Efficient bike-sharing repositioning with cooperative multi-agent deep reinforcement learning

Yao Jing, Bin Guo, Yan Liu, Daqing Zhang, Djamal Zeghlache, Zhiwen Yu

▶ **To cite this version:**

Yao Jing, Bin Guo, Yan Liu, Daqing Zhang, Djamal Zeghlache, et al.. Efficient bike-sharing repositioning with cooperative multi-agent deep reinforcement learning. ACM Transactions on Sensor Networks, 2024, 10.1145/3639468 . hal-04392255

HAL Id: hal-04392255
https://hal.science/hal-04392255v1

Submitted on 17 Jan 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons CC0 - Public Domain Dedication 4.0 International License

Check for updates

Efficient Bike-sharing Repositioning with Cooperative Multi-Agent Deep Reinforcement Learning

YAO JING, Northwestern Polytechnical University, China
BIN GUO*, Northwestern Polytechnical University, China
YAN LIU, Peking University, China
DAQING ZHANG, Télécom SudParis, France
DJAMAL ZEGHLACHE, Télécom SudParis, France
ZHIWEN YU, Harbin Engineering University; Northwestern Polytechnical University, China

As an emerging mobility-on-demand service, bike-sharing system (BSS) has spread all over the world by providing a flexible, cost-efficient, and environment-friendly transportation mode for citizens. Demand-supply unbalance is one of the main challenges in BSS because of the inefficiency of the existing bike repositioning strategy, which reallocates bikes according to a pre-defined periodic schedule without considering the highly dynamic user demands. While reinforcement learning has been used in some repositioning problems for mitigating demand-supply unbalance, there are significant barriers when extending it to BSS due to the dimension curse of action space resulting from the dynamic number of workers and bikes in the city. In this paper, we study these barriers and address them by proposing a novel bike repositioning system, namely BikeBrain, which consists of a demand prediction model and a spatio-temporal bike repositioning algorithm. Specifically, to obtain accurate and real-time usage demand for efficient bike repositioning, we first present a prediction model ST-NetPre, which directly predicts user demand considering the highly dynamic spatio-temporal characteristics. Furthermore, we propose a spatio-temporal cooperative multi-agent reinforcement learning method (ST-CBR) for learning the worker-based bike repositioning strategy in which each worker in BSS is considered an agent. Especially, ST-CBR adopts the centralized learning and decentralized execution way to achieve effective cooperation among large-scale dynamic agents based on Mean Field Reinforcement Learning (MFRL), while avoiding the huge dimension of action space. For dynamic action space, ST-CBR utilizes a SoftMax selector to select the specific action. Meanwhile, for the benefits and costs of agents' operation, an efficient reward function is designed to seek an optimal control policy considering both immediate and future rewards. Extensive experiments are conducted based on large-scale real-world datasets, and the results have shown significant improvements of our proposed method over several state-of-the-art baselines on the demand-supply gap and operation cost measures.

CCS Concepts: • **Computing methodologies** → **Planning and scheduling**; **Multi-agent planning**.

Additional Key Words and Phrases: Deep reinforcement learning, Multi-agent Reinforcement Learning, Bike-Sharing System, bike repositioning

Authors' addresses: Yao Jing, jy_jingyao@163.com, Northwestern Polytechnical University, Xi'an, China, 710129; Bin Guo, guob@nwpu.edu.cn, Northwestern Polytechnical University, Xi'an, China; Yan Liu, 253004847@qq.com, Peking University, Beijing, China; Daqing Zhang, daqing.zhang@telecom-sudparis.eu, Télécom SudParis, Paris, France; Djamal Zeghlache, djamal.zeghlache@telecom-sudparis.eu, Télécom SudParis, Paris, France; Zhiwen Yu, zhiwenyu@nwpu.edu.cn, Harbin Engineering University; Northwestern Polytechnical University, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1550-4859/2024/1-ART
https://doi.org/10.1145/3639468

Just Accepted

1 INTRODUCTION

Bike-sharing system (BSS), as an emerging mobility-on-demand service, has spread all over the world, including docked systems like the Citi bike in New York City[1], Capital bike in Washington[2], and dockless systems like Hellobike[3], DiDi Bike[4] in China, etc. According to the survey by Bike-sharing Blog[5], as of May 2018, there are more than 1,600 BSS companies in operation around the world, providing a total of more than 18 million bikes for people to use, and this number is still growing. BSS not only bridges the gap between traditional public transport hubs (e.g., metro, bus station, and railway station) and the origin/destination, but also provides a flexible, cost-efficient, and environment-friendly transportation mode for citizens. One of the critical issues in BSS is how to balance the demand and supply. Bike repositioning is a cost-effective way to reduce the demand-supply gap, by redistributing bikes between stations in advance according to the prediction of user demand, so as to maximize the satisfaction of future coming demand. If the superfluous bikes of a station are diverted to a station with strong rent demand, the number of riders served will be significantly increased. As a result, both user satisfaction and platform revenue will improve.

Bike repositioning is a typical sequential decision-making problem. In this scenario, decisions need to be made sequentially over time to optimally redistribute bikes among different stations. The objective is to balance supply and demand, maximize user satisfaction, and minimize the repositioning costs. Combinatorial optimization algorithms (e.g., integer programming, linear programming, and genetic algorithms) are usually used to solve this problem[13, 14], where the goal is to find the best solution from a finite set of possible repositioning solutions. These algorithms help in optimizing an objective function by considering various combinations of elements and selecting the most optimal one. However, combinatorial optimization methods do have certain limitations when it comes to large-scale dynamic spatio-temporal data and real-time decision-making. More importantly, they are myopic, only focusing on finding the current optimal solution, without considering the future implications of the decision. This always leads to sub-optimal decisions.

To address these limitations, deep reinforcement learning (DRL) provides a more flexible and adaptive way to tackle the dynamic and complex bike repositioning problem. DRL is a combination of deep learning and reinforcement learning techniques, aimed at solving complex sequential decision-making problems. It enables agents to learn optimal decision-making policies through interactions with their environment while considering high-dimensional state and action spaces. Benefiting from the power of DRL that can take both short- and long-term sequential decision impacts [19, 31] into consideration, DRL performs well compared to other traditional methods. In recent years, DRL has made significant progress in the field of games playing (e.g., AlphaGo [29] and its variants [35, 43]), robotics[1, 22, 34], autonomous vehicles[2, 16], and resource allocation[45, 52]. In light of such advances, several works have tried to apply DRL to solve the bike repositioning problem in BSS [19, 31]. Li et al. [19] propose a multi-agent reinforcement learning model for bike repositioning, aiming to learn the optimal worker-based repositioning policy, which closely aligns with our work. However, this multi-agent approach considers each worker as a separate agent who learns and acts independently while interacting with the environment, their simplified problem formulation and agent settings do not adequately capture the intricate multi-agent interactions and cooperation within bike-sharing systems, resulting in poor performance.

To learn highly efficient repositioning policy for the real-time BSS platform, we summarized the significant technical challenges as follows:

[1]https://citibikenyc.com/
[2]https://capitalbikeshare.com/
[3]https://www.hello-inc.com/
[4]https://www.didiglobal.com/travel-service/bike
[5]http://bike-sharing.blogspot.com/

Just Accepted

- ***Dynamic demand prediction in BSS.*** Accurate prediction in advance is essential for subsequent DRL-based decision-making. Even with abundant historical bike usage data, predicting user demand is very challenging since bike rent and return patterns are very dynamic and influenced by many complex factors, including temporal dependencies (e.g., hour of a day, day of a week, holiday), spatial correlations (e.g., location, surrounding POIs), even the mixture of these two and other contextual factors (e.g., weather). How to capture the complex spatio-temporal factor and predict user demand for the next coming period in such a highly complex and dynamic environment is not easy.
- ***Large-scale agents.*** To avoid the dimension curse of action space under a multi-agent setting, each worker is considered as an independent agent. However, such a setting needs to maintain a large number of agents interacting with the environment simultaneously, resulting in a non-stationary environment. How to train large-scale agents and enable them to cooperatively learn efficient repositioning policy is very challenging in multi-agent reinforcement learning (MARL).
- ***Dynamic agents & action sets.*** Traditional MARL models require a fixed number of agents. However, the number of agents in BSS always changes over time as the status of the agent can be switched between online and offline depending on whether the repositioning task is completed or not. Moreover, if we model moving a certain number of bikes from a full station to a hungry station, there is no guarantee of a fixed action space as the the status of station keeps changing. How to model cooperation among a dynamic number of agents is challenging, and how to deal with variable actions in MARL is also difficult.
- ***Immediate & Future Rewards.*** RL is a reward-driven learning method, the policy determines the sequence of actions only by analyzing the reward signal from the environment. A key challenge in seeking an optimal control policy is to find a trade-off between immediate and future rewards. In the bike repositioning task, we need to consider not only meeting more customer needs but also workers' operation costs, i.e., the repositioning distance from one station to another. Hence, how to carefully design the reward function in bike repositioning scenarios is challenging.

Specifically, the main contributions of this paper are summarized as follows.

- We propose a novel bike repositioning system, namely BikeBrain, which consists of a demand prediction model and a spatio-temporal bike repositioning algorithm, for addressing demand-supply unbalance.
- We propose ST-NetPre, a spatio-temporal net demand prediction model. For the bike repositioning task, compared to solely predicting the rent demand and return demand, we focus more on directly predicting the net demand (i.e., the difference between return the rent demand, net = return- rent), which avoids the superposition of twice prediction errors.
- We propose ST-CBR, a spatio-temporal cooperative multi-agent reinforcement learning system for learning the optimal bike repositioning strategy to minimize the gap between user demand and supply at a low operation cost. Being more reliable than existing approaches, our proposed ST-CBR solution follows a centralized learning and decentralized execution way, which not only can effectively capture large-scale spatio-temporal demand-supply dynamics but also can achieve cooperation among a dynamic number of agents through Mean Field Reinforcement Learning (MFRL).
- We design a SoftMax selector and a reward function to enable efficient bike-sharing repositioning. A SoftMax selector is utilized to deal with the dynamic action space by selecting the specific action, and a reward function is leveraged to seek an optimal control policy to meet more customer needs at low worker operation costs.
- Extensive experiments are implemented using large-scale real-world bike usage records datasets from a typical bike-sharing system, i.e., Citi Bike in NYC) and other multi-source city datasets. Results have shown that our approach outperforms the state-of-the-art baselines on both the demand-supply gap and operation cost measures.

Just Accepted

The rest of this paper is organized as follows: We'll start with the related work in Section 2. Section 3 gives a high-level overview of our method. The proposed approach is further elaborated in Section 4. Section 5 displays the experiment settings and results. The discussion is introduced in Section 6, and the conclusion is enclosed in Section 7.

2 RELATED WORK

We summarized two types of related works, bike repositioning and multi-agent reinforcement learning (MARL).

Bike repositioning.

Existing bike repositioning studies can be divided into two categories: user-based [12, 31, 36] and worker-based [14, 19, 25]. The user-based way [12, 31, 36] encourages customers to rent or return bikes at certain locations in exchange for a monetary benefit, thereby facilitating the redistribution of bikes. Worker-based repositioning needs to hire some specialized workers to reallocate bikes by trikes. Most of the studies focus on the worker-based way, as it is more efficient and manageable in practice. From the perspective of the repositioning time, worker-based works can also be divided into two categories, static bike repositioning [23] and dynamic repositioning [13, 14, 19]. In static bike repositioning, workers reallocate bikes when the system does not operate or at midnight. These works focus on using rule-based combinatorial optimization algorithms to get periodic schedules for bike repositioning [23]. Combinatorial optimization algorithms, such as integer programming, linear programming, and genetic algorithms, are commonly employed to address the bike-sharing repositioning problem by identifying the most optimal solution from a finite set of possibilities. These techniques optimize objective functions by evaluating various element combinations and selecting the most suitable one. However, they possess certain limitations when dealing with large-scale, dynamic spatio-temporal data and real-time decision-making. Most notably, these methods are myopic, concentrating solely on determining the current optimal solution without considering the future consequences of their decisions. This often results in sub-optimal choices. Dynamic repositioning problems can also be solved by optimization models [13, 14], but they are hard to find the optimal solution and cannot deal with the uncertainties that arise during the actual operation process. Some studies forecast future bike demand and relocate bikes in a greedy or heuristic policy, which is not a smart approach because it is myopic and may not be optimal over time.

This survey[5] serves as a valuable resource for our proposed research, as it not only offers an overview of the state-of-the-art research in data-driven optimization for bike-sharing systems but also highlights potential opportunities for future research and enhancement. Specifically, it outlines the primary challenges and presents a comprehensive technical framework for BSS optimization. Distinct from the methods summarized by this survey, we employ DRL to learn dynamic bike repositioning strategy considering both short- and long-term impacts [19, 31]. Instead of relying on workers to reposition bikes, Pan et al.[31] aims to incentivize users themselves to rebalance the bike-sharing system by offering them rewards. They propose a DRL framework that decides how to pay different users at each time to incentivize users to rebalance the dockless bike-sharing system. Li et al. [17] investigate the application of policy-gradient-based reinforcement learning to address the rebalancing problem. Their approach aims to enhance user experience while reducing operational expenses simultaneously. Chen et al.[4] focus on Dockless Public Bicycle-sharing Systems (DBSS), which allow users to rent and return bikes without fixed docking stations, and they propose a multi-objective reinforcement learning method to optimize dispatching by considering costs, truck loads, workload balance, and supply-demand balance. Li et al.[18] propose a multi-agent reinforcement learning method for dynamic bike repositioning in DBSS. They employ the double deep q network (DDQN) algorithm and the shadow environment trick to address the non-stationary learning problem, and the training process is decomposed into sequential independent single-agent training. Li et al.[19] present a novel spatio-temporal reinforcement learning approach to address the dynamic bike repositioning problem, which is very close to our work. However, this multi-agent approach considers each worker as a separate

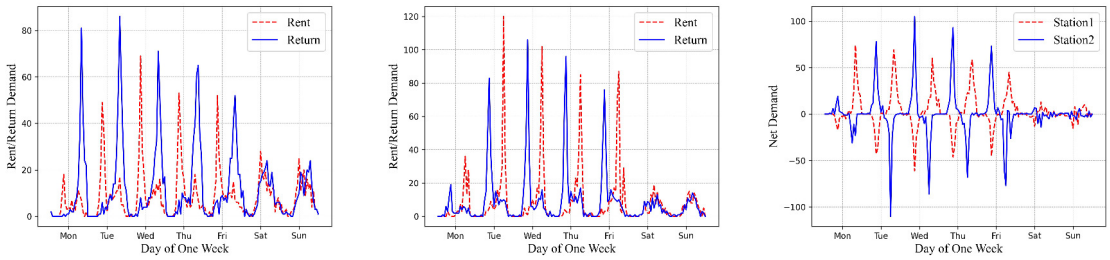
Just Accepted

agent who learns and acts independently while interacting with the environment. Their approach is limited to accurately modeling the complex interactions and cooperation of agents in BSS. Therefore, their model may not be able to effectively capture the complexities of real-world bike-sharing operations and decision-making processes, and further research and development are needed to address these challenges.

Multi-agent reinforcement learning (MARL). In recent years, deep learning has achieved great success in various fields [50, 51]. By leveraging the power of deep learning to approximate complex functions and the ability of reinforcement learning to learn optimal policies through interactions with the environment, DRL achieved state-of-the-art performance in various domains. The success of AlphaGo [29], AlphaGo Zero [35], and AlphaStar [43] demonstrates the effectiveness of DRL [8, 28, 42, 46, 48, 49]. Different from the traditional DRL setting, MARL means there is a group of agents in the environment. The main challenges in MARL are the non-stationarity between independent agents [40, 41] and the exponential increase in state and action space. Training multi-agent using independent DRL is a straightforward method [40, 41], where agents interact with the environment autonomously and take action based on their own observations and policies. However, this is not a good solution, because all of the agents are simultaneously learning and influencing the environment, making the environment non-stationary. It's difficult for agents to learn useful knowledge in such an unstable environment. Most of the existing studies use multi-agent centralized learning to enable cooperation among agents, which learns the joint policy by agents' joint state and action [15]. Unfortunately, the joint state and action grow exponentially as the number of agents increases. Even applying it to a small-scale problem could already be computationally intensive. And, the centralized learning setting is inadequate in real-world scenarios, because the joint policy is always subject to a synchronous constraint of all agents, whereas the agents are usually asynchronous in practice. Furthermore, as the status of the agent can be switched between online and offline depending on whether the repositioning task is completed or not, the number of agents in BSS always changes over time, which is challenging for traditional MARL approaches in such a dynamic environment. Moreover, as the number of bikes at each station is constantly changing, it is not fixed whether the status of the station will be too full or hungry, and how many bikes need to be moved from a full station to a hungry station is also variable. As a result, the action space for the repositioning task is dynamic. It is difficult to deal with variable actions in MARL.

Existing studies have proposed several approaches to mitigate the effects of these challenges, e.g., the policy parameters sharing [15], value-decomposition Networks [39], monotonic value function factorization [33]. A subset of these methods follows the "centralized planning with decentralized execution way [11, 26, 32]. A typical method, i.e., MADDPG [26] extends DDPG [27] into a multi-agent setting. To enhance learning, the critic is not only fed the agent's own information, but also the actions of other agents to learn the centralized Q-function during the training phase. After the training, agents can directly use the learned policy (the actor) to select actions based on their own observations. Though MARL has made significant progress in various challenging applications, e.g., games [26, 39], in complex real-world urban scenarios, MARL is rarely used due to the complexity of the urban decision-making problem. [20] use reinforcement learning to learn the courier management policy in express systems. For an electric carsharing system, [44] propose a repositioning and charging management method based on RL. Lin et al. solve the fleet management problem through MARL. Moreover, most of these approaches have limitations for the problems with large-scale agents, dynamic number of agents, and variable actions. To address this, Yang et al.[47] introduce a new method that combines MARL with mean field approximations. To verify the effectiveness of this way, they also proved its convergence by theoretical analysis. Li et al.[20] address the large-scale order dispatching problem using mean field approximations. These methods, however, cannot be directly applied to our bike-sharing repositioning task.

Just Accepted



(a) Station 1. (b) Station 2. (c) Station 1 & Station 2.

Fig. 1. Bike usage patterns of different stations in one week.

3 DATA ANALYSIS AND FRAMEWORK DEVELOPMENT

3.1 Preliminary

Definition 1: Time slot. We split the whole day into a sequence of small time slots, each of which represents a fixed time duration, e.g., one hour, denoted by t.

Definition 2: Station capacity. c_i is one of the stations in the city ($c_i \in C$). Station capacity represents the number of total docks at each station, denoted by $|c_i|$.

Definition 3: Station status. The station status indicates how many bikes and empty docks are available at each station. Let $c_i^{b(t)}$ and $c_i^{d(t)}$ represent the number of available bikes and available docks of station c_i respectively during time slot t.

Definition 4: Station bike demand. Let $c_i^{p(t)}$ and $c_i^{r(t)}$ represent the number of rents and returns of station c_i during time slot t. To better understand the unbalance of the station, let $c_i^{n(t)}$ denote the net demands, which is defined as the number of returns minus the number of rents in a time slot t:

$$c_i^{n(t)} = c_i^{r(t)} - c_i^{p(t)}. \tag{1}$$

3.2 Data-Driven Investigation

We initially performed extensive investigations and analyses based on a real-world dataset to better understand the critical issues in forecasting user demand and learning scheduling strategies. In a typical bike-sharing system, there are five parties involved, i.e., the operator, users, bikes, docks, and service stations. The operator can monitor the real-time status of BSS and record all the users' behaviors on the platform. When a user rents or returns a bike via the mobile application, there is a bike usage record generated and uploaded to the platform, meanwhile, the status of the station is updated.

In this research, we use the dataset from Citi Bike, a real-world bike-sharing system in New York City. The dataset includes 1-year bike usage record data, from January to December 2016, including a complete list of all users' bike usage records. The details of the order are contained in each record, e.g., the bike ID, the start and end station ID, longitude, latitude, timestamp, trip duration, and user profiles. We conduct a comprehensive data study based on the rich historical dataset to better understand the research challenges existing in the current BSS and to motivate our solution. The details are as follows:

Highly dynamic bike usage patterns. Fig. 1 displays the bike rent and return trends of several stations over the course of a week. In particular, we found there are two important factors influencing bike usage. (i) Time factors. The daily rent and return patterns fluctuate largely in different hours on different days. In one day, the bike usage patterns have a distinct characteristic of morning and evening peaks. For example, as shown in Fig.

Just Accepted

1(a), the number of rents rises at the start of the day and peaks in the morning rush hours (e.g., 06:00-10:00). The number of returns peaks in the evening rush hours (e.g., 16:00-20:00). In a week, the patterns on weekdays are different from that on weekends. There are more rent and return demands on weekdays than on weekends. It is easy to understand that users usually need to go to work or go home at that time. Moreover, bike usage patterns may differ on some special holidays. (ii) Spatial factors. Bike usage patterns may be extremely different at different locations. From 1(a), we can see that the rent demands are larger than returns during morning rush hours at station1, the rent demands are much lower than returns during the evening rush hours. However, as shown in 1(b), the patterns of station2 are different from station1. We display the net demand distributions in 1(c) to understand more about the unbalance in bike usage trends between these two stations. It is clear that the pattern of station 2 is almost the exact opposite of station 1 in the same time period. Station 1 needs more bikes in the morning, while station 2 needs more empty docks. That's because the stations are located in different functional areas. Station 1 is in the residential area, while station 2 is in the CBD district.

Unbalanced Spatio-temporal Bike Usage Patterns. The fine-grained rent and return distributions for various bike stations at different times of the day are investigated further. We use the net demand to represent the unbalanced demand of a station. As shown in Fig. 2, a circle demonstrates the net demand of a station, where a red one means the net demand is positive (i.e., the return demand is higher than rent at this station), and blue circle means the negative net demand (i.e., the return demand is lower than rent). The absolute value of net demand is shown by the size of each circle. Fig. 2 shows the bike net demand distributions during different time periods of one day (e.g., 06:00-10:00, 11:00-15:00, 16:00-20:00 and 21:00-23:00). Ideally, the net demand of a self-balanced station will close to zero. However, as can be seen, the majority of stations in NYC are far from being balanced, particularly during the morning (e.g., 6:00-10:00) and evening rush hours (e.g., 16:00-20:00). The station demand distribution is uneven both geographically and temporally. For example, there are more rent demands in residential regions and more returns in CBD areas during the hours of 06:00-10:00. If some stations have large continuous positive bike demands or negative bike demands, the bike demand distributions will become seriously unbalanced, which will even cause outage event, e.g., the station becomes empty or full. In order to satisfy users' rent and return needs as much as possible, each bike service station should have enough shared bikes for customers to pick up and sufficient empty docks for users to return bikes.

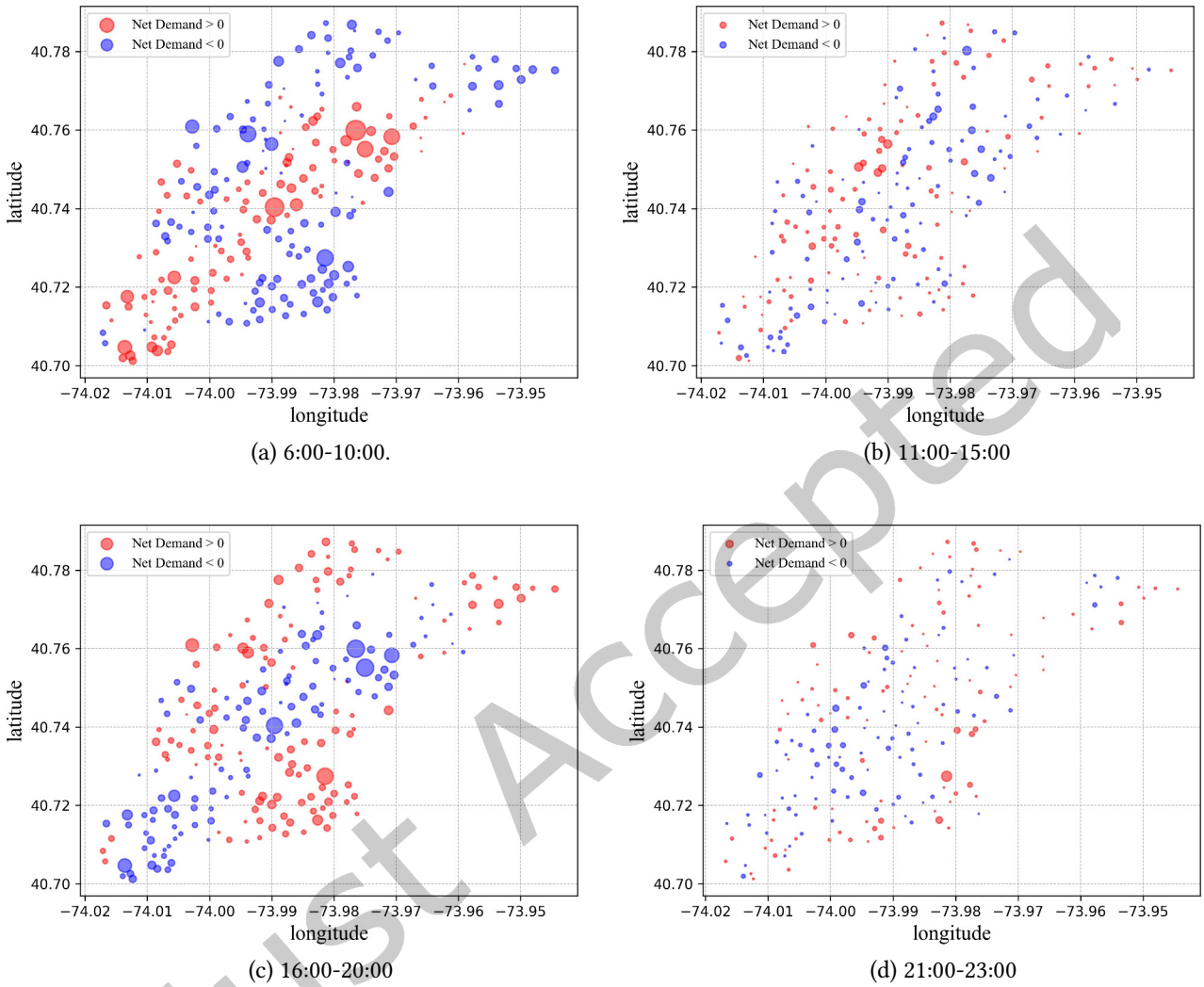
Contextual factors related to user behaviors. In common sense, the frequency of bike usage is related to the weather, temperature, etc., since it is not convenient for users to ride in bad weather. In order to understand the correlation between these contextual factors, (e.g., weather, temperature, wind, etc.) and user rent frequency, we use the Person Correlation Coefficient (PCC) to evaluate it. Fig. 3 shows the results. We can see that all these factors will influence the users' behavior, especially temperature. In demand forecasting, we should take these factors into account.

Ideally, if there are infinite shared bikes and parking docks launched at each service station, users' rent and return demands are easy to meet. However, it is not realistic, as such a large number of bikes and docks increase will incur huge costs for BSS operators. Extra repositioning efforts are necessary to better meet future user demand. To achieve this, forecasting future user demand is the primary task, and then learning an optimal strategy to reposition bikes between service stations to bridge the demand-supply gap. We discovered that the bike usage patterns are very dynamic and uneven, influenced by a variety of spatial and temporal factors, which makes it difficult to forecast user demand and design an efficient bike repositioning system.

3.3 Framework

The framework of BikeBrain is given in Fig. 4, which mainly includes offline prediction, offline repositioning, and online repositioning.

Just Accepted



(a) 6:00-10:00. (b) 11:00-15:00

(c) 16:00-20:00 (d) 21:00-23:00

Fig. 2. Spatio-temporal unbalanced bike usage patterns.

Just Accepted

Offline Prediction and Repositioning. To make decisions efficiently, user future demand prediction is the first key task. To deal with this, we propose a prediction model ST-NetPre. Specifically, we collect multi-source data and extract the fine-grained features related to bike usage based on our previous data investigations and analysis, mainly including time features, geographic features, and contextual features. We train ST-NetPre offline based on rich historical multi-source data and finally get a trained predictor.

To get an optimal repositioning policy, we propose ST-CBR, a cooperative multi-agent reinforcement learning method for bike repositioning. Each worker can be regarded as an agent and interact with the environment asynchronously. When a worker is idle, i.e., completing a previous repositioning task, it instantly begins a new

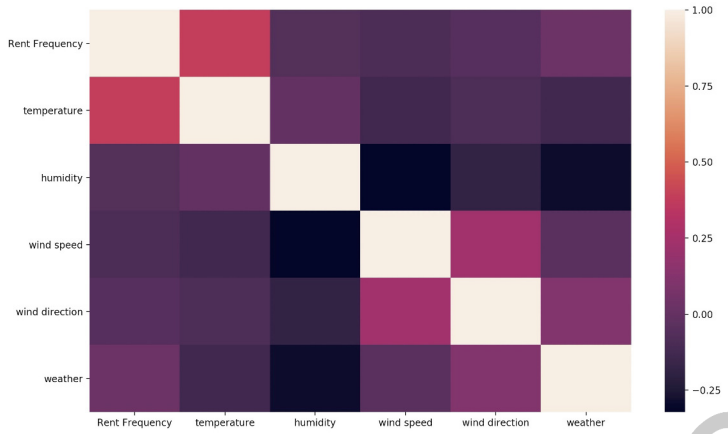


Fig. 3. Person Correlation Coefficient (PCC) between various contextual factors and rent frequency.

repositioning task generated by ST-CBR depending on its current state, without waiting for other workers. The state is carefully designed in order to reflect the BSS dynamics as well as real-time uncertainties. Specifically, the state consists of the current BSS status (e.g., current bikes and docks distribution, the status of agents, current time, etc.) and net demand distribution of the next time slot, which is predicted by the offline trained predictor ST-NetPre. After completing a task, according to the situation of users rent/return bikes, the agent gets a certain reward, which can guide the policy to be updated iteratively. We use a multi-agent deep deterministic policy gradient algorithm to estimate the value function, which allows agents to cooperate through a centralized learning and decentralized execution way and adaptively learn an optimal long-term repositioning strategy to bridge the gap between user demand and supply at low operation cost.

Online repositioning. We get a learned policy network after the offline training phase. In the online process, when an agent is idle and ready for a new repositioning task, we first generate the current status of BSS and get the predicted net demand from ST-NetPre. The learned policy network is then used to calculate the best long-term value for each feasible action under the current state. The most valuable action will be chosen and returned.

4 METHOD

In this paper, we design BikeBrain, a novel data-driven bike repositioning system for BSS to improve the user experience and operator profits (e.g., satisfy more user rent and return demands) at low operation cost (e.g., minimize the travel distance of workers moving the shared bikes between stations).

4.1 Bike demand prediction

Accurate net demand forecasting is critical because it has a direct impact on future decision-making. As indicated in the data-driven investigation, the bike usage pattern is highly dynamic in both spatial and temporal dimensions. Many studies have put effort into the bike demand prediction problem [3, 6, 7]. However, compared to solely predicting the rent demand and return demand, we focus more on the net demand. On the one hand, it is more efficient to forecast the net demand directly. On the other hand, it avoids the superposition of twice prediction errors. Hence, in this paper, we develop a spatio-temproal net demand prediction model ST-NetPre. We calculate the net flow $c_i^{n(t)}$ of a station in each time slot t based on real-world bike usage records. Furthermore, ST-NetPre takes different complex factors that are highly related to users' behaviors into consideration.

Just Accepted

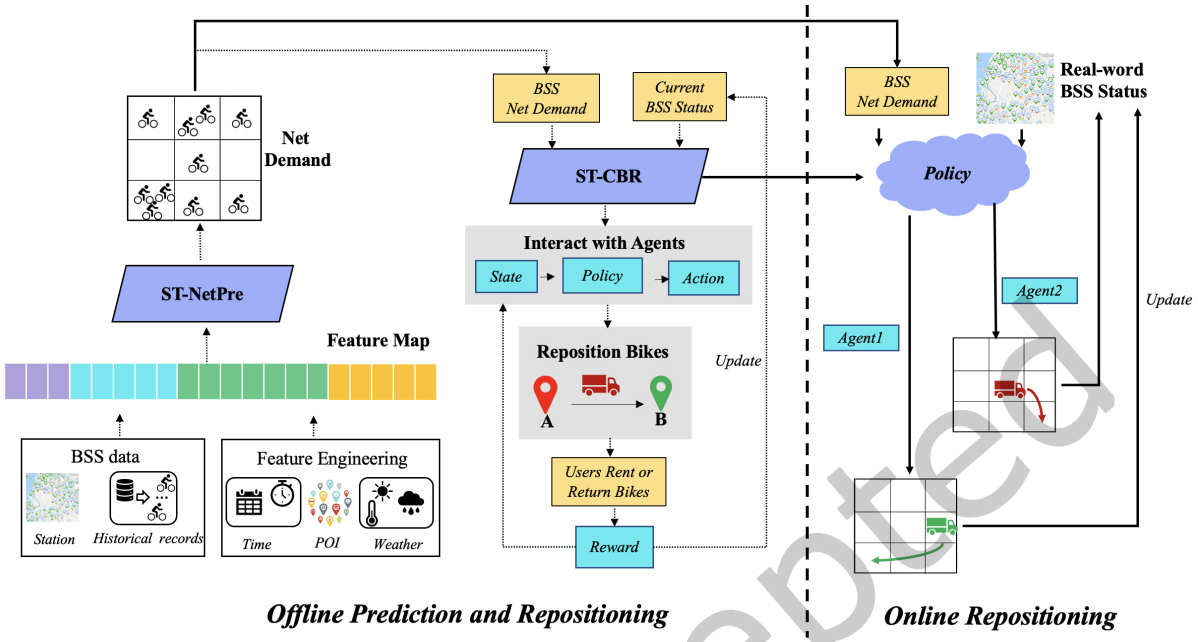


Fig. 4. Overview of BikeBrain.

4.1.1 Feature Engineering. We perform a comprehensive feature engineering and extract four types of features to predict the net demand $c_i^{n(t)}$.

Temporal Features f^T**:** We extract three temporal features, including time of a day, day of a week, and holiday [37, 38].

Spatial Features f^S**:** Assuming that a station can influence the surrounding circle area of radius 1 kilometer. For each station, we measure the spatial features of the surrounding area by analyzing the set of places that lie in the surrounding circle. From the fine-grained perspective, we choose the number of each POI category near the station as spatial features to characterize the stations. There are 30 POI categories, e.g., residence, education, medical care, park, police station, gas station, bank, library, cinema, sports center, hotel, restaurant, bar, shopping mall, supermarket, electronics store, pharmacy, boutique, clothes store, enterprise, government, conference center, church, museum, theater, entertainment place, scenic spots, parking lot, life service, transportation hub.

Contextual Features f^C**:** We also found that contextual features like weather, temperature, and wind conditions also have great impacts on users' behaviors. Hence, we collect meteorology data and extract five types of fine-grained contextual features: weather, temperature, wind speed, wind direction, and humidity.

Historical Usage Features f^H**:** Since the bike usage shows a typical weekly pattern, we utilize our long-term record data to capture the historical bike usage patterns. We extract the net demand of each station in the same time slot of four previous consecutive weeks as the historical usage features.

4.1.2 Spatio-Temporal Prediction. After identifying the related features i.e., $\mathbf{f} = [f^T, f^S, f^C, f^H]$, we then propose a ST-NetPre model to predict the net demand of each station in each time slot. Specifically, ST-NetPre is a Tree-Enhanced Regression model that utilizes the strengths of XGBoost [9] and linear regression model. XGBoost is one of the state-of-the-art models for prediction, which leverages some decision trees to learn the high-order cross features for accurate prediction. Distinct from the original XGBoost that directly sums all over the weights

Just Accepted

of activated leaf nodes as the prediction results. ST-NetPre considers the value of activated leaf nodes as new extracted cross features from XGBoost and then feeds them with raw input features together into a sparse linear regression model to learn the importance of each feature, as shown in Eq. 2

$$\hat{y_i}(\mathbf{f}_i) = \sum_j b_j(\mathbf{f}_i \oplus h_M(\mathbf{f}_i)) + b_0, \tag{2}$$

where $\mathbf{f}_i$ is the extracted four types of features, M is the number of decision trees in XGBoost, $h_M(\mathbf{f}_i)$ is all the cross features learned from $\mathbf{f}_i$ by XGBoost with M decision trees, and $\hat{y_i}$ is the prediction result. To avoid overfitting, we also use a regularization term in the loss function:

$$\mathcal{L} = \sum_{i=1}^{n} l_(y_i, \hat{y_i}) + \sum_j \Omega(b_j). \tag{3}$$

where l represents the squared loss function, Ω is the L_2 norm.

4.2 Bike repositioning as a Markov Decision Process

The bike repositioning problem can be characterized by sequential decision-making, thus we formulate the problem as a Markov Decision Process (MDP) with multi-agents and use the MARL techniques to solve it. In particular, we carefully formulate the elements in MARL. We then give an independent spatio-temporal bike repositioning algorithm (ST-IBR), and based on ST-IBR, we further propose a cooperative spatio-temporal bike repositioning algorithm (ST-CBR) based on MARL. On the one hand, ST-CBR can adaptively fulfill the highly unbalanced spatio-temporal demand based on the result of demand prediction. On the other hand, it can provide long-term repositioning benefits at low operation costs.

4.2.1 Problem Formulation. Formally, we formulate the bike repositioning task as an MDP for $\mathcal{N}$ agents, which is defined by a six-tuple $\Gamma = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{N}, \gamma)$, where $\mathcal{S}$ is the state, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward function, $\mathcal{P}$ is the transition probability function, $\mathcal{N}$ is the number of agents, and γ is the discount factor. An agent acts in the environment according to a policy that instructs it to choose the best action at each MDP state. The following are detailed definitions:

- **Agent w:** We consider each worker in the BSS as an agent w, $w \in \mathcal{W} \equiv \{1, 2, \ldots, \mathcal{N}\}$. $\mathcal{W}$ represents the set of $\mathcal{N}$ agents in the system. In particular, all the $\mathcal{N}$ agents are homogeneous in the BSS. As the workers can switch the status between busy (i.e., working on a repositioning task) and idle (i.e., finishing the last repositioning task), the number of active agents at each time slot $\mathcal{N}^t$ may fluctuate over time.
- **State $\mathcal{S}$:** At each time slot t, agent w gets an environment state $s_w^t \in \mathcal{S}$, which is denoted by the Cartesian product of a global-view environment state e^t and a local-view agent state o^t, defined as:

$$s_w^t = e^t \times o^t. \tag{4}$$

A typical global-view environment state e^t during time slot t is defined as the current BSS spatio-temporal status e_b^t plus the future net demand e_n^t:

$$e^t = e_b^t + e_n^{t+1}, \tag{5}$$

where the former e_b^t describes the distributions of bikes in the city, and the later e_n^{t+1} represents the distribution of net demands during the next time slot $t+1$, which is generated by the offline trained predictor considering the environment spatio-temporal dynamics (e.g., geographical location, time, weather conditions, etc.). In particular, o^t indicates the current local-view state of the agent w, which includes three

Just Accepted

components: the index of agent w, current location of agent loc_w and the time slot index t:

$$o^t = [w,\ loc_w,\ t]\,. \tag{6}$$

- **Action space $\mathcal{A}$:** In our bike repositioning task, an action means a complete repositioning process by the agent, which is defined as a tuple, describing where the agent should go to pick up how many bikes, and where it should go to drop off these bikes. An action a_w^t is defined as:

$$a_w^t = \left(c_i, c_j, m\right), \tag{7}$$

It means the agent w should pick up m bikes from the origin station c_i and drop off these bikes at the destination station c_j. Hence, to avoid illegal repositioning, at each time slot t, we need to get the valid candidate action space first for agent w based on the global-view environment state e^t. Specifically, we select the future jammed stations from the environment as the origin station set C_{ori}, which have sufficient bikes. The starved stations are selected as the destination station set C_{des}, which has enough empty docks. According to C_{ori} and C_{des}, we get the candidate action space $\mathcal{A}_w^t$ of agent w:

$$\begin{aligned}\mathcal{A}_w^t &= \left\{a_w^t\right\} \\ &= \left\{\left(c_i,\ c_j,\ m\right) \mid c_i \in C_{ori},\ c_j \in C_{des},\ m = \min\left(c_i^{b(t)},\ c_j^{d(t)}\right)\right\},\end{aligned} \tag{8}$$

where m is the number of moving bikes, constrained by the available bikes of start station $c_i^{b(t)}$ and the empty docks of end station $c_j^{d(t)}$. For an example, if there are x bikes at station c_i, and y empty docks at station c_j, $x > y$, so $m = \min(x,\ y)$, the agent should move y bikes from c_i to c_j.

- **Reward function $\mathcal{R}$:** In MDP, the reward function usually determines the optimization goal and reflects the immediate performance of the action taken under the current state. Formally, each agent w obtains the immediate rewards r_w^t after taking action a_w^t under s_w^t and transiting to the next state s_w^{t+1}. For simplification, in this work, we mainly focus on maximizing the satisfaction of the user's rent demands. we assume that a user who wants to return a bike but arrives at a jammed station without an empty dock will choose the nearest neighborhood station to return to. Hence, the reward is defined by the demand-supply gap g_w^t during $(t,\ t+1)$, which is calculated by the available bikes and net demands. In particular, as described in action space, $a_w^t = \left(c_i,\ c_j,\ m\right)$, an action could affect the bikes distribution of two stations, thus the g_w^t is defined as:

$$g_w^t = g_w^{start(t)} + g_w^{end(t)}. \tag{9}$$

For the start station, there are m bikes picked up, and the number of available bikes is updated by $c_i^{b(t)} = c_i^{b(t)} - m$. The real net demand of the station is $c_i^{n(t)}$. If $c_i^{n(t)} \le 0$ and $|c_i^{n(t)}| \ge c_i^{b(t)}$, there are more rent demands and existing available bikes cannot satisfy the users' rent demands, the gap $g_w^{start(t)}$ between available bikes $c_i^{b(t)}$ and the net demand $c_i^{n(t)}$ is calculated by $g_w^{start(t)} = |c_i^{n(t)}| - c_i^{b(t)}$. On the contrary, if $c_i^{n(t)} \le 0$ and $|c_i^{n(t)}| \le c_i^{b(t)}$, indicating there are enough bikes for users to rent, $g_w^{start(t)} = 0$. If $c_i^{n(t)} > 0$, there is no extra rent demand, the $g_w^{start(t)} = 0$. For the end station, the calculation of $g_w^{end(t)}$ is the same as the start station $g_w^{start(t)}$. Moreover, to avoid greedy repositioning and reduce the operation cost, we encourage the agents to choose the action with a short travel distance. Specifically, we use the total travel distance of a repositioning process as a reward constraint d_w^t. Precisely, d_w^t is the sum of the pick-up distance and the drop-off distance, defined as follows:

$$d_w^t = dis\left(loc_w,\ loc_{c_i}\right) + dis\left(loc_{c_i},\ loc_{c_j}\right), \tag{10}$$

Just Accepted

where the function *dis* calculates the real distance of these two locations. The first part indicates the pick-up distance, which is the distance of worker goes to the origin station to pick up bikes from their current location. The second part is the drop-off distance which is the distance between the origin and destination. Overall, the immediate reward r_w^t of action can be defined as follows:

$$r_w^t = -\left(g_w^t \ + \ \alpha d_w^t\right), \tag{11}$$

which are the negative sums of g_w^t and αd_w^t. Agent's goal is to maximize their reward function, which also means both the demand-supply gap and the worker travel distance are minimized. In particular, to scale distinct reward terms into the same range, we usually utilize the regularization ratio α. Each agent tries to maximize its long-term gains G_w^t:

$$G_w^t \ = \ \sum_{h=t}^{\infty} \gamma^{h-t} r_i^h. \tag{12}$$

- **Discount factor γ:** As shown in Eq. 12, γ is the discount factor. The value of γ effectively decides how much the agents care about long-term benefits relative to short-term gains. γ is commonly selected from $[0, \ 1]$. If $\gamma = 0$, the agent will be completely shortsighted and only learn about actions that can get high immediate rewards without considering any future long-term benefits. When $\gamma = 1$, the agent weighs both current and future rewards equally. A mathematical method to make an infinite sum finite is to limit γ less than 1 and larger than 0.
- **Probability function $\mathcal{P}$:** At time step t, each agent takes an action $a_w^t \in \mathcal{A}_w^t$ forming a set of joint action $\mathbf{a}^{\mathrm{t}} = \mathcal{A}_1^{\mathrm{t}} \times ... \times \mathcal{A}_{\mathrm{N}}^{\mathrm{t}}$, which induces a transition in the environment according to the state transition function $\mathcal{P}\left(s^{t+1}|s^t, \mathbf{a}^{\mathrm{t}}\right) : \mathcal{S} \times \mathcal{A}_1^t \times ... \times \mathcal{A}_N^t \rightarrow \mathcal{S}$.

4.2.2 Independent bike repositioning. Based on the MDP formulation of the bike repositioning problem, we first propose an independent bike repositioning model (ST-IBR), a straightforward MARL approach that directly applies the idea of independent Q-leaning to multi-agent settings. In our ST-IBR, each agent is fully independent and has its own learner. Each learner uses an improved Actor-Critic (AC) framework, Deep Deterministic Policy Gradient [22]. Typically, for each agent w, the Actor uses a policy network v_w with parameters δ_w to select action a_w^t based on current state s_w^t. After each action selection, the Critic evaluates whether the action made by the Actor is better or worse through a state-action function $q_w(s_w^t, \ a_w^t)$, and outputs the Q-value for the agent. However, just as introduced before in MDP, the candidate action space $\mathcal{A}_w^t$ is dynamic and changing over time, so ST-IBR makes some difference with the traditional AC setting.

To solve the problem of dynamic action sets, the Actor in ST-IBR uses an in-action approximation way to evaluate each state-action pair. As illustrated in Fig.5, for each candidate action e.g., $a_{w,1}^t$, ..., $a_{w,|\mathcal{A}_w^t|}^t$, it will input into the Actor with current state s_w^t together, and ST-IBR uses a deterministic policy $\hat{v}_w$: $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ to get the values of state-action $(s_w^t, a_{w,|\mathcal{A}_w^t|}^t)$ approximation $\hat{v}_w(s_w^t, a_{w,|\mathcal{A}_w^t|}^t)$. Moreover, to choose the best action a_w among candidates, ST-IBR uses a Boltzmann SoftMax selector to evaluate each pair and uses τ as the temperature parameter to control the selector exploration rate. :

$$\pi_w\left(a_{w,\ j}^t|s_w^t\right) = \frac{\exp\left(\tau v_w(s_w^t, \ a_{w,\ j}^t)\right)}{\sum_{j=1}^{|\mathcal{A}_w^t|} \exp\left(\tau v_w(s_w^t, a_{w,\ j}^t)\right)}. \tag{13}$$

Specifically, in ST-IBR, Actor works by directly adjusting the parameters δ_w of the policy v_w to maximize the objective $J\left(\delta_w\right)$:

$$J\left(\delta_w\right) = \mathbb{E}_{\mathrm{s,a}\sim\mathcal{D}}[G_w^t], \tag{14}$$

Just Accepted

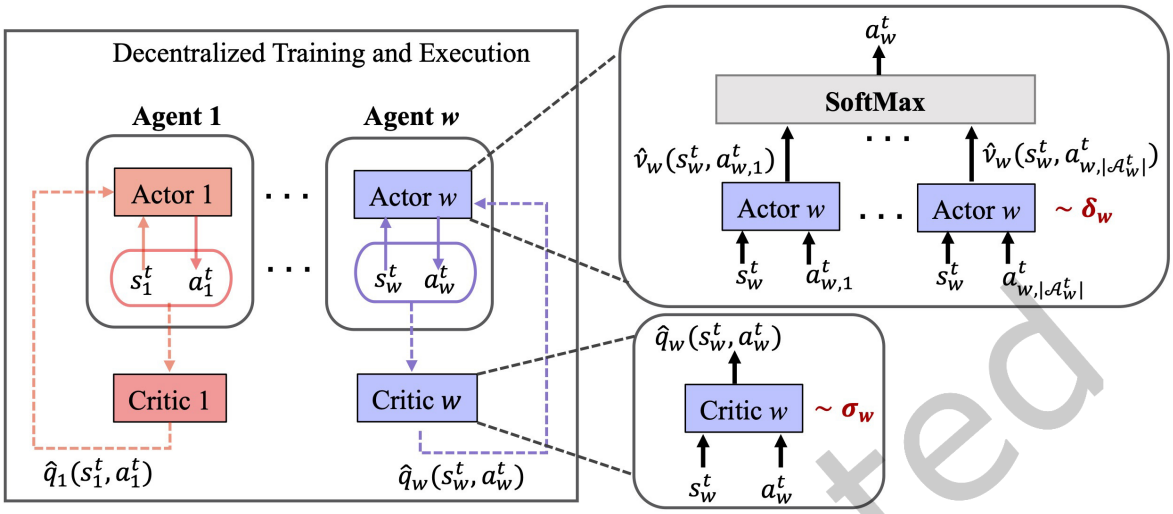


Fig. 5. Overview of ST-IBR.

by taking steps in the direction of $\nabla_{\delta_w} J(\delta_w)$:

$$\nabla_{\delta_w} J(\delta_w) = \mathbb{E}_{s,a\sim\mathcal{D}}[\nabla_{\delta_w} v_w(s_w^t, a_w^t) \nabla_{a_w^t} q_w(s_w^t, a_w^t)]. \tag{15}$$

The critic uses a DQN to learn the state-action function $\hat{q}_w(s_w^t, a_w^t)$, parameterized by σ_w by minimizing the loss:

$$\mathcal{L}(\sigma_w) = \mathbb{E}_{s,\,a,\,r,\,s'}\left[\left(y - \hat{q}_w(s_w^t, a_w^t,)\right)^2\right], \tag{16}$$

$$y = r_w^t + \gamma q_w^-\left(s_w^{t+1}, v_w^-(s_w^{t+1}, a_w^{t+1})\right), \tag{17}$$

where q_w^- is the target Q network, calculating the real state-action value. v_w^- is the target policy network. To improve training stability, these earlier parameters in the target network are periodically updated with the latest weights of evaluate network.

4.2.3 Cooperative bike repositioning. ST-IBR is a straightforward and feasible method to achieve independent bike repositioning for each agent, however, it ignores the impact of other agents' policies in the environment. We present a cooperative bike repositioning approach ST-CBR, which is based on ST-IBR and takes the policies of other agents into account to enhance agents' cooperation. The number of idle agents fluctuates with time, therefore, it is hard to achieve interaction among variable agents directly through simple information sharing. Mean Field Reinforcement Learning (MFRL)[47] is usually used to deal with the problem of variable agents and dimensional explosion when dynamic large-scale agents interact together. The main idea of MFRL is pairwise approximation. MFRL treats interactions among a large number of agents as interactions between one agent and the average effect from other agents, which shields the impact of the exact number of interacting pairs. Inspired by this idea, ST-CBR integrates our ST-IBR model with mean field approximations to address the interaction of variable agents. In the bike repositioning task, agents cooperate with each other to reduce the demand-supply gap and travel distance by selecting an action from a list of potential actions with a high reward. In ST-CBR, the average influence of other agents $\overline{a_w^t}$ is thus defined as the number of idle workers, divided by the number of

Just Accepted

candidate actions in the current time slot. For example, if there are three idle agents in the system, the size of the current candidate action space is 6, and the average influence is 3/6.

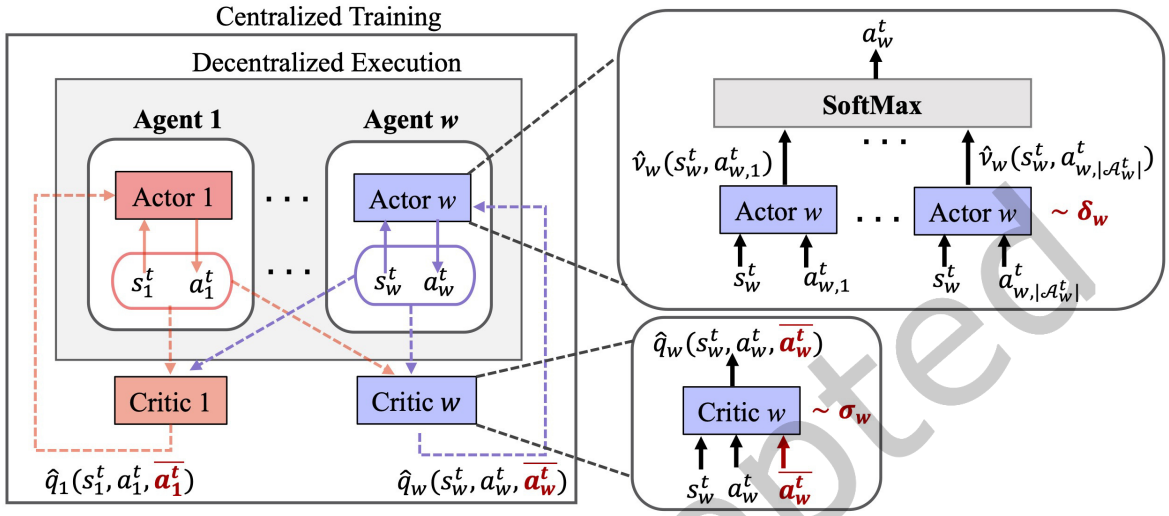


Fig. 6. Overview of ST-CBR.

Fig. 6 gives an overview of our proposed ST-CBR. ST-CBR follows a centralized training and decentralized execution way. During the training stage, ST-CBR enables agents to learn while interacting with others, i.e., in addition to the agent's action a_w^t, the average influence of other agents $\overline{a_w^t}$ is also used to learn the model, which can help to improve the agents' training stability and robustness. After the training stage, each agent can independently work in a decentralized fashion. As a result, it is resistant to a "single point of failure."

Specifically, $\overline{a_w^t}$ is the average influence of agents. The actor of ST-CBR learns the optimal policy by using the policy gradient:

$$\nabla_{\delta_w} J(\delta_w) = \mathbb{E}_{s,a\sim\mathcal{D}}[\nabla_{\delta_w} v_w(s_w^t, a_w^t) \nabla_{a_w^t} q_w\left(s_w^t, (a_w^t, \overline{a_w^t})\right)]. \tag{18}$$

The critic of ST-CBR is trained by minimizing the loss:

$$\mathcal{L}(\sigma_w) = \mathbb{E}_{s,\,a,\,r,\,s'}\left[\left(y - \hat{q}_w\left(s_w^t, a_w^t, \overline{a_w^t}\right)\right)^2\right], \tag{19}$$

$$y = r_w^t + \gamma q_w^-\left(s_w^{t+1}, v_w^-(s_w^{t+1}, a_w^{t+1})\right), \tag{20}$$

In the same time slot, active agents share the same candidate action space, which may cause the collision of action selection, i.e., various agents may choose the same action based on their own policy. Such conflict may make the bike repositioning invalid. To solve this, we make the agents select the action sequentially during the same time slot, once the action is selected by one agent, the current candidate sets will be updated for the other agents.

Just Accepted

Table 1. Citi Bike data in New York City.

Number of Stations	321
Time	1 year
Historical Records	14,191,731

5 EXPERIMENT

5.1 Experimental Setting

Data Description. We utilize 1-year bike usage records data from Citi Bike in NYC, which has been introduced in the previous data-driven investigation. In particular, our experiments focus on the 321 stations in the urban center area of NYC, as illustrated in 2, since the severely unbalanced situation always happens in these stations. In addition to the historical records data, we also collect the corresponding holiday data, meteorological data, and POI data for feature engineering to achieve final demand prediction. We use bike usage data from June to October for evaluation and other months for training. The information of the dataset is shown in Table 1.

Episode Setting. As we can see in Fig. 2, seriously unbalanced situations usually happen in the morning and evening rush hours, and the station can be self-balanced during the midnight hours. Thus, we mainly learn the repositioning strategy during the daytime, i.e., the episode in our works is set from 06:00 to 20:00, and the objective of our method is to maximize the rewards during an episode. We divide the episode into several time slots t_i. The setting of time slot t_i influences the performance of both prediction and repositioning. For prediction, a too short time interval can not get accurate results. For repositioning, a too-long interval will cause more waiting time for idle workers. To solve this, we set the time slot t_i as a 1-hour duration in ST-NetPre to get accurate net demand prediction of each time slot, and we split the time slot t_i into three small time interval t_i^j i.e., $t_i^j = 20$ minutes to conduct each of repositioning, as illustrated in Fig. 7. We use the average net demand of each small time interval as the predicted net demand. For example, if the predicted net demand in t_i is 9, the average net demand during t_i^j is $9/3 = 3$.

Initial Bikes Setting. To verify the effectiveness of our scheduling strategy, especially under conditions where the number of bikes in the system is significantly insufficient, it is indeed important to start with various random initial bike inventories to simulate different scenarios. By doing so, we can assess how well our scheduling strategy adapts to different situations and ensures that more users are satisfied despite the limited number of bicycles available. So, we conducted experiments under the different numbers of initial bikes. At the beginning of an episode, we set the initial bikes of each station as a random value that is between 0 and the product of a and $|c_i|$, $|c_i|$ is the capacity of a station c_i, and a reflects the percentage of available bikes at the station, $a \in (0, 1]$. For example, if $a = 0.8$, the capacity of a station is $c_i = 20$, and the product of a and $|c_i|$ is 16, so we set the initial bikes for this station to be 16. This percentage allows us to better understand the availability of bikes at each station and analyze the efficiency of our proposed optimization technique for different degrees of bicycle scarcity. During our experimentation with various random initial bike inventories at the stations, we indeed ensured that the total number of bikes present at all stations was equal to the total number of bikes available within the system. This was done to maintain consistency and uphold the integrity of our research.

Simulator Setting. Unlike standard supervised learning, where the labeled datasets are provided to evaluate the learning models by training and testing paradigm, DRL requires a dynamic simulation environment for training and evaluation due to the interactive characteristic. Hence, we design a BSS simulator based on real-world datasets that models the generation of bike usage demands, simulates the procedure of assigning bike repositioning tasks for agents, and tracks changes of status in BSS. The simulator is used to train and evaluate RL algorithms. In the BSS simulator, the simulation process in an episode is shown in Fig. 7.

Just Accepted

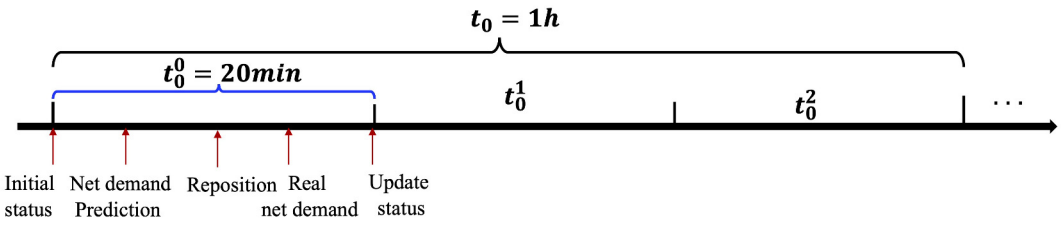


Fig. 7. Simulator Setting.

Specifically, the activities in an episode are conducted sequentially:

- *Get initial status.* At the beginning of an episode, we first get the initial station status, i.e., the distribution of bikes, and the status of agents. At the beginning of each episode, we set all the agents idle.
- *Predict net demand.* The features of stations in time slot t_i are extracted and inputted into our proposed ST-NetPre, and the net demand of each station is predicted. We then get the net demand of a small time interval t_i^j by dividing the net demand equally into three parts.
- *Execute repositioning task.* During each t_i^j, agents interact with the environment and get the action indication by ST-CBR. Agents reposition the bikes from a jammed station to a starved station in advance. The status of stations and agents is updated correspondingly. The distribution of bikes is updated after repositioning. The status of agents changes between busy and idle. If the agents are assigned the repositioning task, the status of agents is updated to busy. Otherwise, vice versa.
- *Get real net demand.* After repositioning, a sequence of rent and return events is generated by a real-world bike usage record dataset. If there are enough bikes, the users' rent demand can be satisfied. Otherwise, users fail to rent bikes and leave.
- *Update status.* The status of stations is updated based on real net demand. To avoid illegal status, the number of bikes in each station is clipped to the range of 0 and the maximum capacity. If the repositioning task has been finished during the current time interval t_i^j, the status of the agent is updated to idle for the next repositioning task, and the location of the agent is updated as the final stayed station.

5.2 Prediction Performance

As introduced earlier, we regard the historical real bike usage data as the ground truth. The prediction performance is measured by the Mean Absolute Error (MAE). MAE can better show the real situation of prediction error. It calculates the total absolute differences between real values and predicted values divided by the total number of predictions, $MAE(y, \hat{y}) = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$, where y_i is the real value of net demand, $\hat{y}_i$ is the predicted result by a model. We compare our ST-NetPre with other state-of-the-art prediction approaches, e.g., Linear Regression (LR), which is commonly used for regression problems; XGBoost [9], a state-of-the-art tree-based boosting method; Deep Neural Network (DNN) [24], which has experienced great success in many fields because its great power of high-order feature representation learning; Wide&Deep (W&D) model [10], a typical hybrid network structure that contains a linear "wide" component to learn the importance of low-order cross features and a "deep" component to model high-order feature interactions simultaneously.

Fig. 8 shows the MAE of different models. Thanks to our comprehensive feature engineering, the extracted features are substantially related to the consumers' bike usage behaviors, thus all of the prediction algorithms perform well. Compared with baselines, our proposed ST-NetPre has the best prediction performance, the MAE is 1.4, which indicates the net demand prediction of most bike stations is accurate. The XGBoost and W&D also

Just Accepted

achieve good performance. The XGBoost is the basic model of ST-NetPre. W&D can better learn the low- and high-order cross features to enhance the prediction result.

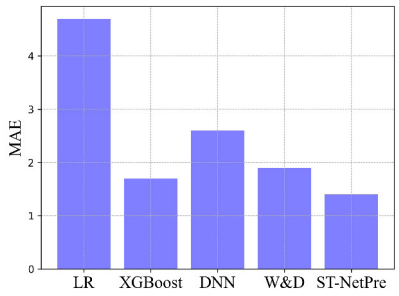


Fig. 8. MAE of Different Models.

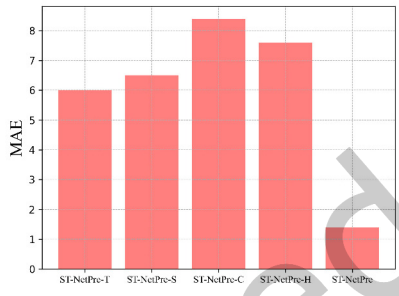


Fig. 9. Performance comparison *w.r.t.* the different feature classes in ST-NetPre.

We also conducted an evaluation of the performance of different feature classes. Specifically, Fig. 9 illustrates the results for various feature classes in ST-NetPre. ST-NetPre-T, ST-NetPre-S, ST-NetPre-C, and ST-NetPre-H have the same architecture as ST-NetPre but consider features differently. ST-NetPre-T assesses the effectiveness of temporal features in demand prediction, while ST-NetPre-S and ST-NetPre-C correspond to considering spatial features and contextual features, respectively. ST-NetPre-H focuses on historical features only. The results indicate that ST-NetPre achieves the best performance when incorporating all features, whereas using a single feature class generally results in poorer performance. This underscores the importance of considering multiple feature classes in the demand prediction model to achieve more accurate and reliable outcomes.

5.3 Repositioning Performance.

5.3.1 Metrics and Baselines. As introduced in the reward setting, the model measures whether the repositioning is worse or better by a reward function, that is related to both the demand-supply gap and operation cost (i.e., the travel distance of moving bikes) in an episode. In this work, the performance of the demand-supply gap is measured by GRR, which is the demand-supply gap reduction ratio compared with no repositioning. Moreover, we use ATD to evaluate the operation cost of different methods, which is the average travel distance of each repositioning over one episode.

To show the effectiveness of our proposed method, we compare it with several state-of-the-art repositioning approaches as follows:

- **No Repositioning (NO).** There is no bike repositioning strategy in the BSS simulator.
- **Random Repositioning (RAN).** No further strategy is taken into account by the random repositioning procedure. Agents always randomly choose an action from candidates at each time interval.
- **Greedy Demand-First Repositioning (DEM).** The greedy demand-first repositioning method prefers to achieve higher GRR by assigning agents to stations with a large demand-supply gap. During each time interval, all the actions in the candidate action space will be sorted by the number of moving bikes. In DEM, agents always select the action that the number of moving bikes is the largest which means it can meet more user demands by repositioning. Multiple actions with the same number of moving bikes will be further ordered by the repositioning distance, and the nearest one will be chosen.

Just Accepted

- **Greedy Distance-First Repositioning (DIS).** Greedy Distance-First Repositioning focuses on a lower reposition cost, i.e., ATD. The nearest repositioning task will be given priority. It calculates the repositioning distance of each candidate action and selects the nearest one to execute. Following the similar principle as mentioned in DEM, a repositioning task with a larger demand-supply gap will be executed first if multiple tasks have the same repositioning distance.
- **DQN [19].** The DQN is frequently used in solving MARL problems. The problem formulation is the same as our proposed method, while the decision model is only based on Deep Q-learning. As introduced in Section 4.2, since the action sets are variable, we use the in-action approximation methods to solve the problem. The Q network is built with both state and action as inputs. Specifically, the state is fed into an MLP with three hidden layers (128, 64, 32), and the action is fed into an MLP with one hidden layer (32). We use the ReLU [30] activation between hidden layers. Then we concat the two parts and transform the final linear output of the Q-network with ReLU.
- **ST-IBR.** The independent bike repositioning model ST-IBR as we introduced in Section 4.2.2. The network setting of state-action function approximation (i.e., the critic) is the same as that in DQN. The policy network (i.e., the actor) is parameterized by an MLP with four hidden layers (256, 128, 64, 32) for state input and one hidden layer (32) for action input. The ReLU activation is used between hidden layers and transforms the final linear output of the Q-network and policy network with ReLU and sigmoid function respectively.
- **ST-CBR.** As described in Section 4.2.3, ST-CBR is our proposed cooperative bike repositioning method with mean field approximation. Compared with ST-IBR, ST-CBR can take the other agents' policies into consideration to enhance the cooperation between agents. The network architecture is the same as ST-IBR, with the exception that a mean action of other agents is fed as an additional input to the critic network, as shown in Fig. 6. We add an MLP with one hidden layer (32) for the mean action input. Moreover, ST-CBR follows a centralized training and decentralized execution way.

In our model, both the target network and the evaluation network have the same architecture. We use the Adam Optimizer with a learning rate of 0.01 for the actor and 0.001 for the critic. The discounted factor γ is 0.90. To stabilize the training process, we use an experience replay buffer[29] for all methods and use an evaluate network to estimate the value, and a target network to calculate the real value, and the size of the replay buffer is 10,000. The batch size is 512. We update the network parameters after every 600 samples are added to the replay buffer. We use the Boltzmann SoftMax selector to select the action. We adopt the ϵ-greedy policy in all methods. Throughout the training phase, ϵ is annealed linearly from 0.9 to 0.1, while during the evaluation phase, it is fixed at $\epsilon = 0.1$.

5.3.2 Result Analysis.

Demand-supply gap reduction ratio (GRR). Table. 2 shows the performance comparison regarding the demand-supply gap reduction ratio. The percentage difference shown for all methods is with respect to No Repositioning (NO). We can clearly see that ST-CBR surpasses all the baselines under the different distribution of initial bikes. RAN suffers from the lowest GRR since it is a fully random way, without any policy or rule instruction. DIS is the second lowest method, that's because DIS is a Greedy Distance-First Repositioning method, it aims to select the action with the shortest travel distance instead of considering the demand-supply gap. Though DEM is a Greedy Demand-First Repositioning method, it is myopic and can not consider the long-term influence of repositioning. Hence, the reinforcement learning-based methods, e.g., DQN, ST-IBR, and ST-CBR perform better than DEM. Through the effective Actor-Critic framework and the idea of MFRL, ST-CBR takes other agents' policies into consideration, enabling cooperation among agents, thus the performance is better than the other independent learning methods, e.g., DQN and ST-IBR. Moreover, as the number of initial bikes decreases, the performance of methods is getting worse. This phenomenon is not difficult to understand. If there is not a sufficient number

Just Accepted

Table 2. Performance comparison regarding the demand-supply gap reduction ratio (GRR). The percentage difference shown for all methods is with respect to No Repositioning (NO).

Setting	0∼ 70% initial bikes	0∼ 50% initial bikes	0∼ 30% initial bikes	0∼ 10%initial bikes
Metrics	Normalized GRR	Normalized GRR	Normalized GRR	Normalized GRR
RAN	49.06% ± 0.25%	45.66% ± 0.20%	36.19% ±0.26%	30.66% ± 0.29%
DEM	65.21% ± 0.14%	57.56% ± 0.11%	50.22% ± 0.28%	35.70% ± 0.24%
DIS	52.26% ± 0.18%	47.69% ± 0.23%	40.34% ± 0.26%	32.92% ± 0.17%
DQN	67.88% ± 0.13%	63.82% ± 0.21%	54.10% ± 0.26%	36.71% ± 0.33%
ST-IBR	70.47% ± 0.20%	68.62% ± 0.19%	56.29% ± 0.16%	37.32% ± 0.30%
ST-CBR	**76.37% ± 0.15%**	**72.70% ± 0.32%**	**64.77% ± 0.14%**	**39.06% ± 0.10%**

Table 3. Performance Comparison regarding the Average Travel Distance (ATD).

Metrics	RAN	DEM	**DIS**
Normalized ATD (km)	4.219 ± 0.231	4.003 ± 0.145	**1.926 ± 0.133**
Metrics	DQN	ST-IBR	**ST-CBR**
Normalized ATD (km)	2.608 ± 0.148	2.311 ± 0.106	**2.098 ± 0.180**

of bikes in the system, the impacts of repositioning methods are limited. Nevertheless, ST-CBR is still the best repositioning method over all baselines.

Average Travel Distance (ATD). Performance Comparison regarding the Average Travel Distance (ATD) is shown in Table. 3. RAN has the largest ATD. DEM has the second largest ATD since it aims to adopt the action that has a large demand-supply gap, without considering the travel distance. As shown in Fig. 2, we can notice that the stations in the neighborhood usually have similar bike usage patterns. DEM may need to pick up bikes from a more distant jammed station to a starved station to meet more user demands. Both DIS and ST-CBR perform well in terms of ATD. DIS is a Greedy Distance-First Repositioning method, while it can not achieve a trade-off between GRR and ATD. DQN, ST-IBR, and ST-CBR achieve the trade-off between GRR and ATD through the reward function. Compared with DIS, ST-CBR is with a slight advantage in ATD. Meanwhile, ST-CBR has the best performance in GRR.

Effectiveness of Prediction. In particular, to evaluate the performance of prediction in repositioning, almost all the baselines, we also conduct a version without prediction. The only difference between them is the calculation of candidate actions. If the one is based on the prediction result, the candidate actions consider the net demand for the next time interval, so that the worker will move a certain number of bikes from a predicted jammed station to a predicted starved station. If the one isn't based on prediction, the calculation of candidate action space doesn't consider the predicted net demand, just based on the current station status. The variants of baselines without prediction are RAN-NP, DEM-NP, DQN-NP, ST-IBR-NP, and ST-CBR-NP. Table. 4 shows the performance of methods without prediction in terms of GRR. ST-CBR-NP has the best performance among these methods. ST-IBR-NP and DQN-NP follow behind. Compared to Table. 2, we can clearly see that the GRR of methods with prediction is better than that without prediction. The careful net demand prediction is critical for repositioning, as it can help the model find the stations that really need to be scheduled.

Effectiveness of Reward Setting. As introduced in Section 4.2.1, the reward function consists of the demand-supply gap and operation cost. To evaluate the effectiveness of the reward function, we remove the operation cost in the reward function of RL methods, i.e., DQN-ND, ST-IBR-ND, and ST-CBR-ND. Table. 5 shows the performance

Just Accepted

Table 4. Performance comparison regarding the demand-supply gap reduction ratio (GRR). The percentage differences shown for all methods **(without prediction)** are relative to No Repositioning (NO).

Setting	0∼ 70% initial bikes	0∼ 50% initial bikes	0∼ 30% initial bikes	0∼ 10%initial bikes
Metrics	Normalized GRR	Normalized GRR	Normalized GRR	Normalized GRR
RAN-NP	42.16% ± 0.13%	34.61 % ± 0.21%	27.18% ± 0.11%	20.13% ± 0.42%
DEM-NP	59.11% ± 0.16 %	51.91% ± 0.27%	43.13% ± 0.19%	33.61% ± 0.31%
DIS-NP	46.31% ± 0.12%	42.14% ± 0.10%	33.90% ± 0.24%	27.26% ± 0.26%
DQN-NP	61.43% ± 0.20%	56.93% ± 0.17%	50.62% ± 0.23%	30.28% ± 0.17%
ST-IBR-NP	63.81% ± 0.14%	60.01% ± 0.20 %	52.38% ± 0.15%	32.01% ± 0.14%
ST-CBR-NP	**71.88% ± 0.13%**	**67.08% ±0.12%**	**56.45% ± 0.22 %**	**34.32% ± 0.17%**

Table 5. Performance Comparison regarding the Average Travel Distance (ATD) **(without considering the operation cost)**.

Metrics	DQN-ND	ST-IBR-ND	ST-CBR-ND
Normalized ATD (km)	3.020 ± 0.130	2.981 ± 0.108	2.866 ± 0.124

of these methods regarding the ATD. All methods without considering the operation cost suffer from a worse performance in ATD. Compared to Table. 3, we can see that ATD is larger than the corresponding method which considers the operation cost in the reward function. The reward function we formulated can efficiently minimize the demand-supply gap, meanwhile, minimize the travel distance of agents.

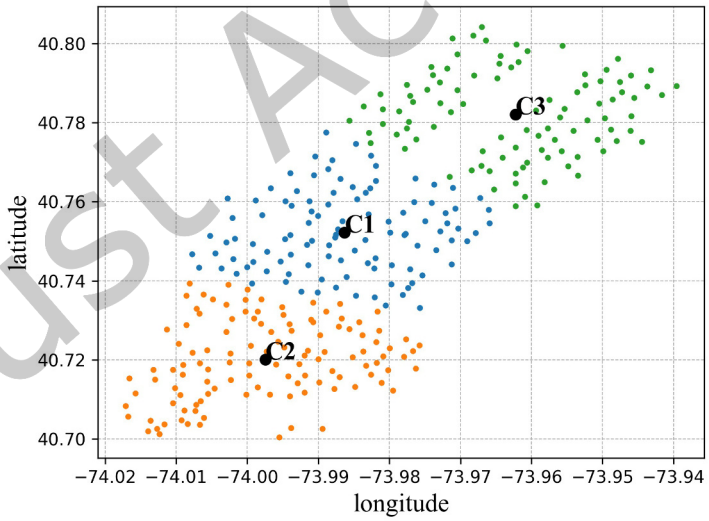


Fig. 10. Stations Clustering

Robustness of the Repositioning Method. Previous experiments have been conducted on the whole city. To evaluate the robustness of our proposed method, we also implement experiments in different small regions

Just Accepted

Table 6. Performance comparison of methods in terms of GRR and ATD over different clusters.

Clusters	Cluster C_1	
Metrics	Normalized GRR	Normalized ATD
RAN	57.98% ± 0.13%	2.211 ± 0.121
DEM	72.91% ± 0.17%	2.809 ± 0.111
DIS	60.66% ± 0.33%	1.896 ± 0.089
DQN	75.88% ± 0.23%	2.083 ± 0.058
ST-IBR	84.47% ± 0.20%	1.911 ± 0.075
ST-CBR	**88.37% ± 0.31%**	**1.759 ± 0.180**
Clusters	Cluster C_2	
Metrics	Normalized GRR	Normalized ATD
RAN	55.96% ± 0.20%	2.318 ± 0.198
DEM	66.91% ± 0.23%	2.403 ± 0.104
DIS	60.29% ± 0.20%	1.526 ± 0.067
DQN	76.26% ± 0.14%	1.986 ± 0.067
ST-IBR	82.26% ± 0.22%	1.722 ± 0.201
ST-CBR	**89.79% ± 0.26%**	**1.657 ± 0.096**
Clusters	Cluster C_3	
Metrics	Normalized GRR	Normalized ATD
RAN	54.76% ± 0.16%	2.811 ± 0.115
DEM	67.98% ± 0.15%	3.103 ± 0.128
DIS	60.45% ± 0.12%	2.099 ± 0.135
DQN	73.26% ± 0.34%	2.370 ± 0.167
ST-IBR	79.26% ± 0.36 %	2.260 ± 0.099
ST-CBR	**85.67% ± 0.19%**	**2.094 ± 0.185**

in the city respectively. We have noticed that from Fig. 2, the bike use pattern has strong regional unbalanced characteristics. Hence, we cluster the stations into different clusters based on location information by K-Means [21]. The clustering result is shown in Fig. 10, i.e., C_1, C_2, C_3. From Fig. 2 and Fig. 10, we can see that each cluster contains both jammed and starved stations, implying that each cluster can keep self-balanced through repositioning policy. Moreover, to ensure fairness, the number of agents in the whole city should be the sum of agents in each cluster. For example, there are twelve agents to learn the strategy and conduct repositioning tasks in the experiments of the whole city, there are four agents for each clustered region. In this experiment, we set the initial bikes as the random value between 0 and 70% of the station capacity. Table. 6 shows the performance comparison of methods in terms of GRR and ATD over different clusters. We can clearly see that, for different clusters, ST-CBR performs best on the measures of both GRR and ATD, it can help the BSS to reposition bikes to minimize the demand-supply gap at low cost. Compared with the results of the whole city, as shown in Table. 2, the metrics are better under the experiment of each cluster. The reason is that our repositioning is a station-level method, the candidate action space is large in the whole city. Repositioning in each cluster rather than the whole city can reduce the searching space of actions (i.e., the number of candidate stations is small), which can help improve the performance of the methods. Moreover, as the range of a cluster is much smaller than the whole city, the repositioning distance is reduced accordingly.

Just Accepted

6 DISCUSSION

- Profitability of operators and user experience are both critical for the development of the bike-sharing system. From our studies, we found our proposed BikeBrain can satisfy highly dynamic bike usage demand, thereby improving the user experience and increasing the profits of BSS. Although our experiments are based on Citi bike in NYC, they can be easily generalized to other cities or even other bike-sharing systems. Feature engineering needs to be reworked in new cities. Moreover, the fleet management problem in ride-sharing is also similar to bike repositioning. We believe that our method has the potential to be adopted to enhance the system management of other forms of sharing systems. Different from our worker-based repositioning method which needs to hire extra workers to perform the repositioning task, the drivers in the ride-sharing system will perform the repositioning task by themselves.
- Although our proposed method can efficiently alleviate the demand-supply unbalance in the bike-sharing system and have the ability to generalize to other cities and platforms, there are also some limitations to be aware of. Firstly, reinforcement learning is known to be unstable and hyperparameters affect the behavior of the learning system. We need to pay attention to improving the stability of reinforcement learning in future work. Secondly, though we have conducted extensive experiments through a simulator, that is designed with real-world historical data, to ensure the performance of each experiment, it is hard to guarantee the performance in a real-world situation. In future work, we are trying to collaborate with BSS operators and conduct experiments in real scenarios.

7 CONCLUSION

In this paper, we propose BikeBrain, a novel bike repositioning system with joint predicting and repositioning bikes in BSS to alleviate demand-supply unbalance at low operation cost. Our research holds significant social implications, including improved urban mobility, reduced traffic congestion, and promotion of sustainable and eco-friendly transportation. Specifically, we first propose a net demand prediction model ST-NetPre. To ensure the prediction performance, we conduct comprehensive data-driven investigations and extract a series of features that are highly related to users' bike usage patterns. Then we carefully formulate the bike repositioning as an MDP with multi-agents and propose ST-CBR, an efficient spatio-temporal cooperative multi-agent reinforcement learning algorithm. ST-CBR not only enables cooperation between a dynamic number of agents via mean-field reinforcement learning but also learns optimal repositioning strategies to bridge the gap between user demand and supply at low operation costs. We conduct extensive experiments based on a large-scale real-world bike usage record dataset from a typical bike-sharing system Citi Bike and multi-source urban data. Results have shown the effectiveness of our proposed method over several state-of-the-art baselines on the demand-supply gap and operation cost measures. Despite BikeBrain's success in addressing demand-supply imbalances and its potential for generalization to other cities, there are still some limitations to consider. Reinforcement learning can be unstable, and hyperparameters can affect the system's behavior, which should be addressed in future work. Additionally, while our experiments used a simulator designed with real-world data, the performance in real-world scenarios may vary. Collaborating with bike-sharing operators to conduct experiments in real scenarios could help validate and refine our method, further enhancing its applicability and effectiveness in real-world situations.

Just Accepted

ACKNOWLEDGMENTS

This work was partially supported by the National Science Fund for Distinguished Young Scholars (62025205), the National Natural Science Foundation of China (No. 62032020, 61960206008, 61725205, 62302017), the China Postdoctoral Science Foundation (No. 2023M730058), and the National Funded Postdoctoral Program of China (No. GZB20230017).

REFERENCES

[1] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. 2017. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641* (2017).

[2] Szilárd Aradi. 2020. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems* 23, 2 (2020), 740–759.

[3] Di Chai, Leye Wang, and Qiang Yang. 2018. Bike flow prediction with multi-graph convolutional networks. In *Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems*. 397–400.

[4] Jianguo Chen, Kenli Li, Keqin Li, Philip S Yu, and Zeng Zeng. 2021. Dynamic bicycle dispatching of dockless public bicycle-sharing systems using multi-objective reinforcement learning. *ACM Transactions on Cyber-Physical Systems (TCPS)* 5, 4 (2021), 1–24.

[5] Longbiao Chen, Zhihan Jiang, Jiangtao Wang, and Yasha Wang. 2019. Data-Driven Bike Sharing System Optimization: State of the Art and Future Opportunities.. In *EWSN*. 347–350.

[6] Longbiao Chen, Daqing Zhang, Gang Pan, Xiaojuan Ma, Dingqi Yang, Kostadin Kushlev, Wangsheng Zhang, and Shijian Li. 2015. Bike sharing station placement leveraging heterogeneous urban open data. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 571–575.

[7] Longbiao Chen, Daqing Zhang, Leye Wang, Dingqi Yang, Xiaojuan Ma, Shijian Li, Zhaohui Wu, Gang Pan, Thi-Mai-Trang Nguyen, and Jérémie Jakubowicz. 2016. Dynamic cluster-based over-demand prediction in bike sharing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 841–852.

[8] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[9] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[10] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[11] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

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

[13] Supriyo Ghosh, Michael Trick, and Pradeep Varakantham. 2016. Robust repositioning to counter unpredictable demand in bike sharing systems. (2016).

[14] Supriyo Ghosh, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. 2017. Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research* 58 (2017), 387–430.

[15] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International conference on autonomous agents and multiagent systems*. Springer, 66–83.

[16] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. 2018. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2034–2039.

[17] Guofu Li, Ning Cao, Pengjia Zhu, Yanwu Zhang, Yingying Zhang, Lei Li, Qingyuan Li, and Yu Zhang. 2021. Towards smart transportation system: A case study on the rebalancing problem of bike sharing system based on reinforcement learning. *Journal of Organizational and End User Computing (JOEUC)* 33, 3 (2021), 35–49.

[18] Xinghua Li, Xinyuan Zhang, Cheng Cheng, Wei Wang, and Chao Yang. 2022. Dynamic Repositioning in Dock-less Bike-sharing System: A Multi-agent Reinforcement Learning Approach. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 170–175.

[19] Yexin Li, Yu Zheng, and Qiang Yang. 2018. Dynamic bike reposition: A spatio-temporal reinforcement learning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1724–1733.

[20] Yexin Li, Yu Zheng, and Qiang Yang. 2019. Efficient and effective express via contextual cooperative reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 510–519.

[21] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. 2003. The global k-means clustering algorithm. *Pattern recognition* 36, 2 (2003), 451–461.

[22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[23] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. 2016. Rebalancing bike sharing systems: A multi-source data smart optimization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1005–1014.

Just Accepted

[24] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. 2017. A survey of deep neural network architectures and their applications. *Neurocomputing* 234 (2017), 11–26.
[25] Meghna Lowalekar, Pradeep Varakantham, Supriyo Ghosh, Sanjay Dominik Jena, and Patrick Jaillet. 2017. Online repositioning in bike sharing systems. In *Twenty-seventh international conference on automated planning and scheduling*.
[26] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).
[27] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.
[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
[30] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Icml*.
[31] Ling Pan, Qingpeng Cai, Zhixuan Fang, Pingzhong Tang, and Longbo Huang. 2019. A deep reinforcement learning framework for rebalancing dockless bike sharing systems. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 1393–1400.
[32] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069* (2017).
[33] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 4295–4304.
[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[35] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.
[36] Adish Singla, Marco Santoni, Gábor Bartók, Pratik Mukerji, Moritz Meenen, and Andreas Krause. 2015. Incentivizing users for balancing bike sharing systems. In *Twenty-Ninth AAAI conference on artificial intelligence*.
[37] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. 2020. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 914–921.
[38] Shangyu Sun, Huayi Wu, and Longgang Xiang. 2020. City-wide traffic flow forecasting using a deep convolutional neural network. *Sensors* 20, 2 (2020), 421.
[39] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).
[40] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.
[41] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.
[42] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.
[43] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.
[44] Guang Wang, Zhou Qin, Shuai Wang, Huijun Sun, Zheng Dong, and Desheng Zhang. 2021. Record: Joint Real-Time Repositioning and Charging for Electric Carsharing with Dynamic Deadlines. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3660–3669.
[45] Jiadai Wang, Lei Zhao, Jiajia Liu, and Nei Kato. 2019. Smart resource allocation for mobile edge computing: A deep reinforcement learning approach. *IEEE Transactions on emerging topics in computing* 9, 3 (2019), 1529–1541.
[46] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1995–2003.
[47] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5571–5580.

Just Accepted

[48] Ke Yu, Chao Dong, Liang Lin, and Chen Change Loy. 2018. Crafting a toolchain for image restoration by deep reinforcement learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2443–2452.
[49] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 167–176.
[50] Xiaokang Zhou, Wei Liang, I Kevin, Kai Wang, Hao Wang, Laurence T Yang, and Qun Jin. 2020. Deep-learning-enhanced human activity recognition for Internet of healthcare things. *IEEE Internet of Things Journal* 7, 7 (2020), 6429–6438.
[51] Xiaokang Zhou, Wei Liang, I Kevin, Kai Wang, and Laurence T Yang. 2020. Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations. *IEEE Transactions on Computational Social Systems* 8, 1 (2020), 171–178.
[52] Xiaokang Zhou, Wei Liang, Ke Yan, Weimin Li, I Kevin, Kai Wang, Jianhua Ma, and Qun Jin. 2022. Edge-enabled two-stage scheduling based on deep reinforcement learning for internet of everything. *IEEE Internet of Things Journal* 10, 4 (2022), 3295–3304.

Just Accepted