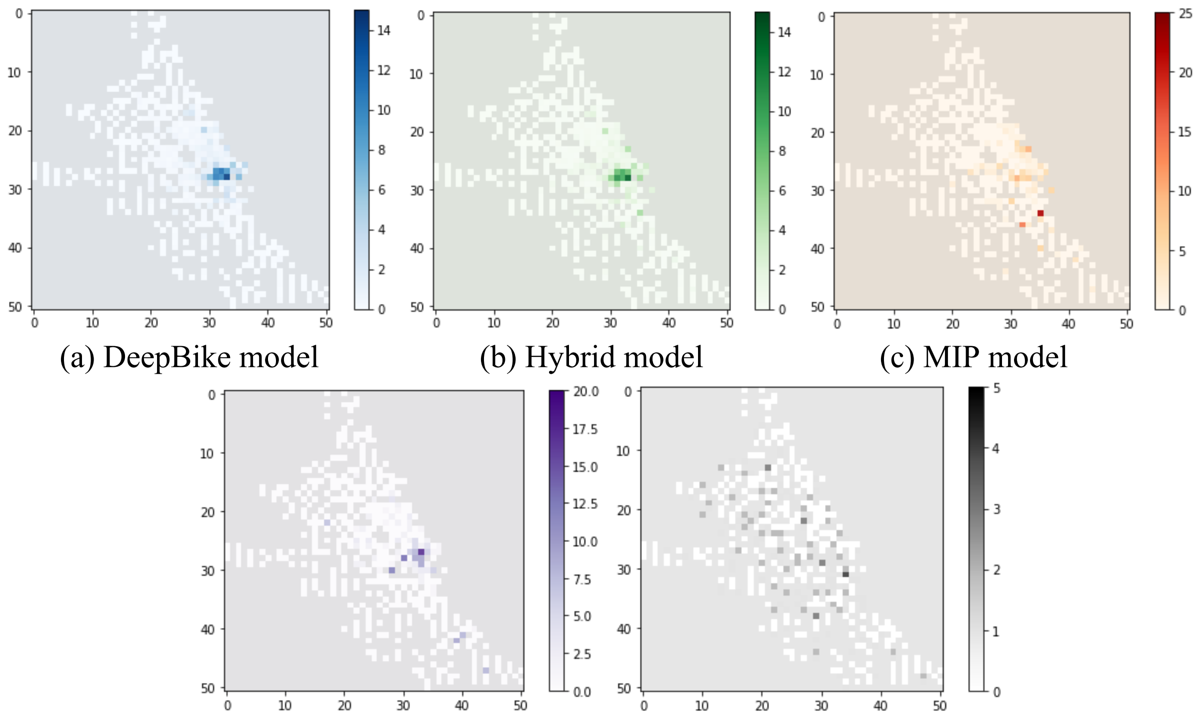
(c) Average reward curve

Figure 5.2. Training curves of the DeepBike model

## 5.3 Performance results

This subchapter discusses the evaluation performances of different algorithms.

Figure 5.3 presents an example of the distribution of the stations visited by vehicles under different policies on Wednesday September. 25, 2019, which reveals the rebalancing patterns of the different policies. Specifically, the DeepBike model focuses on the downtown area where the demand is relatively higher than other regions. The hybrid model mainly dispatches the vehicles to the downtown area but still assigns some vehicles to the regions away from the downtown area. Both of the MIP model and the greedy model dispatch the vehicles to some focused stations (darker cells), but the stations visited are more scattered.


(a) DeepBike model (b) Hybrid model (c) MIP model

(d) Greedy model (e) Random model

Figure 5.3. An example of the distribution of the visited stations under different policies on Wednesday, September 25, 2019

Figure 5.4 demonstrates the improved profits of different policies compared to *no rebalancing strategy*, which reflects the performance of achieving the rebalancing objective defined in Chapter 3. The hybrid strategy outperforms other strategies in improving total profits in a week by +29.9% compared to the DeepBike and +39.8% compared to the MIP, which verifies the effectiveness of the proposed hybrid strategy.

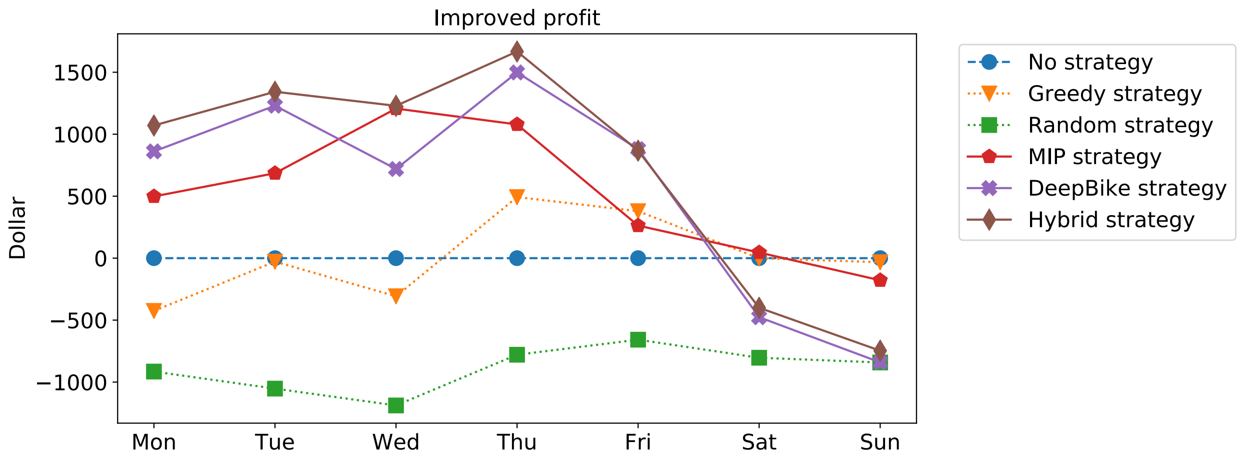


Figure 5.4. Daily improved profits of different policies

Figure 5.5 describes the total customer loss of the entire system in each day under different rebalancing policies. The daily customer loss compared with *no rebalancing strategy* could reflect the effect of the policy in improving/declining the satisfaction of the bike users. The results indicate that, with a rebalancing fleet of 10 vehicles, the hybrid strategy outperforms other baselines in reducing the customer loss and reaches the lowest loss in weekdays. Specifically, for the entire week, the hybrid policy reduces the customer loss by 3.1% compared to the DeepBike approach, 9.4% compared to the MIP approach, and 20.8% compared to no rebalancing approach. Note that the distributions of weekdays (15 days in the training dataset) and weekends' (6 days in the training dataset) records in the dataset are imbalanced. Therefore, the DRL-based agent estimates the Q-values of actions in weekdays more accurately than that in weekends.

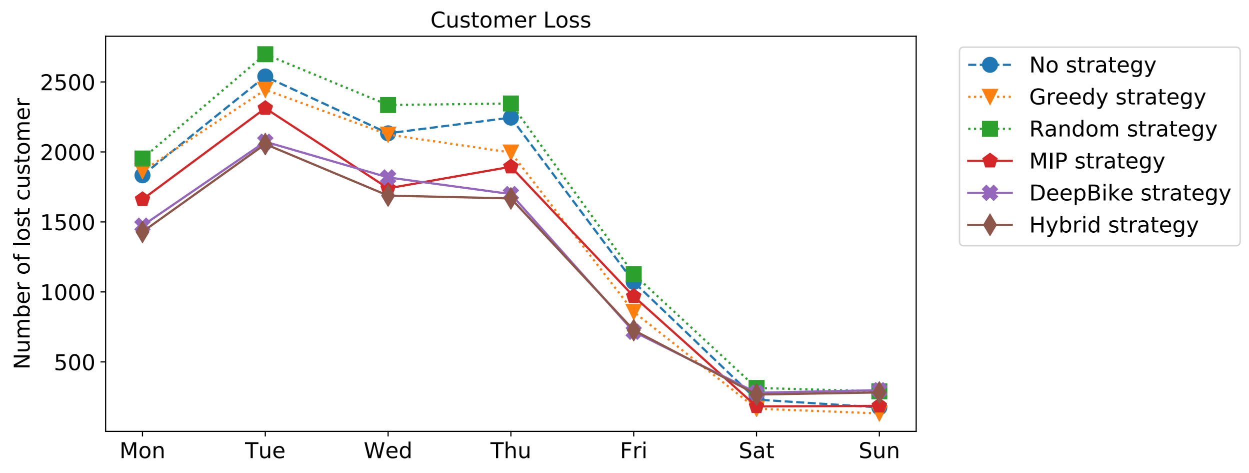


Figure 5.5. Daily customer loss of different policies

Figure 5.6 plots the total vehicle routing distance of the fleet in each day under different rebalancing policies. The vehicles following hybrid strategy travel longer distance than the vehicles controlled by the MIP policy, but travel shorter distance than the vehicles under the control of the DeepBike policy, Greedy rebalancing strategy, and Random rebalancing strategy. The results reveal that the hybrid strategy and the DeepBike strategy make the vehicles route more actively among stations than the MIP strategy and therefore the vehicles are in higher utilization. On weekdays, it is consistent that the vehicles are in high utilization so as to reduce more customer loss. On the other hand, on the weekends, the vehicles controlled by the hybrid and DeepBike strategies are over utilized since they do not help reduce customer loss, but still travel long distance. This suggests that the weekend rebalancing fleet can be reduced to avoid too much fuel cost on the weekends.

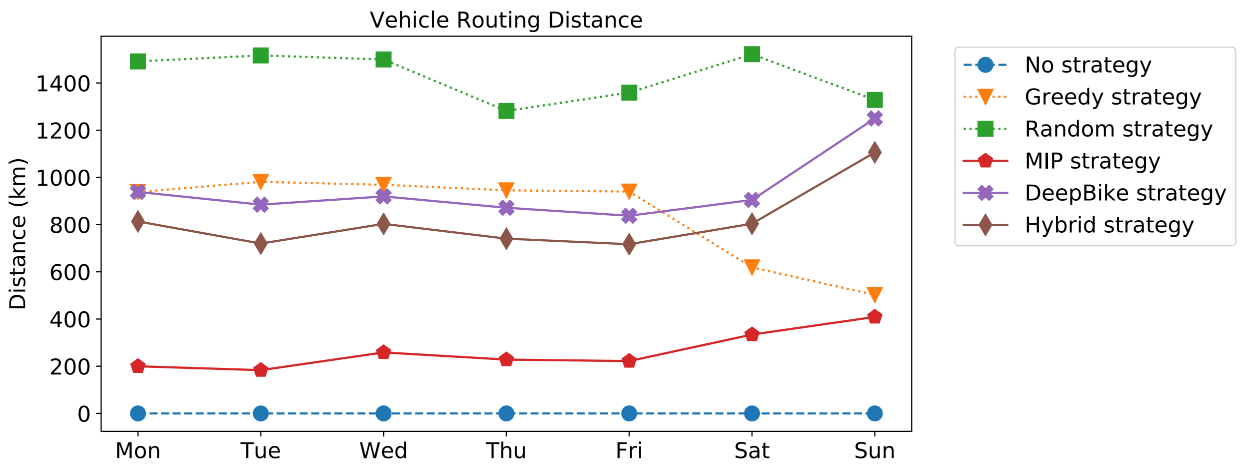


Figure 5.6. Daily vehicle routing distance of policies

Figure 5.7 shows the rate of the stations becoming full or empty after being rebalanced under different policies. The results of  andFigure *5.7* indicate that the MIP policy distributes the vehicles to more stations compared with the hybrid and DeepBike models. The MIP policy redistributes the bikes to avoid starvation or congestion of the stations. Additionally, the hybrid and DeepBike policies prioritize some key stations (darker cells) in the downtown areas and redistribute the bikes to these stations. The key stations have larger bike flows as shown in Figure 5.1. Fulfilling more demands in these stations through rebalancing could produce more long-term cumulative rewards.

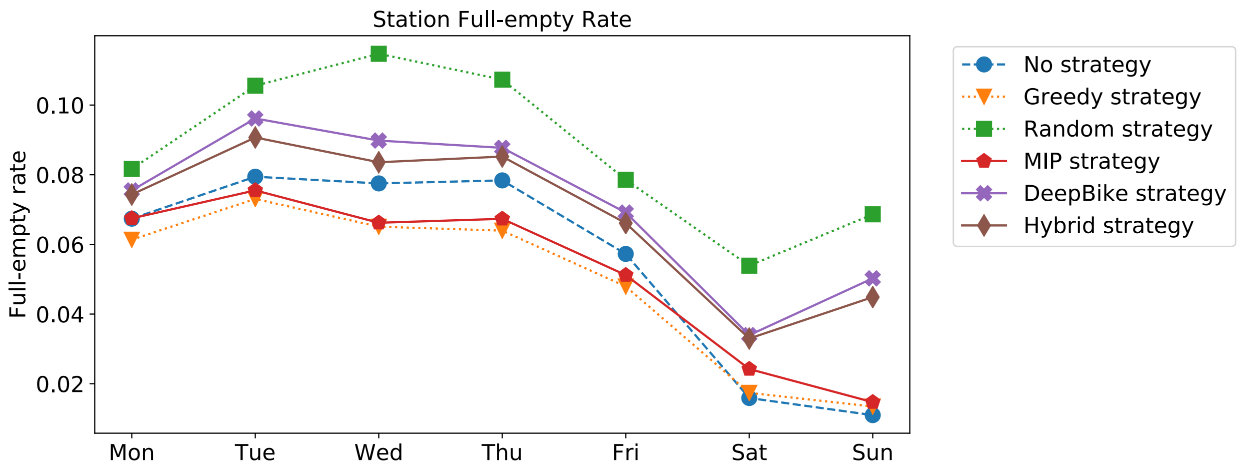
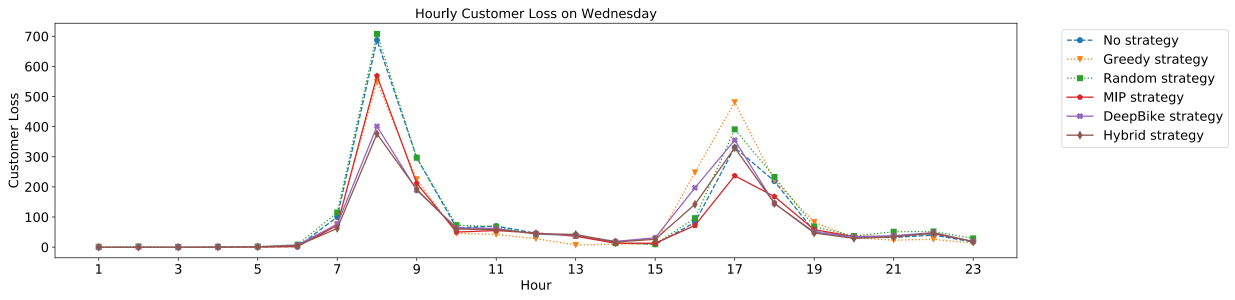
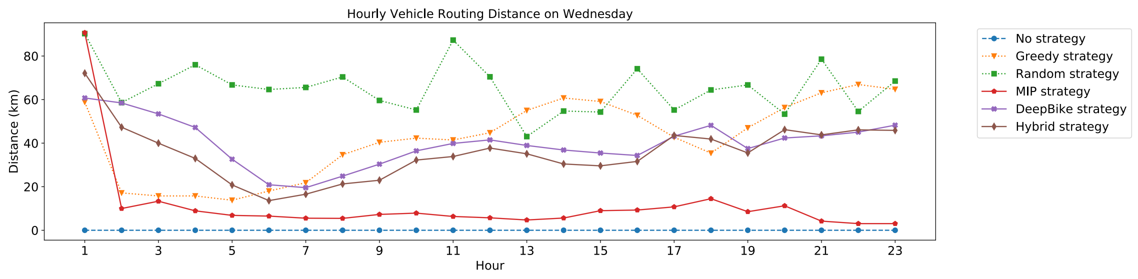


Figure 5.7. Daily station full-empty rate of policies

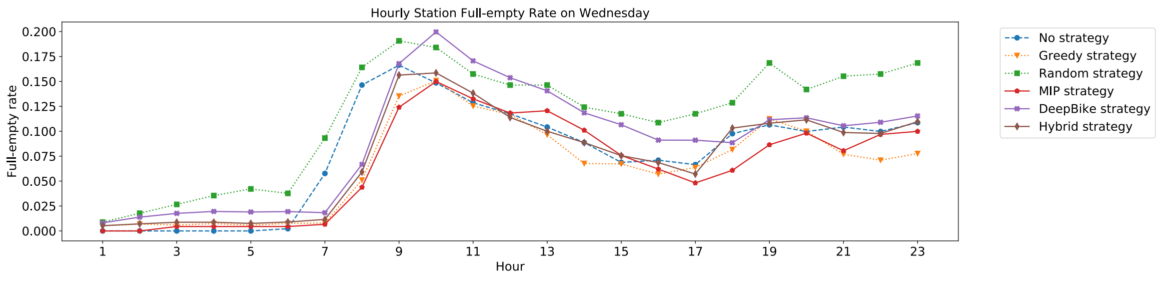
An example of results among different policies in Figure 5.8 reveals how vehicles perform in each hour on Wednesday, September 25, 2019. The MIP assigns vehicles to some prioritized regions at the beginning of the day because the routing distances in the first hour of the day are much longer than that in the following hours. The vehicles then are kept routing in those regions, which could in turn produce low routing distance. On the other hand, the DeepBike generates active actions that keeps the vehicle routing among regions. As the DeepBike model provides “all in” and “all out” actions, the agent learns to select them frequently which reduces the customer loss in the prioritized stations but makes some other stations full or empty instead. Since the hybrid strategy exploits the advantages of both DeepBike and MIP, it could further reduce the unsatisfied customer compared with all other algorithms while reaching lower full-empty rate and shorter routing distance compared to DeepBike.



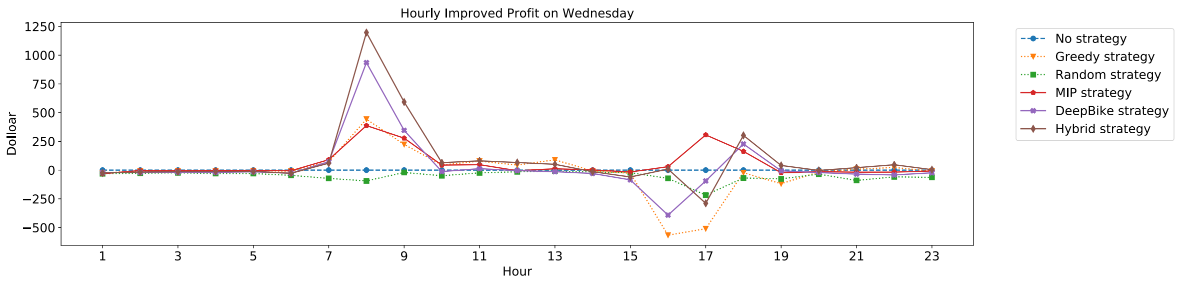
(a) Hourly customer loss



(b) Hourly vehicle routing distance



(c)Hourly full-empty rate



(d) Hourly improved profit (unit: dollar)

Figure 5.8. An example of hourly performances under different policies on Wednesday September 25, 2019

Table 5.1 and Table 5.2 demonstrate the environmental and financial effects under different rebalancing policies, respectively, to understand why developing the rebalancing policy has significant contributions. Table 5.1 shows the reduced GHG emission under different policies when a customer loss is avoided and a user could successfully rent bikes from BSSs, and the generated GHG emission from vehicles when routing among stations. Specifically, the reduced GHG emission from BSS in a week under Hybrid strategy is 1237.05 kg $CO_2$-eq, which is +65.7% and +13.8% higher compared to MIP (746.41 kg CO2-eq) and DeepBike (1087.07 kg CO2-eq) strategies, respectively. On the contrary, the generated GHG emission caused by the routing of vehicles under Hybrid, DeepBike and MIP strategies in a week is 2390.97 kg CO2-eq, 2827.05 kg CO2-eq and 679.46 kg CO2-eq, respectively. The results indicate that the hybrid strategy could help avoid more customer loss than other strategies, which results in higher GHG emission reduction. However, the hybrid and DeepBike strategies actively dispatch the vehicles to stations, which leads to longer routing distance and greater corresponding vehicle emission compared to the MIP strategy. Table 5.2 indicates the improved profits obtained by rebalancing the BSS. The avoided potential customer loss at the stations could contribute the rental fee to the BSS because the users could successfully rent a bike. For the entire week, the hybrid strategy outperforms other strategies in improving profits by +29.9% compared to the DeepBike and +39.8% compared to the MIP, indicating that the hybrid strategy could improve the BSS from the financial perspective better than other strategies.

Table 5.1. The GHG emission in a week under different rebalancing policies

| | Hybrid strategy | MIP strategy | DeepBike strategy | Greedy Rebalancing | Random Rebalancing |
|---|---|---|---|---|---|
| GHG emission (reduction) from BSS (kg CO2-eq) | -1237.05 | -746.41 | -1087.07 | -374.95 | +485.98 |
| GHG emission (generation) from vehicles (kg CO2-eq) | +2390.97 | +679.46 | +2827.05 | +1069.17 | +2829.15 |

Table 5.2. The improved profits in a week under different rebalancing policies

| | Hybrid strategy | MIP strategy | DeepBike strategy | Greedy Rebalancing | Random Rebalancing |
|---|---|---|---|---|---|
| Improved profits compared to no rebalancing (dollar per week) | 5029.87 | 3598.78 | 3872.17 | 77.38 | -6238.47 |

## 5.4 Sensitive Analysis

Given the proposed hybrid strategy, the probability $\rho$ of choosing the MIP model for each vehicle could impact the performance of the hybrid strategy. Additionally, the number of vehicles in the rebalancing fleet could influence the effect of the rebalancing. This chapter conducts sensitive analysis to study how different probability of choosing the MIP model in the hybrid strategy and how different number of vehicles impact the performance of improving profits.

### 5.4.1 The impact of hybrid strategy setup

The DeepBike model and the MIP model carry out different policies in rebalancing bikes. Therefore, in the hybrid strategy where the vehicle is controlled by the MIP with probability $\rho$ and controlled by DeepBike with probability $1-\rho$, it is natural to investigate what the proper probability $\rho$ is for the hybrid strategy. The sensitive analysis in this subchapter tests different value $\rho$ ranging from 0 (pure DeepBike) to 1 (pure MIP) with a step size of 0.1, with other setups unchanged in the original model, such as the number of vehicles in the fleet $M = 10$.

Table 5.3. Varying improved profits (unit: dollar) in a week under different probability of choosing MIP policy in the hybrid strategy

| $\rho$ | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
|---|---|---|---|---|---|---|---|
| 0 | 861.44 | 1229.99 | 719.52 | 1498.54 | 880.02 | -476.46 | -840.88 |
| 0.1 | 883.81 | 1302.40 | 920.57 | 1562.47 | **907.16*** | -459.41 | -810.09 |
| 0.2 | 1070.55 | **1342.77*** | **1229.04*** | **1667.28*** | 867.31 | -400.33 | -746.74 |
| 0.3 | **1100.15*** | 1297.42 | 1063.53 | 1606.44 | 838.27 | -364.98 | -653.58 |
| 0.4 | 1066.33 | 1268.55 | 1052.41 | 1546.26 | 786.45 | -310.60 | -601.95 |
| 0.5 | 1091.64 | 1276.04 | 1037.31 | 1482.71 | 769.71 | -235.64 | -539.35 |
| 0.6 | 1064.65 | 1158.38 | 1042.12 | 1531.79 | 682.74 | -209.38 | -452.56 |
| 0.7 | 1006.37 | 1042.19 | 1079.01 | 1422.68 | 587.77 | -170.10 | -394.75 |
| 0.8 | 930.46 | 969.82 | 1118.90 | 1333.64 | 490.59 | -78.35 | -349.33 |
| 0.9 | 745.83 | 860.53 | 1156.48 | 1239.80 | 375.61 | -41.19 | -254.28 |
| 1 | 498.08 | 685.30 | 1206.95 | 1078.94 | 262.69 | **45.42*** | **-178.60*** |

Note: * marks the lowest customer loss in each day among different probability

The results in Table 5.3 indicate that, on weekdays, setting $\rho$ from 0.1 to 0.3 in the hybrid strategy generates the largest improved profit compared with setting other values of $\rho$ in the hybrid strategy; on weekends, setting $\rho = 1$ in the hybrid strategy could result in the largest improved profit compared with other values.

### 5.4.2 The impact of the number of vehicles in the fleet

In the simulator, the number of vehicles in the fleet was originally set to $M = 10$. Since the rebalancing of bikes fully relies on the vehicles, the sensitive analysis in this subchapter tests the impact of different number of vehicles for different policies in improving profits. Specifically, the number of vehicles in the fleet $M$ is set to range from 2 to 10, with a step size of 2. The performances are evaluated under the hybrid, DeepBike, and MIP strategies

Table 5.4. Varying improved profits (unit: dollar) in a week with different number of vehicles in the fleet under the hybrid policy ($\rho = 0.2$)

| Day / $M$ | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
|---|---|---|---|---|---|---|---|
| 2 | 505.80 | 632.04 | 239.90 | 600.49 | 285.47 | **-16.40*** | **-142.14*** |
| 4 | 757.34 | 891.30 | 436.67 | 1025.15 | 532.07 | -101.01 | -309.39 |
| 6 | 913.46 | 1134.53 | 687.96 | 1263.69 | 733.37 | -212.32 | -442.80 |
| 8 | 1031.18 | 1334.41 | 818.91 | 1446.55 | 856.56 | -303.97 | -580.58 |
| 10 | **1070.55*** | **1342.77*** | **1229.04*** | **1667.28*** | **867.31*** | -400.33 | -746.74 |

Table 5.5. Varying improved profits (unit: dollar) in a week with different number of vehicles in the fleet under the MIP policy

| Day / $M$ | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
|---|---|---|---|---|---|---|---|
| 2 | 58.57 | 241.27 | 232.52 | 345.44 | 53.75 | 28.79 | **-28.07*** |
| 4 | 181.64 | 347.40 | 497.03 | 451.31 | 107.71 | **68.93*** | -69.19 |
| 6 | 286.70 | 416.19 | 732.83 | 752.37 | 160.66 | 51.55 | -108.46 |
| 8 | 443.39 | 602.77 | 937.47 | 882.32 | 217.01 | 37.68 | -130.09 |
| 10 | **498.08*** | **685.30*** | **1206.95*** | **1078.94*** | **262.69*** | 45.42 | -178.60 |

Table 5.6. Varying improved profits (unit: dollar) in a week with different number of vehicles in the fleet under the DeepBike policy

| Day / $M$ | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
|---|---|---|---|---|---|---|---|
| 2 | 475.51 | 652.55 | 236.50 | 630.00 | 327.25 | **-46.23*** | **-184.25*** |
| 4 | 797.92 | 990.06 | 548.19 | 1072.69 | 632.05 | -172.64 | -390.75 |
| 6 | 921.35 | 1195.41 | 553.67 | 1392.41 | 815.67 | -284.65 | -545.83 |
| 8 | **1010.78*** | 1183.78 | **874.47*** | **1516.33*** | **955.92*** | -367.63 | -709.16 |
| 10 | 861.44 | **1229.99*** | 719.52 | 1498.54 | 880.02 | -476.46 | -840.88 |

Table 5.4 -Table *5.6* shows the different number of vehicles in the fleet under different policies can lead to different improved profits in each day. On the weekdays, especially under the hybrid and MIP policies, the more vehicles in the fleet can help improve more revenues. This is consistent with the results shown in Figure *5.5* that customer loss incurred is high when no repositioning is executed in the BSS during the weekdays. Therefore, more vehicle resources are needed to execute rebalancing. On the weekends, especially on Sunday, two vehicles in the fleet can lead to larger improved profits than having more vehicles in the fleet. This is consistent with the results shown in Figure *5.5* andFigure *5.6* that the vehicles are over-utilized on the weekends, and having more vehicles in the fleet may not always benefit the system. This sensitive analysis implies that the specific number of vehicles in fleet should be properly adjusted based on the real-time customer demand.

## 5.5 Evaluating the hybrid strategy

To better understand how the hybrid strategy leverages the benefits of both the DeepBike and the MIP strategies to generate better performance than each individual strategy, this subchapter conducts a case study about the rebalancing patterns and corresponding performances under different strategies based on the testing data on September 25, Wednesday.

For the DeepBike strategy, as demonstrated in Figure 5.9 and Figure 5.10, the agent prefers to dispatch vehicles to the downtown area where a large amount of customer loss occurs. As the agent approximates the relationship between the input system features and the Q-values of the actions based on the neural network, the agent is less sensitive to the routing cost, so it routes vehicles to the downtown area actively even though the vehicles are initialized at the suburban area. Specifically,

Table 5.7 demonstrates that the number of reduced customer loss is much greater than that could be achieved only by the bike rebalancing executed at 8am, which indicates the vehicles are assigned to rebalance stations with high customer loss in advance before morning peak hour (8 am) so that a large number of customer loss is reduced in stations at 8 am. Likewise, as shown in Table 5.8, the vehicles are assigned to rebalance the stations in the downtown area during evening peak hours (5 pm). However,

Table 5.7 and Table 5.8 show that the agent learns the rebalancing policy with different qualities for morning and evening peak hours respectively. The agent reduces the customer loss during the morning and evening peak hours at different scales, so that the stations are better rebalanced during the morning peak hours than the evening peak hours.

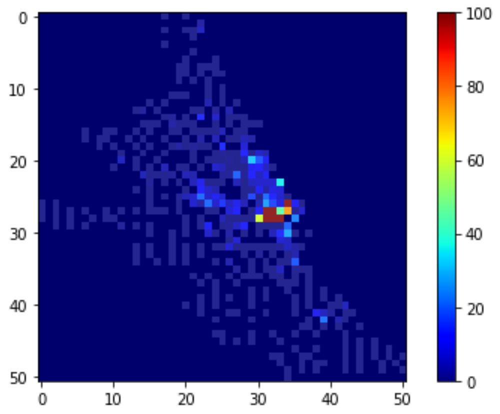


Figure 5.9 The distribution of customer loss under No rebalancing strategy on Wednesday, September 25, 2019

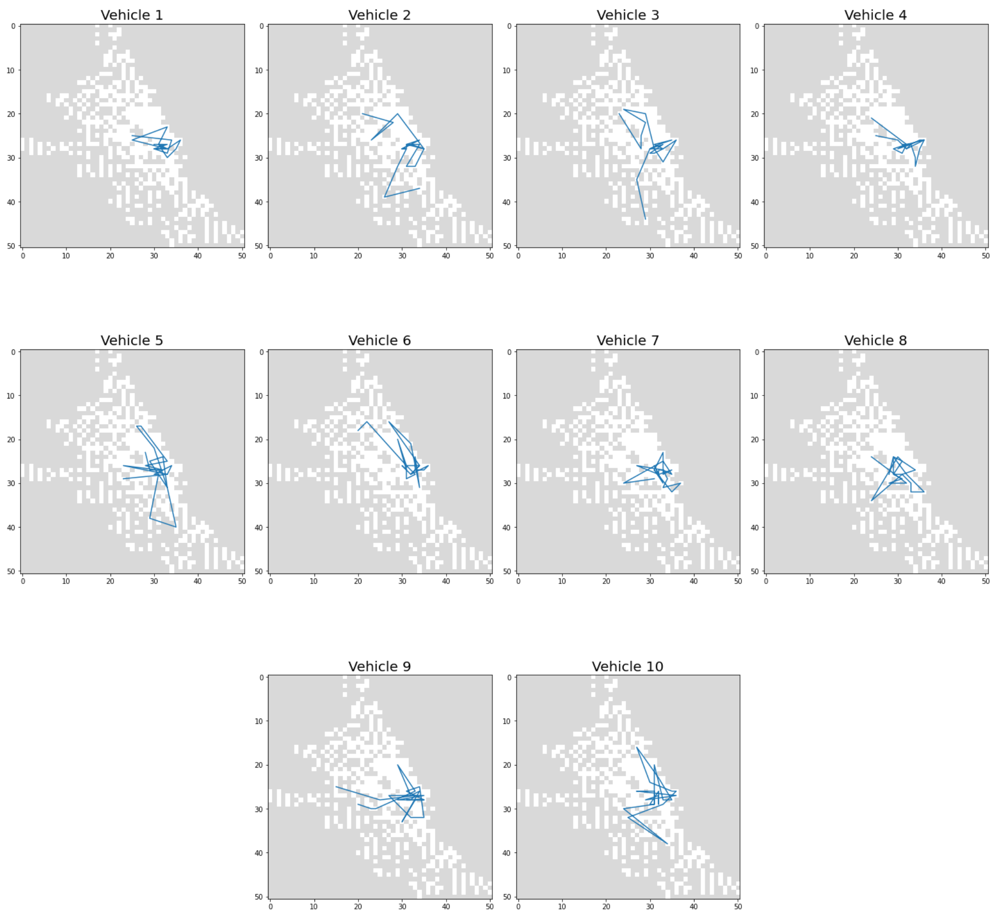


Figure 5.10. An example of routing trajectory of each vehicle under DeepBike strategy on Wednesday September 25, 2019

Table 5.7. The rebalancing decisions at 8 am under DeepBike strategy on Wednesday September 25, 2019

| Vehicle # | Visited station | Repositioning action | Effect |
|---|---|---|---|
| 1 | **27_32*** | -10 | Reduce 90 customer loss |
| 2 | **27_31*** | -15 | Reduce 60 customer loss |
| 3 | **28_33*** | -5 | Reduce 50 customer loss |
| 4 | **27_33*** | +5 | Reduce 35 customer loss |
| 5 | **28_30*** | 5 | Reduce 30 customer loss |
| 6 | **26_34*** | -5 | Reduce 23 customer loss |
| 7 | **28_32*** | 5 | Reduce 20 customer loss |
| 8 | **28_31*** | -20 | Reduce 50 customer loss |
| 9 | 27_35 | 10 | 0 |
| 10 | 29_31 | -5 | 0 |

Note: * indicates the customer loss in this station at current time step ranks top 10 among the stations, and these marked stations are also in the downtown area.

Table 5.8. The rebalancing decisions at 5 pm under DeepBike strategy on Wednesday September 25, 2019

| Vehicle # | Visited station | Repositioning action | Effect |
|---|---|---|---|
| 1 | **27_31*** | +20 | Reduce 20 customer loss |
| 2 | **27_34*** | +10 | Reduce 10 customer loss |
| 3 | **28_31*** | +15 | Reduce 15 customer loss |
| 4 | 28_35 | +10 | 0 |
| 5 | 17_26 | 0 | 0 |
| 6 | 16_27 | 0 | 0 |
| 7 | **23_33*** | -20 | Reduce 3 customer loss |
| 8 | **30_33*** | -10 | Reduce 12 customer loss |
| 9 | **26_34*** | +20 | Reduce 12 customer loss |
| 10 | 23_32 | -10 | 0 |

Note: * indicates the customer loss in this station at current time step ranks top 10 among the stations, and these marked stations are also in the downtown area.

For the MIP strategy, it is more sensitive to weigh the customer loss and the routing cost that occurs when serving to reduce the customer loss, as represented in the objective function. As demonstrated in Figure 5.11, due to the initialization of the vehicle's location at the beginning of each episode, if a vehicle is randomly assigned to the stations in the suburban area, the vehicle would not aggressively route to the downtown area to serve the high potential customer loss. Instead, the vehicle would gradually move closer to the stations in the downtown area step by step, and in each time step, the vehicle only moves in a short distance. Thus, as shown in Table 5.9 and Table 5.10, the stations in the downtown area are better rebalanced and the customer loss are reduced significantly during the evening peak hours than the morning peak hours, because a vehicle is not able to reach the downtown area to reduce lots of customer loss there during the morning peak hours.

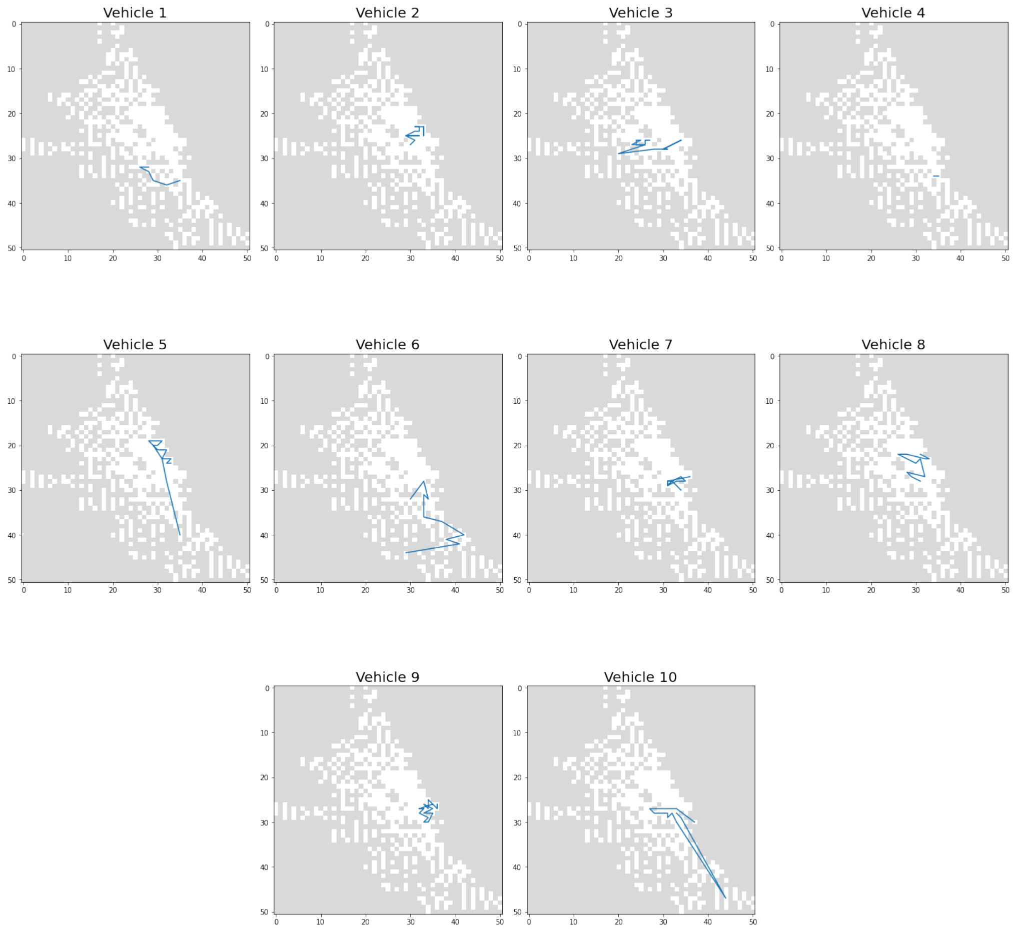


Figure 5.11. An example of routing trajectory of each vehicle under MIP strategy on Wednesday September 25, 2019

Table 5.9. The rebalancing decisions at 8 am under MIP strategy on Wednesday September 25, 2019

| Vehicle # | Visited station | Repositioning action | Effect |
|---|---|---|---|
| 1 | 36_32 | 0 | 0 |
| 2 | 24_32 | +10 | Reduce 4 customer loss |
| 3 | 26_25 | +10 | Reduce 7 customer loss |
| 4 | 34_35 | 0 | Reduce 4 customer loss |
| 5 | 20_29 | +11 | 0 |
| 6 | 40_42 | 2 | 0 |
| 7 | **28_31*** | 0[a] | Reduce 20 customer loss |
| 8 | **27_32*** | 0 | 0 |
| 9 | **28_33*** | 0 | Reduce 15 customer loss |
| 10 | 29_31 | 14 | 0 |

Note: * indicates the customer loss in this station at current time step ranks top 10 among the stations, and these marked stations are also in the downtown area.

a: there is no effective repositioning in this station at this time step. The customer loss reduction is caused by the repositioning in this station in the previous time steps.

Table 5.10. The rebalancing decisions at 5 pm under MIP strategy on Wednesday September 25, 2019

| Vehicle # | Visited station | Repositioning action | Effect |
|---|---|---|---|
| 1 | 36_32 | +6 | 0 |
| 2 | **23_33*** | -5 | Reduce 3 customer loss |
| 3 | **28_31*** | +20 | Reduce 20 customer loss |
| 4 | 34_35 | 0 | 0 |
| 5 | **28_32*** | +20 | Reduce 45 customer loss |
| 6 | **28_33*** | +20 | Reduce 68 customer loss |
| 7 | 29_31 | -20 | 0 |
| 8 | 23_32 | 0 | 0 |
| 9 | **27_32*** | +20 | Reduce 30 customer loss |
| 10 | 27_33 | 0 | Increase 2 customer loss |

Note: * indicates the customer loss in this station at current time step ranks top 10 among the stations, and these marked stations are also in the downtown area.

The vehicles following DeepBike could redistribute the bikes in the downtown area and reduce a large amount of potential customer loss during the morning peak hours while the vehicles following MIP could reduce lots of potential customer loss during the evening peak hours. Such rebalancing patterns provides the gaps to integrate the DeepBike and the MIP strategies as a hybrid one, so as to get a station better rebalanced both during the morning and evening peak hours. Moreover, due to the approximation of the neural network, not all vehicles could be dispatched exactly to the "right" stations that have large potential customer loss and need rebalancing. As indicated in Table 5.7, the rebalancing of vehicles 9 and 10 does not help reduce the customer loss at 8 am. Therefore, substituting the policy of a few vehicles (10%-30% shown in Chapter 5.4.1) by MIP policy would not significantly worsen the rebalancing performance. Figure 5.12 showcases an example of how a station with high customer loss in the downtown area is better rebalanced by the hybrid strategy compared to the DeepBike and MIP strategies. In station 27_32, the DeepBike and MIP strategies rebalance it and reduce the potential customer loss significantly during the morning and evening peak hours, respectively. The hybrid strategy rebalances this station both during the morning and evening peak hours and therefore reduces the number of customer loss much more than using only the DeepBike or only the MIP policies.

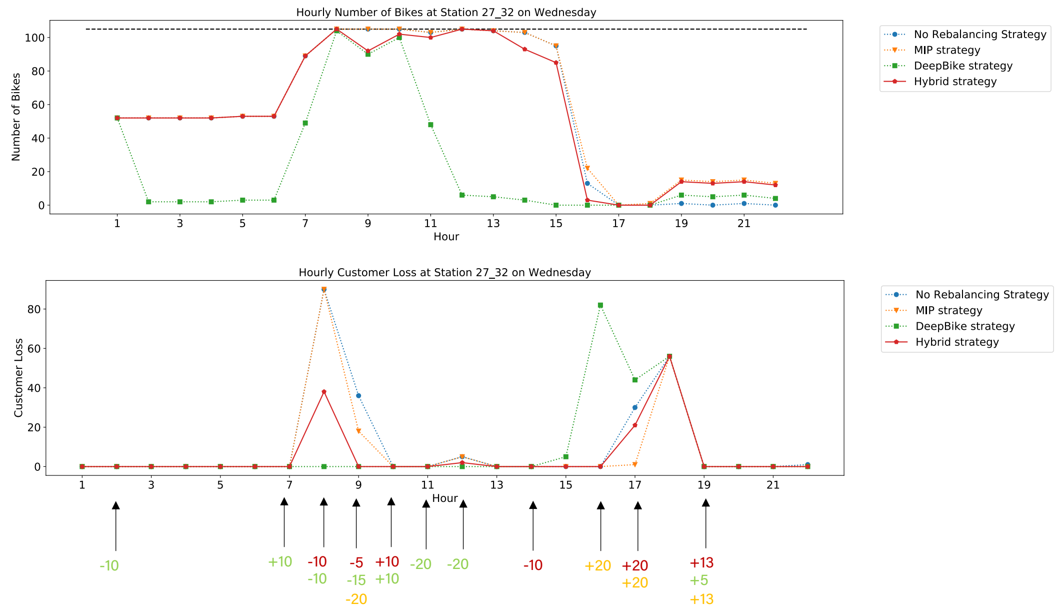


Figure 5.12. An example of the rebalancing decisions in station 27_32 and corresponding system dynamics on Wednesday September 25, 2019

# 6. DISCUSSION AND CONCLUSION

This work proposes a hybrid rebalancing strategy, leveraging the DRL model (DeepBike) and MIP model, to help improve the revenue in real-time BSS operation for the entire day. The hybrid strategy is able to generate online rebalancing decisions considering the dynamics and uncertainties of the system. DeepBike is a model-free framework that utilizes deep learning and reinforcement learning to learn the optimal policies for DBSRP for multiple steps. The agent in the DeepBike model improves its policy through iteratively interacting with the external bike sharing environment and receiving the reward signal. Based on the neural network, the agent can efficiently adapt to changing system and make distributed rebalancing action for each vehicle. The MIP approach is a model-based optimization algorithm that optimizes the objective for the whole system. While both DeepBike and MIP models are able to generate good rebalancing decisions, this work integrates these two models as a hybrid strategy to further improve the performance in reducing customer loss, while maintaining the scalability and efficiency of the policy. Under the hybrid mechanism, the pure DeepBike model and the pure MIP model can be regarded as two special cases of the hybrid policy. To train the proposed strategies and evaluate the performance of different strategies, a BSS simulator is constructed based on the real-world historical bike trip data in Divvy, Chicago. The empirical evaluation results indicate that the hybrid strategy outperforms other methods in reducing customer loss especially in weekdays. The hybrid strategy prioritizes some stations in the downtown area and redistributes the bikes among these stations to make them empty or full, in order to prepare for the large volume of upcoming user demands. The sensitive analysis verifies that a small probability of choosing the MIP model (range from 0.1 to 0.3) in the hybrid strategy could further improve the performance in reducing the customer loss compared to the DeepBike and the MIP models. Furthermore, the sensitive analysis reveals that the number of vehicles in the fleet should be adjusted in accordance with the real-time usage demand and two vehicles can reduce more customer loss than ten vehicles on Sunday.

The DRL- and MIP-based hybrid framework contributes to build a more efficient BSS from the perspective of the rebalancing operation, but has the following limitations that need to be addressed in future works to further improve the performance of the policy. First, given the fact that the historical trip data only records the observed behaviors (only the users who successfully rent and return bikes and were recorded) instead of the true usage demand (observed behaviors plus unrecorded users who leave the system when seeing stations are empty), the customer loss derived from the *net demand*-based simulation in this work only represents the observed part of the true lost users. This partial observed data will impact the policy in the realistic implementation, such as the exact number of bikes to be moved might be different because the upcoming true demand could be higher. Therefore, a true demand estimation model would benefit the simulator setup and the decision-making model. Second, only one neural network is used in the DeepBike model and is shared by all vehicle entities. Each vehicle generates rebalancing decision sequentially based on the same neural network. While ensuring the scalability of the algorithm, the rebalancing policy for each vehicle is homogeneous because one neural network actually represents only one rebalancing policy learned from the historical experience. In BSS, the vehicles could potentially be cooperative with each other to maximize the total revenues. Using the emerging multi-agent reinforcement learning (MARL) algorithms to treat each vehicle as an individual agent could help better learn the optimal policy in DBSRP that considers the collaboration of all vehicles (Zhang et al., 2019). Thirdly, our model does not study the impact of the weather and events, such as snowy days and holidays, due to having limited information on these factors. Future investigation on these factors, especially in the demand prediction model, can improve the model to better adapt to more uncertainties under different real-life scenarios (Li et al., 2015; Zhou et al., 2019). Lastly, this work does not constrain the working hours of the operators, so that the vehicle fleet routes among the stations to redistribute the bikes throughout the day. This would increase not only the fuel cost of the rebalancing but also the rebalancing emission. Moreover, the customer loss in the stations during the off-peak hours is in the lower scale compared with that during the peak hours. It is difficult for the DRL-based agent to collect rewards and learn an efficient policy during the off-peak hours. The samples collected during the off-peak hours reduce the learning efficiency of the agent. Thus, the future study that constrains the eight hours of rebalancing work could make the rebalancing policy more realistic and efficient.

In summary, since the unbalanced bike usage demands can lead to potential customer loss in BSS, an efficient rebalancing policy should be developed to put the stations back to the proper status in order to ensure the users can rent or return bikes at the target stations. This study proposes a hybrid rebalancing policy that integrates a DRL-based model-free algorithm and a

MIP-based model-based algorithm. Leveraging the benefits of two algorithms, this hybrid policy is scalable and efficient to generate online rebalancing decision for each vehicle separately in the real-time BSS for the entire day. The evaluation results show the improved performance the hybrid strategy in improving the profits compared to the pure DRL-based algorithm and pure MIP-based algorithm.

# REFERENCES

Al-Abbasi, A. O., Ghosh, A., & Aggarwal, V. (2019). DeepPool: Distributed Model-Free Algorithm for Ride-Sharing Using Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems*, *20*(12), 4714–4727. https://doi.org/10.1109/TITS.2019.2931830

Chemla, D., Meunier, F., & Wolfler Calvo, R. (2013). Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, *10*(2), 120–146. https://doi.org/10.1016/j.disopt.2012.11.005

Chen, J., Umrawal, A. K., Lan, T., & Aggarwal, V. (2021). DeepFreight: A Model-free Deep-reinforcement-learning-based Algorithm for Multi-transfer Freight Delivery. *ArXiv Preprint ArXiv:2103.03450*.

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

Contardo, C., Morency, C., & Rousseau, L.-M. (2012). Balancing a dynamic public bike-sharing system. *Cirrelt*. https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2012-09.pdf

Degris, T., Pilarski, P. M., & Sutton, R. S. (2012). Model-Free reinforcement learning with continuous action in practice. *Proceedings of the American Control Conference*, 2177–2182. https://doi.org/10.1109/acc.2012.6315022

Dell'Amico, M., Hadjicostantinou, E., Iori, M., & Novellani, S. (2014). The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega (United Kingdom)*, *45*, 7–19. https://doi.org/10.1016/j.omega.2013.12.001

DeMaio, P. (2009). Bike-sharing: History, Impacts, Models of Provision, and Future. *Journal of Public Transportation*, *12*(4), 41–56. https://doi.org/10.5038/2375-0901.12.4.3

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

Fricker, C., & Gast, N. (2016). Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO Journal on Transportation and Logistics*, *5*(3), 261–291. https://doi.org/10.1007/s13676-014-0053-5

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

Golden, B. L., Wasil, E. A., Kelly, J. P., & Chao, I.-M. (1998). The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. In *Fleet Management and Logistics* (pp. 33–56). Springer. https://doi.org/10.1007/978-1-4615-5755-5_2

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, Issue 2). MIT press Cambridge.

Hasselt, H. (2010). Double Q-learning. *Advances in Neural Information Processing Systems*, *23*, 2613–2621.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Deepmind, D. S. (2018). Rainbow: Combining Improvements in DQN. *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 3215–3222. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewFile/17204/16680

IRS. (n.d.). *Standard Mileage Rates*. https://www.irs.gov/tax-professionals/standard-mileage-rates

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kou, Z., & Cai, H. (2019). Understanding bike sharing travel patterns: An analysis of trip data from eight cities. *Physica A: Statistical Mechanics and Its Applications*, *515*, 785–797. https://doi.org/10.1016/j.physa.2018.09.123

Kou, Z., Wang, X., Chiu, S. F. (Anthony), & Cai, H. (2020). Quantifying greenhouse gas emissions reduction from bike share systems: a model considering real-world trips and transportation mode choice patterns. *Resources, Conservation and Recycling*, *153*(May 2019), 104534. https://doi.org/10.1016/j.resconrec.2019.104534

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*(2), 231–247. https://doi.org/10.1016/0377-2217(92)90138-Y

Legros, B. (2019). Dynamic repositioning strategy in a bike-sharing system; how to prioritize and how to rebalance a bike station. *European Journal of Operational Research*, *272*(2), 740–753. https://doi.org/10.1016/j.ejor.2018.06.051

Li, Y., Zheng, Y., & Yang, Q. (2018a). 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

Li, Y., Zheng, Y., & Yang, Q. (2018b). 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

Li, Y., Zheng, Y., Zhang, H., & Chen, L. (2015). Traffic prediction in a bike-sharing system. *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, *03-06-Nove*. https://doi.org/10.1145/2820783.2820837

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*(3–4), 293–321.

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., Kou, Z., Zhao, F., & Cai, H. (2019). Comparative life cycle assessment of station-based and dock-less bike sharing systems. *Resources, Conservation and Recycling*, *146*(April), 180–189. https://doi.org/10.1016/j.resconrec.2019.03.003

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

Meddin, R., & DeMaio, P. J. (2020). *The Meddin Bike-sharing World Map*. Google Maps. https://bikesharingworldmap.com/#/all/2.3/8.06/54.59/%0Ahttps://bikesharingworldmap.com/#/all/2.3/-1.57/33.92/%0Ahttps://bikesharingworldmap.com/#/all/6.9/-72.01/19.73/

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*. 1–9. http://arxiv.org/abs/1312.5602

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

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.

Oda, T., & Joe-Wong, C. (2018). MOVI: A Model-Free Approach to Dynamic Fleet Management. *Proceedings - IEEE INFOCOM*, *2018-April*, 2708–2716. https://doi.org/10.1109/INFOCOM.2018.8485988

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.

Raviv, T., Tzur, M., & Forma, I. A. (2013). Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, *2*(3), 187–229. https://doi.org/10.1007/s13676-012-0017-6

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

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

Thrun, S. B. (1992). *Efficient exploration in reinforcement learning*.

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.

Watkins, Christopher J C H, & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3–4), 279–292.

Watkins, Cristopher John Cornish Hellaby. (n.d.). *Learning from Delayed Rewards*.

Xing, Y., Wang, K., & Lu, J. J. (2020). Exploring travel patterns and trip purposes of dockless bike-sharing by analyzing massive bike-sharing data in Shanghai, China. *Journal of Transport Geography*, *87*(June), 102787. https://doi.org/10.1016/j.jtrangeo.2020.102787

Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H., Yang, S., Wu, X., Guo, Q., Chen, Q., Yin, Y., Zhang, H., Shi, T., Wang, L., Fu, Q., Yang, W., & Huang, L. (2019). *Mastering Complex Control in MOBA Games with Deep Reinforcement Learning*. http://arxiv.org/abs/1912.09729

Zhang, K., Yang, Z., & Başar, T. (2019). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *ArXiv*, 1–72.

Zhou, Y., Chen, H., Li, J., Wu, Y., Wu, J., & Chen, L. (2019). Large-scale station-level crowd flow forecast with ST-UnET. *ISPRS International Journal of Geo-Information*, *8*(3). https://doi.org/10.3390/ijgi8030140