

# A Real-Time Bike Trip Planning Policy With Self-Organizing Bike Redistribution

Junheng Wang<sup>✉</sup>, Fan Li<sup>✉</sup>, *Member, IEEE*, Song Yang<sup>✉</sup>, *Member, IEEE*,  
Youqi Li<sup>✉</sup>, and Yu Wang<sup>✉</sup>, *Fellow, IEEE*

**Abstract**—Bike Sharing Systems (BSSs) have emerged as an economical and eco-friendly solution to alleviate the last-mile problem in intelligent transport systems. As an exclusive route selection problem in BSSs, Bike Trip Planning (BTP) has a clear goal: to select the available routes with minimum time cost for bike users, while satisfying their basic needs, e.g., the bounded longest walking distance. In this paper, we propose a real-time Lyapunov-based Bike Trip Planning (LBTP) policy that considers the users' waiting at stations, which has not been sufficiently studied in the literature. Both the total time cost and the service rate of all users are the core criteria of a BSS, so we make use of the Lyapunov optimization theory to make a tradeoff between them. Our policy can achieve self-organizing bike redistribution without extra redistribution budget required. The evaluation results show the superiority of our policy on the both system utility and user service rate compared with the existing BTP policies, and reveal the extra travel time fairness degree among different types of users under our policy.

**Index Terms**—Bike trip planning, bike sharing system, crowd-sourcing, queueing.

## I. INTRODUCTION

IN RECENT years, the concept of Bike Sharing System (BSS) [1] has gained increasing attention as one of the solutions to the traffic congestion and environmental problems. As a flexible, efficient, economical and eco-friendly way for urban commuting, a BSS can be part of an efficient multi-modal transport system and enhance the satisfaction of the daily commuters and travelers.

A BSS consists of a number of stationary bike stations with limited bikes and docks, through which users can finish their trips by renting and returning bikes. A key to the efficient operation of a BSS is offering users with bike and

dock resources to enable them to take their optimal routes. Unfortunately, in the practical deployment with mass user demands, the restricted resources of bikes and docks and the imbalance of the resources among stations can hinder the efficiency of a BSS. Therefore, there have been the extensive recent studies on the bike resource (or demand) prediction [2], [3], Bike Trip Planning (BTP) [4], [5] and resource redistribution [6]–[8].

As a way to help users enjoy the benefits of BSSs and improve their experience, BTP has become a focal research area in BSSs. BTP aims to help users find available routes with minimum time cost, while avoiding the unavailable stations and satisfying the users' requirements (e.g., free-ride time limits and user preferences). Different from traditional routing problems [9]–[11], a bike trip in a BSS composes of three moving segments, i.e., two walking parts and one bike part. Moreover, the BTP policies need to additionally consider the real-time station availability (i.e., the availability of resources at the moment when users reach the station), which brings more difficulties in policy design to have the foresight to avoid the bike station congestion. Furthermore, the conventional routing policy that simply greedily evades the areas of station congestion is not the best solution. This is because routing the users with contrary demands from the waiting users of one station (for example, the returning behavior users to a station with no bikes, or the renting behavior users to a station with no free docks, which called *opposite users* to the station) can mitigate the congestion of the station. A “smart” policy should adopt more proactive moves to allocate more opposite users to the congested stations.

The imbalance of bike resources at different stations makes a BSS unstable, and is a primary obstacle to the throughput capacity of the BSS. The efficiency of a BTP policy is strongly affected by the number of busy (or unavailable) stations, especially under a large-scale station congestion. Fig. 1 shows an example of the optimal trip routes given by a BTP policy under a small-scale station congestion and a large-scale station congestion. In Fig. 1(a) and 1(b), the ideal trip route is unattainable because it includes an unavailable station. Compared with Fig. 1(a), the detour in Fig. 1(b) is even longer, which is due to less available stations. The statistics in [12]–[14] reveal that the large-scale station congestion is common on account of the temporal factors (such as rush hours) and the diversity of functional areas (such as office blocks and residential zones).

Manuscript received 3 September 2020; revised 25 February 2021 and 19 May 2021; accepted 30 June 2021. Date of publication 4 August 2021; date of current version 9 August 2022. The work of Fan Li was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62072040 and Grant 61772077 and in part by the Beijing Natural Science Foundation under Grant 4192051. The work of Song Yang was supported by the NSFC under Grant 61802018. The Associate Editor for this article was H. B. Celikoglu. (*Corresponding author: Fan Li.*)

Junheng Wang, Fan Li, and Song Yang are with the School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: wangjunheng0309@163.com; fl@bit.edu.cn; s.yang@bit.edu.cn).

Youqi Li is with School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: 3120160457@bit.edu.cn).

Yu Wang is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: wangyu@temple.edu).

Digital Object Identifier 10.1109/TITS.2021.3095177

To solve the station congestion problem, some bike redistribution schemes are commonly used, which usually require extra budget and may not be suitable for the BTP policy. Therefore, we propose a novel BTP policy which achieves self-organizing redistribution without extra redistribution budget required. It makes every user detour a little to improve the system stability and the service rate of users (i.e., the rate of the users who successfully finish their trips), and maintains the extra travel time fairness among users at the same time.

In the field of BTP, the existing work [4], [13], [15], [16] in the literature is limited since they often ignore the existence of the queues in bike stations, which is far from the reality. The users' waiting at stations is rarely discussed, and the consideration of the users' waiting at stations brings a practical challenge to our study. The policies that simply search the shortest route which only involves the available stations may not be optimal, because they always make detours to avoid busy stations, even when users can get resources after waiting shortly at these stations. Also, considering the users' waiting is profitable for the fully loaded stations in reality, because queueing at the busiest stations will decrease the stations' idle time. It is because whenever a user arrives the station and brings the needed resource (the bike or free dock resource), the waiting users at the stations will immediately utilize the resource. This increases the busy stations' throughput, which is crucial to the system efficiency. Thus a BTP policy that only considers station's availability is neither realistic nor efficient.

With the users' waiting at stations considered, it is reasonable to involve the *longest-waiting-time guarantee* in the BTP policy design to improve user experience, because users cannot tolerate the long waiting at bike stations. Users' experience, to a great extent, determines whether or not they will choose the BSS again, and hence largely influences the economic and environmental benefits of the BSS. In addition, the *longest-walking-time guarantee* should also be considered, since users will give up using the BSS if the walking distance is too long.

We summarize our contributions as follows:

- We propose a real-time waiting-time-aware BTP policy referred to as Lyapunov-based Bike Trip Planning (LBTP) policy. The LBTP policy balances a perfect tradeoff between the total time cost and the service rate of all users, while satisfying the user experience requirements. It performs well especially under the serious station congestion situations.
- Compared to the previous work, our study captures the rarely considered waiting time feature. Such first characterization involves a novel system with original station queues modeling, which provides a new base for the waiting-time-aware BTP policy design. We also make more efforts to satisfy user experience requirements and formulate an innovative BTP policy design problem taking the longest-waiting-time and longest-walking-time guarantees into consideration. Furthermore, our BTP policy successfully maintains the extra travel time fairness among all users for the practical deployment requirements.

- We evaluate our BTP policy under different station distributions, traffic scales and station congestion situations. Three kinds of demand pattern of congestion areas are modeled with distinctive traffic pattern in the simulation, and the evaluation results of our policy confirm its superiority on both system utility and user service rate. Also, the evaluation results show our policy maintaining the basic fairness among all users.

The rest of the paper is organized as follows. In Section II, we briefly review the related work in BSS field. In Section III and Section IV, we present the system model and the queue model, respectively. We detail our BTP policy in Section V. Our policy is evaluated in Section VI. Finally, we conclude the paper in Section VII.

## II. RELATED WORK

In this section, we first briefly review the related work in BSS field, then introduce the background of Lyapunov theory and its applications.

### A. Bike Sharing System

Research in BSS field can be classified into usage prediction [2], [3], [5], [17]–[19], resource redistribution [6]–[8] and BTP [4], [5], [15].

1) *Prediction Methods*: The aim of the prediction methods in BSS field is to predict the bike mobility and the station availability, as a decision support tool for resource redistribution, BTP and the BSS operation.

A large number of prediction work using data mining and machine learning have emerged, including some state-of-the-art methods like station availability prediction based on adjacent stations [2], forecast in light of seasonal trend and unpredictable events [5] and a comprehensive prediction considering other public traffic condition information [17], etc. Additionally, some Internet-of-Thing (IoT) enabled systems are employed to ensure the robustness and decrease the prediction errors [20]. Also a series of study [18], [19] discussed the accuracy differences when choosing different machine learning algorithms.

However, prediction in BSS itself is a challenging problem, because the usage of the bikes is inconstant and further affected by many factors [3].

2) *Redistribution Schemes*: The redistribution schemes aim to find the best way to rebalance the resource among stations with less redistribution cost, in order to maintain a high level of user satisfaction [21].

Some work [6], [22], [23] used trucks to redistribute bikes. Legros [24] developed a decision-support tool to determine the number of bikes and the priorities for redistribution. Caggiani *et al.* [25] further proposed a dynamic redistribution methodology that focused on the free-fleeting bike-sharing system. However, since the quantity of trucks is limited, the redistribution cannot directly help users find available bikes and empty docks in time [4], which does not improve user experience that well. Also, a BSS with a fixed number of trucks will cause a serious waste in most time of a day when

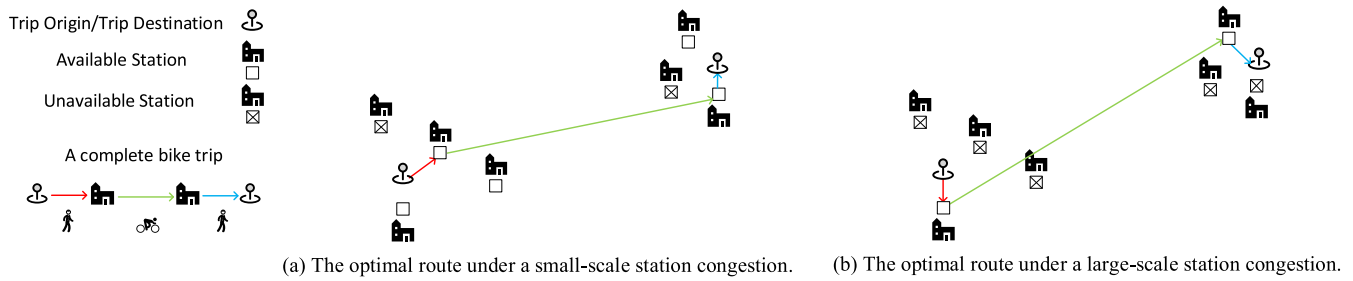


Fig. 1. BTP under different scales of station congestion.

satisfying the demands in rush hours, owing to the inflexibility of the truck-based redistribution systems.

Another state-of-the-art method [7], [26], [27] attempt to pay users to start trips or make detours to do the redistribution work. Luo *et al.* [28] divided users into passive scheduling users and active scheduling users, and used reward or punishment to incentive users. Gimon [29] assumed that users prefer the stations with more resources, and took the load of stations into account when deciding the incentive. Unfortunately, the users are unwilling to detour in rush hours, and the redistribution budget may be extremely high when facing a large-scale station congestion.

3) *BTP Policies:* BTP policy has a clear goal to find available routes with minimum time cost when considering the users' requirements.

Most of the BTP policies [5], [15] are based on the prediction methods. Yoon *et al.* [5] first proposed a comprehensive BTP policy considering the time cost and the probability of the availability of stations, and Xu *et al.* [15] further took the free-of-charge time period into account. Since they use the predicted probability of station availability as known information, the risk of binding all the planning efficiency to the prediction accuracy of the prediction method they chose is inevitable. As mentioned before, there is not a dominant prediction method that can get rid of all the effects of user randomness and unpredictable events, which results in the non-existence of a "perfect" prediction based BTP policy.

Considering such insufficiency, a small amount of the existing work focuses on the *real-time* trip planning with little prediction information (or history data). Li *et al.* [16] proposed a static trip planning solution, which fits well in the static large-scale problem but may suffer performance loss in real BTP scenes. This is because the user demands in real BTP scenes arrive dynamically and could never be known in advance. Zhang *et al.* [4] designed a game theory based solution that was inspired by the crowdsourcing. It was proved that the BTP problem is equivalent to a symmetric network congestion game within a few transformation steps, which means the participants can reach a Nash Equilibrium in finite turns, then all of them can find a nearly-best solution for bike trip selecting.

All these real-time policies enable all the participants to share their future usage information of the BSS to select bike trips, and the BTP operator can allocate the resources

before bikes and docks are actually occupied. The policies provide further foresight instead of simply using the real-time station availability information, while placing little reliance on prediction methods. Such real-time policies strike a balance between reliability and prescience.

Unfortunately, all above policies [4], [5], [15], [16] take no account of mitigating the station congestion. For this reason, to avoid efficiency loss caused by station congestion, additional redistribution schemes have to be applied, which usually require extra budget [6]–[8]. When facing a large-scale station congestion, as mentioned in Section I, they may still not perform well under such situation. Therefore, in this paper, we propose a real-time bike trip planning policy with self-organizing bike redistribution.

### B. Lyapunov Theory

The Lyapunov drift theory is a well-developed stability theory in the fields of Markov Chains and discrete stochastic processes [30]. It is further extended to the Lyapunov optimization theory, which is able to optimize the stability and performance at the same time. The Lyapunov drift theory is commonly used in queue system control, in order to maintain the system stability and achieve the throughput-optimal property. Leonardi *et al.* [31] proposed a flow-level dynamic routing policy based on Lyapunov drift theory, which can achieve a minimum total backlog among network. Neely [32] focused on the energy-efficient routing problem in the multi-hop wireless network. Lyapunov optimization theory is applied in [33] to achieve a maximum utility under the worst-case delay constraints. Some state-of-the-art works [34], [35] used the Lyapunov optimization theory in the mixed-cast traffic scene. Sinha *et al.* [34] first proposed a dynamic policy to solve the generalized network flow problem with point-to-point links. Zhang *et al.* [35] further studied the mixed-cast traffic flow problem under the distributed computing circumstances. Both the Lyapunov drift theory and the Lyapunov optimization theory are rarely used in BSS scene.

## III. SYSTEM MODEL

In this section, we introduce a comprehensive BSS system model, which can be divided into the bike trip model, the trip cost model, the user model and the station model. The definition of the longest-waiting-time and the longest-walking-time is also given.

### A. Bike Trip Model

A BSS includes a station set  $\mathbf{B} = \{b_1, \dots, b_i, \dots, b_M\}$ , a user set  $\mathbf{U} = \{u_1, \dots, u_j, \dots, u_N\}$ , a trip planning request set  $\mathbf{S} = \{s_1, \dots, s_k, \dots, s_P\}$ , and a trip set  $\mathbf{R} = \{r_1, \dots, r_k, \dots, r_P\}$ . Every user  $u_j \in \mathbf{U}$  holds a trip planning request set  $\mathbf{S}_j \subseteq \mathbf{S}$  and a bike trip set  $\mathbf{R}_j \subseteq \mathbf{R}$ . The whole BSS operates in discrete time with unit time slots  $t \in \{0, 1, \dots\}$ .

A complete bike trip  $r_k$  contains walking, biking and waiting parts with five segments: the first segment is from the user's source location (i.e., trip origin) to the starting station, the second segment is from the starting station to the target station, the third segment is from the target station to the terminal location (i.e., trip destination), and the last two segments correspond to the waiting in starting and target stations. We use  $r_k = \{e_1^k, e_2^k, e_3^k\}$  to represent the three moving segments of trip  $r_k$ , and let  $l_s(r_k)$  and  $l_t(r_k)$  denote the source and terminal locations of  $r_k$ . We further apply  $b_s(r_k) \in \mathbf{B}$  and  $b_t(r_k) \in \mathbf{B}$  to represent the starting and target stations that are chosen by the LBTP policy for trip  $r_k$ , respectively. Such three moving segments of one trip are divided on account of the differences in the time cost and physical consumption for traveling through a unit distance, which is a typical modeling approach that is realistic for practical use [4], [16].

The time cost and physical consumption are closely relevant to the distinctiveness of individual fitness condition, biking velocity, and the actual selected route, which are hard to qualify. For simplicity, we assume that there is *always only one shortest path* between any two locations and the differences among individuals are negligible. We therefore apply  $v_1$  and  $v_2$  to represent the travel velocity of the walking and biking parts, and use  $d(l_1, l_2)$  to represent the length of the shortest path between the two locations  $l_1$  and  $l_2$ . Such assumptions have been commonly made in BSS field [4], [16].

A trip planning request  $s_k$  contains the sources and terminal locations of the corresponding trip  $r_k$ , i.e.,  $s_k = \{l_s(r_k), l_t(r_k)\}$ .

### B. Longest-Walking-Time and Longest-Waiting-Time

As a part of user experience assurance, the longest-walking-time and longest-waiting-time guarantees are considered in this paper for practical deployment purposes.

Adhering to general intuition, the finite patience of users will finally lead to the upper limit of waiting time at stations. Unlike the existing work, stations with insufficient resource but can provide a longest-waiting-time guarantee will be considered as available stations in our system. This provides a better solution, since it provides more available stations for selection.

Considering the users may stand in line during their waiting, the First Input First Output (FIFO) service rule should be kept. Suppose if the longest-waiting-time settings of users depend on other attributes (e.g., the trip length or individual differences), it may happen that a user who is the last to arrive can first get the resource, due to his/her shorter longest-waiting-time limit setting. It is against the FIFO nature of queuing in reality. Therefore, we assume that the longest-waiting-time limits of different users are set to a same constant.

It is proved in [36] that the maximum tolerable walking distance settings are generally acceptable. It is also proved that the walking distance will not severely extend with the distance between the source and terminal locations of the trip, but depend on the distribution of stations. Hence, it is reasonable to set the longest-walking-time as a constant. Liu *et al.* in [37] detected that most of the bike-sharing stations are located near users' home/workplace or on a sidewalk near transit stations. Lu *et al.* in [36] went further in related fields and calculated an average walking distance in a BSS based on statistics, which provides us theoretic support for the longest-walking-time setting.

Without loss of generality, we assume that the longest-waiting-time limits in starting and target stations are separated and independent of each other. We consider similar assumptions for the longest-walking-time limits in the two walking parts (i.e., in  $e_1$  and  $e_3$ ).

Let  $L_{max}$  denote the longest-walking-time, and  $W_{max}$  denote the longest-waiting-time.

### C. Trip Cost Model and User Model

For simplicity, the total cost of one trip can be viewed as the total time cost consisting of the walking, biking, and waiting parts, the last of which is seldom discussed in the existing work. Taking the waiting part into consideration, the total time cost of trip  $r_k$  can be written as  $T_k = T_{0-1}^k + T_1^k + T_{1-2}^k + T_2^k + T_{2-3}^k$ , where  $T_{0-1}^k$ ,  $T_{1-2}^k$  and  $T_{2-3}^k$  represent the transfer time cost of  $e_1^k$ ,  $e_2^k$  and  $e_3^k$  (i.e.,  $T_{0-1}^k = d(l_s(r_k), b_s(r_k))/v_1$ ,  $T_{1-2}^k = d(b_s(r_k), b_t(r_k))/v_2$  and  $T_{2-3}^k = d(b_t(r_k), l_t(r_k))/v_1$ ), and  $T_1^k$  and  $T_2^k$  represent the real waiting time in starting and target station, respectively. Let  $T_0^k$  represent the basic walking time consumption of trip  $r_k$  without using the BSS as a baseline. We can calculate  $T_0^k$  as  $T_0^k = d(l_s(r_k), l_t(r_k))/v_1$ . We use  $\mathbf{T}_k = \{T_{0-1}^k, T_1^k, T_{1-2}^k, T_2^k, T_{2-3}^k\}$  to denote all the time cost composition for trip  $r_k$ .

In this paper, when dealing with the longest-waiting-time guarantee, we use the *real waiting time* that is different from the approximate waiting time considered in previous work [38]. The approximate waiting time is derived based on the queuing theory, and is a theoretical average value calculated by using the queue length and average departure rate. It is fairly obvious that when considering the longest-waiting-time guarantee, we should focus on the real waiting time.

The static BTP problem is then described as: given the station set  $\mathbf{B}$  and the trip planning request set  $\mathbf{S}$ , find the available routes (i.e., the available starting and target stations) for the requests so that the total system utility is maximized. We formulate the static BTP problem as follows:

$$\begin{aligned}
 \max \quad & \sum_{t=0}^{T_{\max}} \left[ \sum_{r_k \in \mathbf{R}(t)} g_k \left( \sum_{T \in \mathbf{T}_k} T \right) - \alpha \sum_{b_i \in \mathbf{B}} D_i(t) \right] \\
 \text{s.t.} \quad & T_{0-1}^k, T_{2-3}^k \leq L_{\max}, \quad \forall r_k \in \mathbf{R}(t), \\
 & T_1^k, T_2^k \leq W_{\max}, \quad \forall r_k \in \mathbf{R}(t), \\
 & T_k \leq T_0^k, \quad \forall r_k \in \mathbf{R}(t),
 \end{aligned} \tag{1}$$

where  $\mathbf{R}(t)$  is a set of the trips that are finished at slot  $t$ , and  $T_{\max}$  is the operation time of the BSS. Let  $g_k(\sum_{T \in T_k} T)$  denote the utility function of trip  $r_k$  based on the total travel time, which is a decreasing function. Let  $D_i(t)$  denote the drop rate (i.e., the number of the users who fail to rent or return bikes considering the longest-waiting-time constraint and exit the BSS per slot) of station  $b_i$  at slot  $t$ .  $\alpha$  is a weight parameter. As the users in the BSS arrive dynamically without prior knowledge, extra efforts should be made to deal with the BTP problem under dynamic settings.

In most of BTP papers [4], [5], [15], [16], the user model is seldom explicitly defined, and it can often be confused with the trip model. This is because in these papers, a user simply starts one trip. In this paper, a user  $u_j$  can submit multiply trip planning requests continuously.

#### D. Station Model

For any station  $b_i \in \mathbf{B}$ , its available bike and dock quantities  $n_i(t) \leq o_i^{\max}$  and  $o_i(t) \leq o_i^{\max}$  at slot  $t$  are updated in real time, where  $o_i^{\max}$  denotes the maximum quantity of the docks in station  $b_i$ . That means  $n_i(t) + o_i(t) = o_i^{\max}$  holds for all  $t \in \{0, 1, \dots\}$ . Each station's location  $l(b_i)$  and its initial quantity of bikes  $n_i(0)$  are recorded in the server beforehand as well. Also, all the planned trips and their selected station pairs will be recorded through the server, and the remaining arrival time to the next station and the waiting time of each user will be real-time calculated at the server. Such functions can be commonly realized by the IoT-based BSSs [39].

The station models are diverse among different types of stations. Considering the differences between the renting and returning bike behavior in the BSS, the asymmetries between the stations with renting user waiting queues (or called Real renting Queue Stations, RBQS in the following paper) and the stations with returning user waiting queues (Real Returning Queue Stations, RRQS) lead to the station modeling differences. Taking the longest-walking-time guarantee into account, the renting users seem to appear stochastically with less travel time (i.e., less than  $L_{\max}$ ) to the starting stations. However, the returning users can be all known in advance, but hard to arrive in time to offset the nearly overtime renting users for their longer travel time (i.e., the travel time  $T_{1-2}^k$ ) to the target stations. The diversity of the average travel time to the next stations between the renting and returning users as well as the longest-walking-time guarantee will not only lead to the difficulty disparity on trip planning, but also cause the station queue form differences that will be discussed in Section IV-A.

### IV. LYAPUNOV-BASED QUEUE MODEL

In this section, we introduce a Lyapunov-based queue model, and give a system objective function based on the queue model and the Lyapunov Optimization theory.

#### A. Virtual Station Queues

In this paper, the BSS server is required to record the planned arrival queues and the user waiting queues of stations. Consequently, a station  $b_i$  is designed to maintain one real queue  $\tilde{Q}_i(t)$ , and two virtual queues  $Q_i(t)$  and  $Q'_i(t)$ .

Let  $\tilde{Q}_i(t)$  represent the amount of waiting users queuing at station  $b_i$  for renting or returning bikes in the real world. The users in  $\tilde{Q}_i(t)$  are constrained by the longest-waiting-time guarantee, and can be dropped (or exit the BSS by themselves) whenever their waiting time is overtime, because of the insufficient resource.

Let  $Q_i(t)$  denote a virtual queue of station  $b_i$  that contains all the users whom are planned at slot  $t$  to pass station  $b_i$  in the future, but without arriving at the station at that time. To be more exact, a virtual queue  $Q_i(t)$  will only contain renting users and the returning users whose remaining arrival time is less than  $L_{\max} + W_{\max}$ . In this way we distinguish queue  $Q_i(t)$  from queue  $Q'_i(t)$  as follows.

Similarly,  $Q'_i(t)$  (which can be called pure virtual returning queue) is a virtual queue of station  $b_i$  consisting of the planned users who have not actually arrived and have remaining arrival time more than  $L_{\max} + W_{\max}$ . It is obvious that in queue  $Q'_i(t)$ , a user can only be a returning user who can never be matched with (or be offset by) any renting user.

As a result of the asymmetry between the RBQS and RRQS, the pure virtual renting queue is *non-existed*. Such properties provide necessity for the differentiation between  $Q_i(t)$  and  $Q'_i(t)$ .

As the difference between the remaining arrival time of two users in queue  $Q_i(t)$  can exceed  $W_{\max}$ , a virtual queue  $Q_i(t)$  may contain both renting users and returning users but none of them can be successfully offset according to the longest-waiting-time guarantee. Note that some examples include an RBQS with some renting users who have almost arrived and some returning users whose arrival time is far more than  $W_{\max}$ , or an RRQS with some returning users who have almost arrived and some renting users who are far away when  $L_{\max}$  is much larger than  $W_{\max}$ . In these examples, the virtual queues may contain both types of users but none of them can be offset. On account of such considerations, the virtual queue  $Q_i(t)$  needs to be divided into the virtual renting queue  $Q_{bi}(t)$  and the virtual returning queue  $Q_{ri}(t)$ . The update functions of  $Q_{bi}(t)$  and  $Q_{ri}(t)$  are listed as follows.

$$\begin{aligned} Q_{bi}(t+1) &= \max[Q_{bi}(t) - \mu_{bi}(t) - L_{bi}(t), 0] + A_{bi}(t), \\ Q_{ri}(t+1) &= \max[Q_{ri}(t) - \mu_{ri}(t) - L_{ri}(t), 0] \\ &\quad + A_{ri}(t) + C_i(t), \end{aligned}$$

where  $A_{bi}(t)$  and  $A_{ri}(t)$  are the arrival rates of queues  $Q_{bi}(t)$  and  $Q_{ri}(t)$  for station  $b_i$  at slot  $t$ , respectively.  $\mu_{bi}(t)$  and  $\mu_{ri}(t)$  are the service rates of queues  $Q_{bi}(t)$  and  $Q_{ri}(t)$ , which equal to the amounts of users who are offset by matching the different type of users at slot  $t$ .  $L_{bi}(t)$  and  $L_{ri}(t)$  are the passive departure rates of queues  $Q_{bi}(t)$  and  $Q_{ri}(t)$ , which equal to the amounts of users who arrive at station  $b_i$  without being immediately served and start to wait at the station. As the pure virtual queue  $Q'_i(t)$  only contains returning users, the passive arrival rate  $C_i(t)$  which equals to the amount of the returning users whose remaining arrival time reaches  $L_{\max} + W_{\max}$  at slot  $t$  will be only contained in  $Q_{ri}(t)$ .

All above variables can be affected by the LBTP routing decisions (i.e., the selected station pairs) for trips. It should be emphasized that the normal variables like the arrival rates

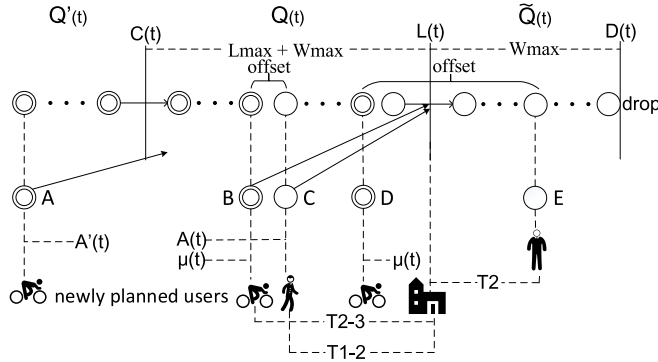


Fig. 2. Station queues model.

$A_{bi}(t)$  and  $A_{ri}(t)$  and the service rates  $\mu_{bi}(t)$  and  $\mu_{ri}(t)$  can be controlled in real time by scheming the newly planned users. However, the passive variables like  $L_{bi}(t)$  and  $L_{ri}(t)$  are only determined by the number of the users who depart to the connected queue at slot  $t$  without being offset, which can be hardly controlled flexibly in real time. These passive variables cannot be simply considered as normal controllable variables, which brings difficulties to the policy design.

The queue  $\tilde{Q}_i(t)$  is updated as follows:

$$\begin{aligned} \tilde{Q}_i(t+1) = & \max[\tilde{Q}_i(t) - \tilde{\mu}_i(t) - D_i(t), 0] \\ & + L_{bi}(t) + L_{ri}(t), \end{aligned} \quad (2)$$

where  $D_i(t)$  denotes the passive drop rate of the overtime waiting users, which equals to the amount of users whose waiting time reaches  $W_{\max}$  and exit the system.  $\tilde{\mu}_i(t)$  denotes the service rate for the actual queue, which equals the amount of users who successfully rent or return bikes at the station at slot  $t$ . The service rate  $\tilde{\mu}_i(t)$  can be controlled in real time, but the passive arrival rate  $L_{bi}(t) + L_{ri}(t)$  and the passive drop rate  $D_i(t)$  change passively.

The drop rate  $D_i(t)$  is a passive variable that cannot be proactively controlled because of the following reasons: on the one side, abandoning users proactively seems to make effort in mitigating the imbalance among stations, but neither makes contributions to the abandoned users' utility nor helps the long-term system utility, because it cannot avoid more users from being dropped. On the other side, proactively abandoning users during their traveling or waiting should never be accepted in a service provider position. Therefore we regard the variable  $D_i(t)$  can be only passively changed whenever a user's waiting time is overtime.

The pure virtual queue  $Q'_i(t)$  is updated as:

$$Q'_i(t+1) = \max[Q'_i(t) - C_i(t), 0] + A'_i(t),$$

where  $A'_i(t)$  denotes the arrival rate of the pure virtual queue for station  $b_i$  at slot  $t$ . Contrary to the real queue  $\tilde{Q}_i(t)$ , the arrival rate  $A'_i(t)$  here can be controlled in real time but the passive departure rate  $C_i(t)$  changes passively.

As will be readily seen, these three kinds of queues are end to end as shown in Fig. 2. Fig. 2 illustrates an example of the three queues of one station, where users A, B, C, D are the newly planned users at slot  $t$ , and user E is an example

of the waiting users. The arrivals of the newly planned users are directly converted into the change of the arrival rate or service rate according to the type of the users in the real queue of the station, but they can hardly influence  $C_i(t)$  and  $L_i(t)$  directly. Whenever the interval time between two opposite arrivals is less than  $W_{\max}$ , they are able to offset each other even if they are in different queues (e.g., user D and user E). The remaining users will consecutively move to the next queue until being served or reaching the longest-waiting-time threshold and being dropped.

Additionally, we need a virtual queue  $Z_i(t)$  to make sure the sum of the service rate and the drop rate of the real queue  $\tilde{Q}_i(t)$  is no far from the preset departure rate, which makes a foundation to meet the longest-waiting-time guarantee requirement. The virtual queue  $Z_i(t)$  is updated as follows:

$$\begin{aligned} Z_i(t+1) = & \max[Z_i(t) + (\hat{\mu}_i - \tilde{\mu}_i(t))\mathbf{1}\{\tilde{Q}_i(t) > 0\} \\ & - D_i(t) - \tilde{\mu}_i^{\max}\mathbf{1}\{\tilde{Q}_i(t) = 0\}, 0], \end{aligned} \quad (3)$$

where  $\mathbf{1}\{A\}$  represents an indicator function that takes value 1 if statement  $A$  is true, and parameter  $\hat{\mu}_i$  denotes the pre-specified departure rate of the real queue of station  $b_i$ . The service rate  $\tilde{\mu}_i(t)$  and the passive drop rate  $D_i(t)$  of overtime users compose a total departure rate of the real queue  $\tilde{Q}_i(t)$ , just the same as the total departure rate in equation (2). Let parameter  $\tilde{\mu}_i^{\max}$  denote the maximum service rate, which aims at changing the queue length of  $Z_i(t)$  to zero whenever  $\tilde{Q}_i(t) = 0$ .

### B. Lyapunov Optimization

Let  $\Theta_{(\tilde{Q}, Z, Q, Q')}(t) \triangleq [\tilde{Q}(t), Z(t), Q(t), Q'(t)]$  denote a collective vector of  $\tilde{Q}_i(t)$ ,  $Z_i(t)$ ,  $Q_i(t)$  and  $Q'_i(t)$  queues of all the stations in a BSS network at slot  $t$ . Then we can use a quadratic Lyapunov function  $L(\Theta)$  which is listed as below:

$$\begin{aligned} L(\Theta_{(\tilde{Q}, Z, Q, Q')}(t)) = & \frac{1}{2} \sum_{b_i \in \mathbf{B}} [a\tilde{Q}_i^2(t) + bZ_i^2(t) \\ & + cQ_i^2(t) + dQ_i'^2(t)], \end{aligned} \quad (4)$$

where  $a, b, c, d$  are the parameters which are used to weight different queues.

With the help of the Lyapunov function, we can further use one-step Lyapunov drift  $\Delta(\Theta)$  to represent the change of the Lyapunov function during one step (i.e., one slot), which reflects the stability of the BSS.

We define the Lyapunov drift  $\Delta(\Theta)$  as follows:

$$\begin{aligned} \Delta(\Theta_{(\tilde{Q}, Z, Q, Q')}(t)) = & L(\Theta_{(\tilde{Q}, Z, Q, Q')}(t+1)) \\ & - L(\Theta_{(\tilde{Q}, Z, Q, Q')}(t)). \end{aligned}$$

In most cases, the weights among different queues are typically set to 1 or  $\frac{1}{2}$  to stabilize all queues simultaneously. But in the context of our research, the priorities of stabilizing the queues  $Q'$ ,  $Q$  and  $\tilde{Q}$  at a low length have a non-negligible impact on system utility, so additional discussions on the weight setting are involved in the paper.

In a BSS, the total travel time and the service rate of users are always two core criteria of system, which we set as a utility function. To balance the total travel time, service rate

and system stability, we list the system objective function at slot  $t$  as follows:

$$\begin{aligned} \min \quad & \Delta(\Theta(\tilde{Q}, Z, Q, Q')(t)) - V \left[ \sum_{r_k \in \mathbf{R}(t)} g_k \left( \sum_{T \in \mathbf{T}_k} T \right) \right. \\ & \left. - \sum_{b_i \in \mathbf{B}} (\alpha D_i(t) - \beta E_i(\tau_i(t))) \right] \\ \text{s.t.} \quad & T_{0-1}^k, T_{2-3}^k \leq L_{\max}, \quad \forall r_k \in \mathbf{R}(t), \\ & T_1^k, T_2^k \leq W_{\max}, \quad \forall r_k \in \mathbf{R}(t), \\ & T_k \leq T_0^k, \quad \forall r_k \in \mathbf{R}(t), \end{aligned} \quad (5)$$

where  $V$ ,  $\alpha$ ,  $\beta$  are the weight parameters, and  $\tau_i(t)$  denotes a set of relevant positive variables. The definition of  $\tau_i(t)$  and  $E_i(\tau_i(t))$  will be introduced in Section V-C.

We turn the system utility function  $\sum_{r_k \in \mathbf{R}(t)} g_k(\sum_{T \in \mathbf{T}_k} T) - \sum_{b_i \in \mathbf{B}} \alpha D_i(t)$  into a component of the “penalty” part besides the Lyapunov drift, and use a parameter  $V$  to achieve a tradeoff between the system stability and the final performance.

The term  $\beta E_i(\tau_i(t))$  is additionally added into the object function to represent the urgency degree for the users in queue  $Q_i(t)$  or  $\tilde{Q}_i(t)$  to be served. Note that the urgency degree here corresponds to the remaining arriving time or the remaining waiting time of the head-of-line user in the queue. Take queue  $\tilde{Q}_i(t)$  as an example. The intuition is that the more a user’s current waiting time is close to the longest-waiting-time guarantee, or the larger the reduction on the waiting time of the head-of-line user in station  $b_i$  caused by a user’s departure is, the more motivation a system will have to serve this user. Neely in [33] did a good job of transferring the actual waiting time of the head-of-line packet into the length of a virtual queue, but it does not perform well under the scene of our work. This is because under the scene of our work, the relationship between the urgency precedence and the remaining waiting time is nonlinear, which we will discuss in Section V-C.

From the one-step update function of queues  $Q_{b_i}(t)$ ,  $Q_{r_i}(t)$ ,  $\tilde{Q}_i(t)$ ,  $Q'_i(t)$ ,  $Z_i(t)$ , we can get that the Lyapunov drift plus penalty can be upper-bounded by inequality (6), shown at the bottom of the page.

In inequality (6), the constant  $B$  is defined as  $B \triangleq \frac{1}{2}[3(A^{\max})^2 + 12(\mu^{\max})^2]$ , where  $A^{\max}$  and  $\mu^{\max}$  are the maximum values of  $A_i^{\max}$  and  $\mu_i^{\max}$  among all stations, respectively. The proof is achieved by bounding the maximum length of all queues by  $\mu_i^{\max}$ . In this sense, the service rate, passive

departure rate and drop rate can all be bounded by  $\mu_i^{\max}$ , for they can never be larger than the related queues’ current lengths. For the same reason, the arrival rate of all queues can be bounded by  $A_i^{\max}$ , equaling to the maximum acceptance rate of station  $b_i$  considering the maximum queue length constraint.

## V. LYAPUNOV-BASED BIKE TRIP PLANNING

In this section, we introduce the LBTP algorithm for both the user-side and the server-side, and then propose an improved LBTP policy to meet the longest-waiting time guarantee and the fairness guarantee. We finally discuss the simplicity of model and the rules of value settings.

### A. LBTP Algorithm

The LBTP algorithm is divided into two parts: the user-side algorithm to choose the optimal station, and the server-side algorithm for the server to exchange the latest BSS station information with the users. The two parts of LBTP algorithm are represented in Algorithm 1 and Algorithm 2, respectively.

We plan to reduce the the right-hand of inequality (6), which is the upper bound of the objective function (5). We first exploit the separable nature in inequality (6). In (6), all the variables can be divided into three classes:  $\tilde{\mu}_i(t)$ ,  $A'_i(t)$ ,  $A_{b_i}(t)$ ,  $\mu_{b_i}(t)$ ,  $A_{r_i}(t)$  and  $\mu_{r_i}(t)$  are normal decision variables;  $L_{b_i}(t)$ ,  $L_{r_i}(t)$ ,  $C_i(t)$  and  $D_i(t)$  are also decision variables but cannot be directly controlled; and  $\tilde{\mu}_i^{\max}$  and  $\hat{\mu}_i$  are pre-specified and uncontrollable.

---

#### Algorithm 1 User-Side LBTP Algorithm for User $u_j \in \mathbf{U}$

---

**Input:** The source location  $l_s(r_k)$  and the terminal location  $l_t(r_k)$  of the current trip  $r_k$  of user  $u_j$ .

**Output:** The selected station  $b_i$ .

- 1: Send a demand with the source location  $l_s(r_k)$  or the terminal location  $l_t(r_k)$  to the server, according to the trip state;
  - 2: Receive a set of optional stations from the server;
  - 3: Check all the optional stations to find the optimal station that can minimize the right hand of inequality (6);
  - 4: Send the selected station  $b_i$  to the server, and return  $b_i$ .
- 

Making the best use of the separable nature, the minimization problem can be solved by deriving a distributed user routing decision policy. When a user arrives at a BSS, after

$$\begin{aligned} & \Delta(\Theta(\tilde{Q}, Z, Q, Q')(t)) - V \left[ \sum_{r_k \in \mathbf{R}(t)} g_k \left( \sum_{T \in \mathbf{T}_k} T \right) - \sum_{b_i \in \mathbf{B}} (\alpha D_i(t) - \beta E_i(\tau_i(t))) \right] \\ & \leq \sum_{b_i \in \mathbf{B}} [B - \tilde{\mu}_i(t)(a\tilde{Q}_i(t) + \mathbf{1}\{\tilde{Q}_i(t) > 0\}bZ_i(t)) + A'_i(t)dQ'_i(t) - D_i(t)(a\tilde{Q}_i(t) + bZ_i(t) - V\alpha) \\ & \quad + A_{b_i}(t)cQ_{b_i}(t) - \mu_{b_i}(t)cQ_{b_i}(t) + A_{r_i}(t)cQ_{r_i}(t) - \mu_{r_i}(t)cQ_{r_i}(t) \\ & \quad - \tilde{\mu}_i^{\max}\mathbf{1}\{\tilde{Q}_i(t) = 0\}bZ_i(t) + \hat{\mu}_i\mathbf{1}\{\tilde{Q}_i(t) > 0\}bZ_i(t) + L_{b_i}(t)(a\tilde{Q}_i(t) - cQ_{b_i}(t)) + L_{r_i}(t)(a\tilde{Q}_i(t) - cQ_{r_i}(t)) \\ & \quad + C_i(t)(cQ_{b_i}(t) - dQ'_i(t)) - V\beta E_i(\tau_i(t))] - V \left( \sum_{r_k \in \mathbf{R}(t)} g_k \left( \sum_{T \in \mathbf{T}_k} T \right) \right). \end{aligned} \quad (6)$$

sending the demand (line 1), s/he will receive a set of optional stations with their storage queues' information from server (line 2), as described in Algorithm 1. The user-side algorithm then is able to check all the available stations to find an optimal station to minimize the right hand of (6), when keeping in view the impact on station queues. In the next step, the user-side algorithm can make its own choice to choose the only optimal route that minimize the objective function (5) (line 3). The selected station is finally sent to the server to update the station queues (line 4). With the selected station, the user-side algorithm is able to do a real-time navigation directing to its destination, by means of the IoT-based location tracking technology [40], [41] and traditional navigation methods (e.g., GPS).

Algorithm 2 is designed to update the trip information (line 1) and exchange the station information with users. The exchanges include the optional station sets for users to make their decisions (line 3) and the stations they finally choose (line 5).

The distributed solution of (5) does not cause a serious loss of system utility due to following analysis. It is not hard to see that the selection order among users only makes a difference when their prior stations are the same in one decision time slot. Such situation is under a little order of magnitude that has little impact, when there are a large quantity of stations and a sufficiently slight time slot. Also many classic algorithms [42], [43] can be easily involved to reduce the utility loss incurred by the decision conflict.

The distributed solution and the limited number of optional stations enable the LBTP algorithm to run in a computationally efficient manner.

### B. An Improved LBTP Policy With Guarantees

1) *Longest-Waiting-Time Guarantee*: To meet the longest-waiting-time guarantee requirement, we bound the lengths of  $\tilde{Q}_i(t)$  and  $Z_i(t)$  in order to bound the waiting time, as shown in the next lemma, which is a little like the theorem in [44].

#### Algorithm 2 Server-Side LBTP Algorithm

- 1: Update the remaining time of the trips in BSS;
- 2: **if** Receive a demand from a user **then**
- 3:   Provide the user with the information of the stations that within a  $v_1 L_{\max}$  radius from the location in the demand, including the location  $l(b_i)$  of each optional station  $b_i$  and its queues' information.
- 4: **end if**
- 5: **if** Receive a selected station from a user **then**
- 6:   Add the trip  $r_k$  of the user to the corresponding queue of the selected station.
- 7: **end if**

*Theorem 1: When the maximum queue length  $\tilde{Q}_i^{\max}$  holds for each queue  $\tilde{Q}_i(t)$  and the maximum queue length  $Z_i^{\max}$  holds for each queue  $Z_i(t)$  in the BSS, the longest-waiting-time  $W_i^{\max}$  for all the non-drop users in station  $b_i$  can be restrained by:*

$$W_i^{\max} \triangleq \lceil (\tilde{Q}_i^{\max} + Z_i^{\max}) / \hat{\mu}_i \rceil + 1, \quad (7)$$

where  $\lceil x \rceil$  represents the smallest integer that no less than  $x$ .

*Proof:* We define that for any user  $u_j$  who arrives at station  $b_i$  at slot  $t$ , if his or her waiting time of current trip  $r_k$  is  $W_k$ , then s/he will be sure to depart (i.e., be served or dropped) at  $t + W_k + 1$ . Because of the FIFO service rule that is true in the real scene for fairness, the trip  $r_k$  of the tail-of-line user  $u_j$  in  $\tilde{Q}_i(t)$  will be last served, and the newly arrival users after slot  $t$  will never influence the trip  $r_k$  or user  $u_j$ . So we assume that the arrival rate of the real queue of station  $b_i$  after slot  $t$  is all zero, i.e.,  $L_{b_i}(\tau) + L_{r_i}(\tau) = 0$  for all  $\tau > t$  without loss of generality.

Then with equation (2), we can get:

$$\tilde{Q}_i(t+1) - \tilde{Q}_i(t+W_k+1) = \sum_{\tau=t+1}^{t+W_k} [\tilde{\mu}_i(\tau) + D_i(\tau)].$$

It's obvious that the departure behavior of user  $u_j$  happens at slot  $t + W_k$ , which implies that the departure rate  $\tilde{\mu}_i(\tau) + D_i(\tau)$  at slot  $t + W_k$  will no less than 1. It is sure that the length of queue  $\tilde{Q}_i(t+W_k+1)$  will be zero after the last user (i.e., user  $u_j$ ) departs, so we get the following inequality:

$$\tilde{Q}_i(t+1) > \sum_{\tau=t+1}^{t+W_k-1} [\tilde{\mu}_i(\tau) + D_i(\tau)].$$

Then we have:

$$\tilde{Q}_i^{\max} > \sum_{\tau=t+1}^{t+W_k-1} [\tilde{\mu}_i(\tau) + D_i(\tau)], \quad (8)$$

for  $\tilde{Q}_i^{\max} \geq \tilde{Q}_i(t)$  holds for all  $b_i \in \mathbf{B}, t \in \{0, 1, \dots\}$ .

For every time slot, because of the maximum function in equation (3), the change of the virtual queue  $Z_i(\tau)$  is bounded as follows:

$$Z_i(\tau+1) - Z_i(\tau) \geq \hat{\mu}_i - [\tilde{\mu}_i(\tau) + D_i(\tau)].$$

Then we can sum over the range of  $\tau \in [t+1, t+W_k]$  and yields:

$$Z_i(t+W_k) - Z_i(t+1) \geq (W_k - 1)\hat{\mu}_i - \sum_{\tau=t+1}^{t+W_k-1} [\tilde{\mu}_i(\tau) + D_i(\tau)].$$

Recall that  $Z_i^{\max} \geq Z_i(t) \geq 0$  holds for all  $b_i \in \mathbf{B}, t \in \{0, 1, \dots\}$ , then we have:

$$Z_i^{\max} \geq (W_k - 1)\hat{\mu}_i - \sum_{\tau=t+1}^{t+W_k-1} [\tilde{\mu}_i(\tau) + D_i(\tau)]. \quad (9)$$

With inequality (8) and (9), finally we have for any waiting time  $W_k$  in station  $b_i$ :

$$\tilde{Q}_i^{\max} + Z_i^{\max} \geq (W_k - 1)\hat{\mu}_i,$$

which means

$$\tilde{Q}_i^{\max} + Z_i^{\max} \geq (W_i^{\max} - 1)\hat{\mu}_i,$$

which is equivalent to the definition given in equation (7).  $\square$

As deduced in Section IV-A, proactively dropping users to maintain the maximum queue length  $\tilde{Q}_i^{\max}$  or  $Z_i^{\max}$  is futile, since it provides no help to the system utility. Under such considerations, the maximum queue length constraints of  $\tilde{Q}_i(t)$  and  $Z_i(t)$  should hold in a “soft” way. This is achieved by controlling the weights in definition (4) with further study efforts, as shown in the next theorem.

**Theorem 2:** *Considering the longest-walking-time guarantee, the opposite users will always be routed to detour to pass the station whose queue  $\tilde{Q}_i(t)$  or  $Z_i(t)$  is at the maximum length, when the parameters in (4) satisfy  $a > \frac{v_1}{v_1+v_2} L_{\max}/\tilde{Q}_i^{\max}$  and  $b > \frac{v_1}{v_1+v_2} L_{\max}/Z_i^{\max}$ .*

**Proof:** Taking the longest-walking-time guarantee into account, the detour of one user is at most  $\frac{v_1+v_2}{v_2} L_{\max}$ , which can be used to delineate the able opposite users. And when  $a\tilde{Q}_i^{\max} > \frac{v_1+v_2}{v_2} L_{\max}$  and  $bZ_i^{\max} > \frac{v_1+v_2}{v_2} L_{\max}$ , due to the objective function (5), the able opposite users will be sure to routed by the user-side algorithm to detour to pass the busy station, whenever the station’s queue length reaches maximum. Such routing will disregard the users’ own time costs.  $\square$

In the practical application, the values of  $a$  and  $b$  in (4) also rely on other variables like the lengths of other queues. So the actual values of  $a$  and  $b$  in the real deployment will be much higher than the theoretical minimum values that calculated in Theorem 2.

2) *Fairness Guarantee Among Users:* It is accord with the thought of “crowdsourcing” [45]–[47] that everyone detours a little to facilitate others to save more time that improves the whole system utility, which is an intuitive way to alleviate the imbalance among stations. Although it provides an encouraging vision, it is unfortunately unpractical in reality because nobody wants to make the first move to impair his personal profits for higher system utility. One will sacrifice his current profits only if he is long-sighted and is assured to receive more benefits from an unobstructed traffic system than his sacrifice. These lead a further requirement of the extra travel time fairness promise among all users.

In line with such demands, all historical trip information of one user should be recorded, which achieves a long-sighted user model that theoretically satisfies the foundational requirements of the fairness request. Let  $\hat{\rho}_j$  denote the expected optimization rate of user  $u_j$  (i.e., the expected ratio of the travel time making use of BTP to the basic walking time cost of any trip of user  $u_j$ ), which maintains the fairness among users. We further apply  $\tilde{\rho}_j^k \triangleq T_k/T_0^k$  to denote the actual optimization rate of trip  $r_k$  of user  $u_j$ . In this paper, only a constant expected optimization rate guarantee constraint does not work. This is because even if all the users are under the same expected optimization rate guarantee constraint, the actual optimization rates of users are different, which directly influences the fairness among users. Therefore, we use both  $\hat{\rho}_j$  and  $\tilde{\rho}_j^k$  to measure how the users benefit from the BTP policy and to maintain the fairness among users.

Consequently, we harness the Lyapunov optimization theory to catch a dynamic balance between the expected optimization rate  $\hat{\rho}_j$  and the actual optimization rate  $\tilde{\rho}_j^k$ , and use a virtual queue  $U_j(t)$  to represent all the former benefits gotten from

the BTP policy, given by:

$$U_j(t+1) = \max[U_j(t) - \hat{\rho}_j, 0] + \tilde{\rho}_j^k(t).$$

The value of  $U_j(t)$  updates whenever a trip  $r_k$  of user  $u_j$  is finished, and the  $\tilde{\rho}_j^k(t)$  can be calculated based on the real situation, using the values of  $T_k$  and  $T_0^k$  of the newly finished trip  $r_k$ . The newly updated queue length  $U_j(t)$  will influence the route decision of the next trip of user  $u_j$  to maintain the fairness.

To control the actual optimization rate  $\tilde{\rho}_j^k$  and the queue length  $U_j(t)$  directly during one trip, a little trick is used as follows. We calculate the estimated whole travel time for the current trip when choosing the *target station*, for the distance of  $e_2^k$  and  $e_3^k$  are fixed for every alternative target station, and the waiting time at the target station can be predicted by the queue lengths. Then we can calculate the estimated optimization rate  $\tilde{\rho}_j^k$  using the estimated whole travel time. The estimated optimization rate  $\tilde{\rho}_j^k$  can be used to assist the decision making to dynamically control the queue length  $U_j(t)$ .

With the queue  $U_j(t)$  added in, the collective vector of queues is revised by  $\Theta_{(\tilde{Q}, Z, Q, Q', U)}(t) \triangleq [\tilde{Q}(t), Z(t), Q(t), Q'(t), U(t)]$ , and the quadratic Lyapunov function  $L(\Theta)$  and the Lyapunov drift  $\Delta(\Theta)$  are changed as:

$$L(\Theta_{(\tilde{Q}, Z, Q, Q', U)}(t)) = \frac{1}{2} \sum_{b_i \in \mathbf{B}} [a\tilde{Q}_i^2(t) + bZ_i^2(t) + cQ_i^2(t) + dQ_i'^2(t)] + \frac{1}{2} \sum_{u_j \in \mathbf{U}} eU_j^2(t), \quad (10)$$

and

$$\Delta(\Theta_{(\tilde{Q}, Z, Q, Q', U)}(t)) = L(\Theta_{(\tilde{Q}, Z, Q, Q', U)}(t+1)) - L(\Theta_{(\tilde{Q}, Z, Q, Q', U)}(t)),$$

where  $e$  is the parameter that is used to weight queue  $U_j(t)$ .

Finally the system objective function at slot  $t$  with queue  $U_j(t)$  involved updates as follows:

$$\begin{aligned} \min \quad & \Delta(\Theta_{(\tilde{Q}, Z, Q, Q', U)}(t)) - V \left[ \sum_{r_k \in \mathbf{R}(t)} g_k \left( \sum_{T \in \mathbf{T}_k} T \right) \right. \\ & \left. - \sum_{b_i \in \mathbf{B}} (\alpha D_i(t) - \beta E_i(\tau_i(t))) \right] \\ \text{s.t.} \quad & T_{0-1}^k, T_{2-3}^k \leq L_{\max}, \quad \forall r_k \in \mathbf{R}(t), \\ & T_1^k, T_2^k \leq W_{\max}, \quad \forall r_k \in \mathbf{R}(t), \\ & T_k \leq T_0^k, \quad \forall r_k \in \mathbf{R}(t). \end{aligned} \quad (11)$$

With the help of the inequality (12), shown at the bottom of the next page, when  $U_j(t)$  is added in, the user-side algorithm in Section V-A can still measure its choice’s influence on system and make its own decision.

In inequality (12), the constant  $B_2$  is defined by  $B_2 \triangleq 1$ , for  $B_2 \geq \frac{1}{2}\hat{\rho}_j^2 + \frac{1}{2}\tilde{\rho}_j^k(t)^2$  and  $\hat{\rho}_j \leq 1$ ,  $\tilde{\rho}_j^k(t) \leq 1$  hold for all  $u_j \in \mathbf{U}$ ,  $r_k \in \mathbf{R}(t)$ ,  $t \in \{1, 2, \dots\}$ .

In order to bound the maximum difference between the actual optimization rate and the expected optimization rate,

the maximum queue length constraint  $U_j^{\max}$  of each queue  $U_j(t)$  is involved, and is maintained by presetting a reasonable value  $e$  in (10). Such bounded maximum differences make the worst optimization rate guarantee among all users. We have the following theorem.

**Theorem 3:** *The user  $u_j$  whose queue  $U_j(t)$  is at the maximum length will always be allocated the shortest feasible route, when the parameter in (10) satisfies  $e > (a \max_{b_i \in \mathbf{B}} \tilde{Q}_i^{\max} + b \max_{b_i \in \mathbf{B}} Z_i^{\max}) / U_j^{\max}$ .*

*Proof:* The proof here is similar to the proof of Theorem 2. We can get the deviation caused by redistribution duties is at most  $a \max_{b_i \in \mathbf{B}} \tilde{Q}_i^{\max} + b \max_{b_i \in \mathbf{B}} Z_i^{\max}$ . And when  $e U_j^{\max} > (a \max_{b_i \in \mathbf{B}} \tilde{Q}_i^{\max} + b \max_{b_i \in \mathbf{B}} Z_i^{\max})$ , due to the objective function (11), the user  $u_j$  will always be allocated the shortest feasible route whenever the queue  $U_j(t)$  is at the maximum length, and ignore their redistribution duties.  $\square$

Thus far, we achieve a rough fairness by maintaining a maximum length  $U_j^{\max}$  for the queue  $U_j(t)$  for every user  $u_j$ .

### C. Further Analysis

Although the value settings of  $a$ ,  $b$  and  $e$  have been exhaustively discussed, the value settings of  $V$ ,  $c$ ,  $d$ ,  $\beta$  and the function expression of  $E(\tau_i(t))$  still require additionally studies.

For simplicity of exposition, we first simplify our model.

1) *For Simplicity of Model:* To simplify the model and reveal our policy performance under a general setting, we assume that all the stations and users in the BSS share the same set of parameters and ignore the disparity, i.e.,  $\tilde{Q}_i^{\max} = \tilde{Q}_{\max}$ ,  $Z_i^{\max} = Z_{\max}$ ,  $W_i^{\max} = W_{\max}$ ,  $U_j^{\max} = U_{\max}$ ,  $E_i(\tau_i(t)) = E(\tau_i(t))$ ,  $g_k(T) = g(T)$  for all  $t \in \{0, 1, \dots\}$ ,  $b_i \in \mathbf{B}$ ,  $u_j \in \mathbf{U}$ ,  $r_k \in \mathbf{R}$ .

We remind readers that all the parameters and the function definitions can be designed diversely based on the station situation, user group diversity, and time period differences, in order to achieve a better performance.

2) *The Value Settings of  $V$ ,  $c$  and  $d$ :* We first regard the utility function  $g(T)$  of time cost as a core criterion and set  $V = 1$ , then the parameter  $c$  indexes the virtual queue length  $Q_i(t)$  to the time cost  $T$ . Consequently,  $cQ_i(t) + g(T_i)$  can be regarded as the utility of choosing station  $b_i$  according to the waiting time cost and the single segment travel time

cost, where  $T_i$  denotes the user travel time to station  $b_i$ . Such settings give the choice practical significance. Unfortunately, the average actual waiting time of one virtual queue is strongly relevant to the macroscopic asymmetric demand level of the BSS and the macroscopic real-time congestion situation of its station (i.e., the lengths of  $Q_i(t)$  and  $\tilde{Q}_i(t)$ ). So under the same traffic scale and asymmetric demand level, we use a constant value  $c$ .

Parameter  $d$  for the pure virtual queue  $Q'_i(t)$  is like  $c$  for  $Q_i(t)$ . As stabilizing the pure virtual queue  $Q'_i(t)$  has a lower priority than stabilizing queue  $Q_i(t)$ , we apply  $d = \frac{1}{2}c$ .

3) *The Value Settings of  $\beta$  and the Function Expression of  $E(\tau_i(t))$ :* As for  $\beta$ , we apply  $\beta = \beta_1$  and  $\beta = \beta_2$  to denote the urgency degree of the trip in  $Q_i(t)$  and  $\tilde{Q}_i(t)$  to be served respectively. As each trip can only offset one trip in one queue, the value of  $\beta$  can be either  $\beta_1$  or  $\beta_2$ , and no conflict exists here on  $\beta$  value.

Although queue  $Z_i(t)$  ensures the average service time of a nonempty real queue  $Q_i(t)$ , it still cannot handle the irregular and non-uniform distribution of the arrival time of the users in  $\tilde{Q}_i(t)$ . We define  $\tau_i(t) = \{T_i(t), T_{wi}(t), \Delta T_i(t)\}$ , where  $T_i(t)$  denotes the remaining arriving or waiting time of the head-of-line trip of station  $b_i$  at slot  $t$ ,  $T_{wi}(t)$  denotes the average service time of the corresponding queue that reaches the maximum length in station  $b_i$ , and  $\Delta T_i(t)$  denotes the increase of the remaining arriving or waiting time of the head-of-line trip of the corresponding queue in station  $b_i$  caused by the newly planned trip.

$\beta_2 E(\tau_i(t))$  aims to prevent the users in the real queue from being dropped, and we assume that whenever  $\Delta T_i(t) \geq T_{wi}(t)$ , an additional user can barely be served instead of being dropped, which avoids a utility loss of the drop penalty  $\alpha$ . Therefore  $\beta_2 E(\tau_i(t)) \leq \alpha$  should hold, where  $E(T_i(t), T_{wi}(t), \Delta T_i(t))$  is defined as:

$$E(T_i(t), T_{wi}(t), \Delta T_i(t)) = \min\left\{\frac{T_{wi}(t)}{T_i(t)} \frac{\sum_1^{T_i(t)+\Delta T_i(t)} \tau^2 - \sum_1^{T_i(t)} \tau^2}{\sum_1^{T_{wi}(t)} \tau^2}, 1\right\}. \quad (13)$$

Both the degree of urgency and the contribution of avoiding the overtime users are taken into account with the help of  $E(\tau_i(t))$ . The calculation of  $T_{wi}(t)$  will be under detailed discussion at the end of this subsection, according to different queues and situations.

$$\begin{aligned} & \Delta(\Theta(\tilde{Q}, Z, Q, Q', U)(t)) - V[\sum_{r_k \in \mathbf{R}(t)} g_k(\sum_{T \in \mathbf{T}_k} T) - \sum_{b_i \in \mathbf{B}} (aD_i(t) - \beta E_i(\tau_i(t)))] \\ & \leq \sum_{b_i \in \mathbf{B}} [B - \tilde{\mu}_i(t)(a\tilde{Q}_i(t) + \mathbf{1}\{\tilde{Q}_i(t) > 0\}bZ_i(t)) + A'_i(t)dQ'_i(t) - D_i(t)(a\tilde{Q}_i(t) + bZ_i(t) - V\alpha) \\ & \quad + A_{bi}(t)cQ_{bi}(t) - \mu_{bi}(t)cQ_{bi}(t) + A_{ri}(t)cQ_{ri}(t) - \mu_{ri}(t)cQ_{ri}(t) \\ & \quad - \tilde{\mu}_i^{\max} \mathbf{1}\{\tilde{Q}_i(t) = 0\}bZ_i(t) + \hat{\mu}_i \mathbf{1}\{\tilde{Q}_i(t) > 0\}bZ_i(t) \\ & \quad + L_{bi}(t)(a\tilde{Q}_i(t) - cQ_{bi}(t)) + L_{ri}(t)(a\tilde{Q}_i(t) - cQ_{ri}(t)) + C_i(t)(cQ_{bi}(t) - dQ'_i(t)) + V\beta E_i(\tau_i(t))] \\ & \quad - \sum_{u_j \in \mathbf{U}} [B_2 + \tilde{\rho}_j^k(t)eU_j(t) - \hat{\rho}_j eU_j(t)] + V(\sum_{r_k \in \mathbf{R}(t)} g_k(\sum_{T \in \mathbf{T}_k} T)). \end{aligned} \quad (12)$$

For the same type of logic,  $\beta_1 E(\tau_i(t))$  aims to prevent the users in virtual queue  $Q_i(t)$  from being added into the real queue  $\tilde{Q}_i(t)$ , from which it can be inferred that  $\beta_1 E(\tau_i(t)) + c \leq a$ . As  $E(\tau_i(t)) \leq 1$  always hold in (13),  $\beta_1 = a - c$  and  $\beta_2 = a$  can be finally decided.

The average service time of  $\tilde{Q}_i(t)$  and  $Q_i(t)$  in RBQS and RRQS is different, according to the asymmetry between the real and virtual queues and the asymmetry between RBQS and RRQS. When defining the width and length of the distribution area of the BSS are  $x_{\max}$  and  $y_{\max}$ , and defining the average generation rate of trip demands per square meter at slot  $t$  is  $d_n(t)$ ,  $T_{w_i}(t)$  under different types of queues and stations can be calculated by:

$$\begin{aligned} \frac{\pi}{4} \sum_1^{T_{wrr}} p v_1^2 \tau^2 &= 1, \\ \frac{\pi^2}{4} \frac{L_{\max} v_1^2}{x_{\max} y_{\max}} \sum_1^{T_{wrb}} p v_2^2 \tau^2 &= 1, \\ \frac{\pi}{4} \sum_1^{T_{wrr}} p v_1^2 \tau^2 &= q, \\ \frac{\pi^2}{4} \frac{L_{\max} v_1^2}{x_{\max} y_{\max}} \sum_1^{T_{wrb}} p v_2^2 \tau^2 &= q, \end{aligned}$$

where  $T_{wrb}$ ,  $T_{wvb}$  denote the average service time of the real and virtual queues of RBQS, and  $T_{wrr}$ ,  $T_{wvr}$  denote the average service time of the real and virtual queues of RRQS, respectively. Let  $p$  denote the mean generation rate of trip demands, i.e.,  $\mathbb{E} d_n(\tau) = p$ . Usually we apply  $q = 1$  when the real queue is empty, and apply  $q > 1$  to neutralize the influence of subordinate priority of the virtual queues.

## VI. EVALUATION RESULTS

### A. System Setup

According to [2], the users whom are not served would never be recorded in the real dataset, not to mention the potential users that are deterred by the serious congestion situation among bike stations. This makes the real dataset cannot objectively reflect the actual amount of demands and the true station congestion scale in the real world. In order to figure out that how the efficiency of our policy and other BTP policies is influenced by different scales of station congestion which is hardly to find out by means of the real dataset, we evaluate our policy in a simulated way.

Our simulation mechanisms and settings are based on [48], which provided an exhaustive mechanism to authentically simulate a specific existing BSS includes station, user demand and redistribution vehicle simulated generation modules.

In this paper, a rectangular area with width  $x_{\max} = 7\text{km}$  and length  $y_{\max} = 7\text{km}$  is used to represent a city centre, which is usually the main distribution region of one BSS. We apply the station simulated generation module settings in [48] with some amount of revisions, where the lateral and longitudinal average distances between stations are set to  $r_1 = r_2 = 0.5\text{km}$  with a random deviation  $r_3 = 0.1\text{km}$  in the first step. The second step in [48] is replaced by two rules which are commonly applied

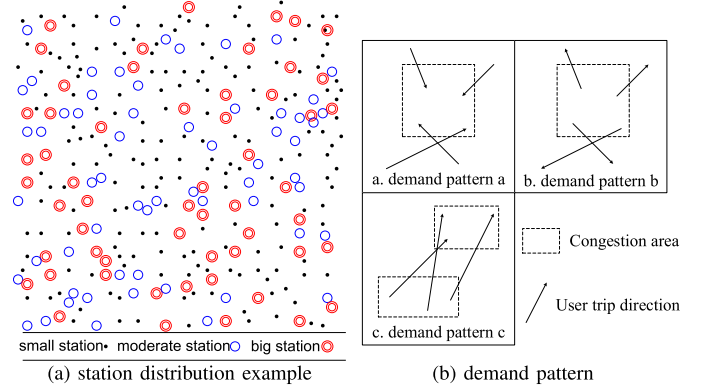


Fig. 3. Example of station distribution and demand pattern.

TABLE I  
TRAFFIC DEMAND SETTINGS

Time	6:30-9:00	9:00-11:30	11:30-13:30	13:30-16:30	16:30-19:30	19:30-21:30	21:30-6:30
Deamnd	peak	offpeak	medpeak	offpeak	peak	offpeak	special
Scale	3	1	2	1	3	1	0.1-1

TABLE II  
THE DISTRIBUTION OF TRIP LENGTHS

Trip Length (Km)	[0.5, 2.0]	[2.0, 3.0]	[3.0, 4.0]	[4.0, 5.0]
Proportion (percent)	33.8	33.5	27.8	4.9

in the BSS design. *Rule 1* defines that everywhere in the map should have an adjacent station within  $0.5\text{km}$  to avoid blank areas, and *Rule 2* defines no two stations could be adjacent to each other less than  $0.2\text{km}$  to avoid waste of resources. After the set up of the first step, we continually add stations into the map until the number of stations reaches  $M = 300$  without violating Rule 1 and Rule 2.

All stations are divided into three categories (i.e., big, moderate, and small stations) with 60, 40, 20 docks and 60% initial bikes respectively, and the ratio of big, moderate and small stations is 2 : 2 : 6. Fig. 3(a) represents an example of the station distribution under above simulation settings.

A daily day is divided into 1000 time slots in this paper. We generate  $P_n = 20000$  normal demands per day, which are allocated to  $N = 10000$  users evenly as the normal daily traffic flow without simulating the asymmetric demands. Following the common views of the latest work, working hours of one station are divided into 7 periods with different traffic scales, as shown in Table I.

Under our settings, the user demands appear stochastically and dynamically without any priori knowledge, which is closer to the reality. The trips of demands randomly select a source location and a terminal location in the rectangular area, unless the length of the trip (i.e.,  $d(l_s(r_k), l_t(r_k))$ ) is less than  $0.5\text{km}$  or more than  $5\text{km}$  due to the real scenario consideration. Table II shows an example of the distribution of the trip lengths for our setting. The walking velocity and biking velocity of users are set to  $v_1 = 0.1\text{km}$  per slot and  $v_2 = 0.3\text{km}$  per slot.

The longest-waiting-time and the longest-walking-time values are defined as  $W_{\max} = L_{\max} = 10$  slots, according to which  $\hat{\mu}$  takes  $\hat{\mu} = 0.2 \sim 0.3$  and  $\tilde{Q}_{\max}$  and  $Z_{\max}$  take  $\tilde{Q}_{\max} = 1 \sim 1.5$  and  $Z_{\max} = 0.8 \sim 1.2$  in term of different traffic scales, respectively. The weight parameters in (10) are finally set to  $a = \frac{30}{\tilde{Q}_{\max}}$ ,  $b = \frac{30}{Z_{\max}}$ ,  $c = 2 \sim 6$  and  $d = 1 \sim 3$  in accordance with different traffic scales and asymmetric demand levels. The penalty parameter  $\alpha$  for the users without being serviced is set to  $\alpha = T_{\max} = 30$ , where  $T_{\max}$  denotes the general longest travel time under above settings.

### B. Demand Patterns of Congestion Areas

In this paper, we propose a group of demand patterns of congestion areas (i.e., the areas full of congestion stations) with three forms, i.e., demand patterns  $a$ ,  $b$  and  $c$  in Fig. 3(b). These three demand patterns are classified by the inflow and outflow differences, where demand pattern  $a$  leads to a congestion area caused by an influx of users flow into an certain area, demand pattern  $b$  leads to the congestion area caused by an influx of users flow out from an certain area, and demand pattern  $c$  leads to the congestion areas caused by an influx of users flow from one certain area to another.

Every congestion area  $c_m$  is determined by four variables: the central position  $l_c(c_m)$ , the scope  $r_c(c_m)$ , the duration  $\mathbf{T}_{c_m}$ , and the additional trip demands generation rate  $d_c(t, c_m)$ . With the help of such settings, different station congestion scales can be simulated agilely. Let  $\mathbf{C}$  be the set of all the congestion areas.

In this paper, the central positions of congestion areas are randomly selected without any priori knowledge to represent the random events. The scope  $r_c(c_m)$  and the duration  $\mathbf{T}_{c_m}$  are uniformly distributed in  $[0.1, 0.5]$  kilometer and  $[10, 50]$  slots respectively. The total amount of additional trip demands for simulating asymmetric demands per day is indexed to the total amount of normal trip demands with a variable ratio  $\eta$ , which represents the asymmetric demand level. The additional trip demands generation rate  $d_c(t, c_m) = \epsilon_{c_m} \eta d_n(t)$  for all  $t \in \mathbf{T}_{c_m}$  is further linked to the parameter  $\epsilon_{c_m}$ , which is determined by the scope  $r_c(c_m)$  but influenced by other random factors. We define  $\sum_{c_m \in \mathbf{C}} \sum_{t \in \mathbf{T}_{c_m}} d_c(t, c_m) = \eta P_n$ . Therefore, the total number of trip demands for one day is finally defined by  $P = P_n(1 + \eta)$ .

We define the demand pattern of congestion areas in *weekday circumstance* is mainly composed of the demand pattern  $c$  with small level of demand pattern  $a$  and  $b$ , and the demand pattern of congestion areas in *weekend circumstance* consists of equal levels of demand patterns  $a$  and  $b$ . In this paper, under the weekday circumstance and the weekend circumstance, the BSS will include  $1 \sim 6$  and  $1 \sim 18$  congestion areas in each period respectively when the asymmetric demand level  $\eta$  is from 0% to 60%.

### C. Comparison Algorithms

Firstly we propose a benchmark algorithm called Greedy Station Selection algorithm (GSS). In GSS, the users will always select the nearest available station, and neither consider

the total time cost nor take the conflicts with other users into account. GSS algorithm can be seen as the route decision of a short-sighted user *with only the real-time station availability information*.

The second comparison algorithm is the Greedy Trip Planning algorithm (GTP) in [16]. In [16], the BTP problem is mapped to the *weighted k-set packing problem solution* [49], which is NP-hard. The GTP algorithm aims to maximize the system utility in the static large-scale problem, while running in a computationally efficient manner. The GTP is a centralized algorithm that can evade the conflicts on user decision.

The third comparison algorithm is one of the state-of-the-art algorithms called Bike Trip Selection (BTS) in [4]. Zhang transferred the BTS game into the *Symmetric Network Congestion Game* [50] within several steps, and proved that all the participants can reach a Nash Equilibrium in finite turns to find a nearly-best solution *in a distributed way*. The evaluation part in [4] showed that BTS policy achieved an approximate rate of 97% under a smooth flow of traffic, and every participants can achieve a Nash Equilibrium within 9 iterations. It performs well in both utility and time complexity aspects.

For fairness, all these four algorithms share a same utility function with same penalty parameters fairly. The utility function  $g(T)$  is set to be  $g(T) = T_{\max} - T$ , and the penalties are divided into two classifications. The abandoned users (i.e., the users reach the longest-waiting-time or without available stations) will return a large negative utility  $-\alpha$  for the inability of finishing their trips and the extra economic loss (e.g., unable to return the bike in time). And the total travel time overtime user (i.e.,  $\tilde{\rho}_j^k > 1$ ) will return a small negative utility  $-\frac{1}{3}\alpha$  for the wastage of time, as these users can still finish their trip successfully and enjoy some benefits of BSS (e.g., save their strength).

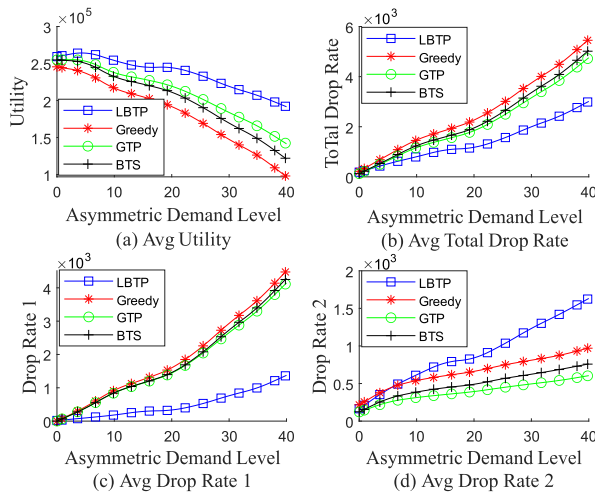
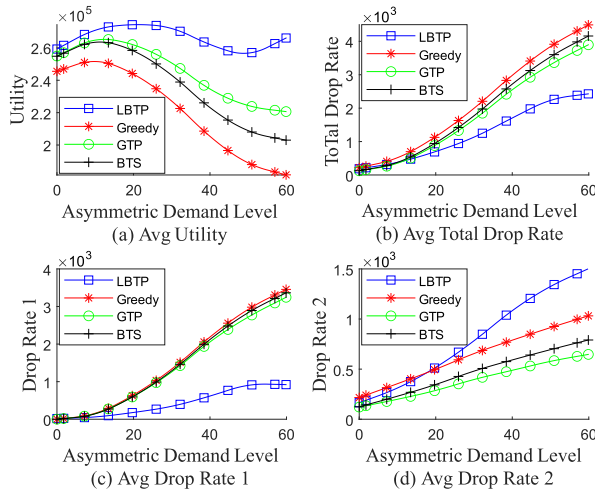
Although planning for the total travel time overtime user would not increase the utility, our LBTP policy is designed to persistently serve the users and *allocate resource* to them until they reach their destination, which is a qualified BTP service provider suppose to do. This brings LBTP algorithm more burden than other algorithms.

All the algorithms are run under the same circumstance without extra redistribution budget. All the simulations are run under 10 random maps, and every result is an average for 20 times simulation.

### D. Simulation Results Without Fairness Guarantee

We simulate all the four algorithms under the weekday and weekend circumstances separately. All the four algorithms are evaluated by the *system utility* and the *total drop rate*, the latter of which is divided into “user drop rate for waiting timeout/no available station” (drop rate 1) and “user drop rate for total travel time timeout” (drop rate 2) for better analysis.

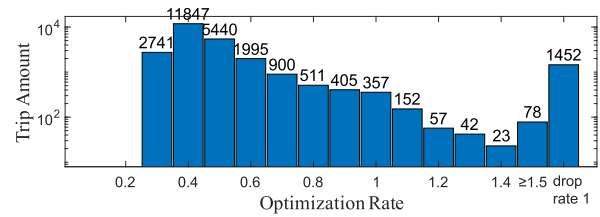
1) *Weekday Circumstance*: Fig. 4(a) shows the average system utility of the four algorithms under the weekday circumstance. We can see the LBTP algorithm outperforms the other three algorithms significantly, and the superiority even holds when the asymmetric demand level is low ( $\eta \leq 5\%$ ).

Fig. 4. Performance under the *weekday* circumstance.Fig. 5. Performance under the *weekend* circumstance.

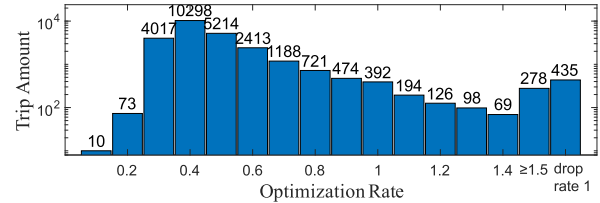
This reveals that the LBTP algorithm performs well under the normal traffic scale, for the waiting-time-aware character provides more available stations for planning.

The LBTP algorithm stands out from other three algorithms on the total user drop rate, which consists of a poor performance in the drop rate 2 but an outstanding advantage in the drop rate 1, as shown in Fig. 4(b), (c), (d). The results are not surprising because the LBTP algorithm aims to sacrifice some individual utility (i.e., some users have to make a detour) for preventing more users from suffering the waiting time timeout or helping other users save more time. When some users make a detour, they have a higher probability for total travel time timeout, which can be regarded as an inevitable price for decreasing the drop rate 1. These convert into a superiority on both the total user drop rate and the system utility.

2) *Weekend Circumstance*: Fig. 5(a) shows the average system utility of the four algorithms under the weekend circumstance. Similarly, the LBTP algorithm's performance surpasses the other three algorithms on system utility, and the superiority under low asymmetric demand level ( $\eta \leq 5\%$ )



(a) Trip optimization rate distribution of BTS algorithm.



(b) Trip optimization rate distribution of LBTP algorithm.

Fig. 6. Trip optimization rate distribution.

still holds good. We emphasize that the densely distributed congestion areas with less additional asymmetric demands in each area under weekend circumstance may not seriously reduce the system utility. Such reasons cause the fluctuation on the total drop rate in Fig. 5(b), which is reflected to the fluctuation on average utility in Fig. 5(a).

3) *Tradeoff Between the Individual Utility and System Utility*: To further evaluate the tradeoff between the individual utility and system utility (i.e., for an unobstructed system, how many of individuals need to have longer travel time, and how long the extra travel time is), we choose the BTS algorithm as a baseline, in order to find out the trip optimization rate distribution differences. We use the weekend circumstance and let the asymmetric demand level  $\eta = 30\%$ . The results are shown in Fig. 6.

The results in Fig. 6 can be divided into three parts: (1) When the trip actual optimization rate  $\tilde{\rho} \geq 1.2$ , the trip amount of LBTP algorithm well exceeds the BTS algorithm, which is the result of the users who detour to alleviate the imbalance of the system. (2) When the optimization rate is between 0.4 and 1.1, the relationship between the trip amount of LBTP and BTS algorithms fluctuates, due to the unavailable station amount differences and users' redistribution duties differences. (3) When  $\tilde{\rho} \leq 0.3$ , the trip amount of LBTP algorithm surpasses the BTS algorithm, which is because the unobstructed system enables more users to choose the nearest station.

### E. Fairness Evaluation

In order to meet the practical deployment requirements, the fairness guarantee is further applied into the evaluation part. To achieve a long-sighted user model, one time simulation is extended to 7 days and the total number of users is reduced to 7000. Under such settings, every user is designed to finish 20 trips during one time simulation, which provides sufficient amount of trips for each user to evaluate the fairness among them. We choose the weekday circumstance, and an inadequate redistribution (no more than  $\frac{1}{5}o_i^{\max}$  of corresponding station  $b_i$ ) is provided at every midday and

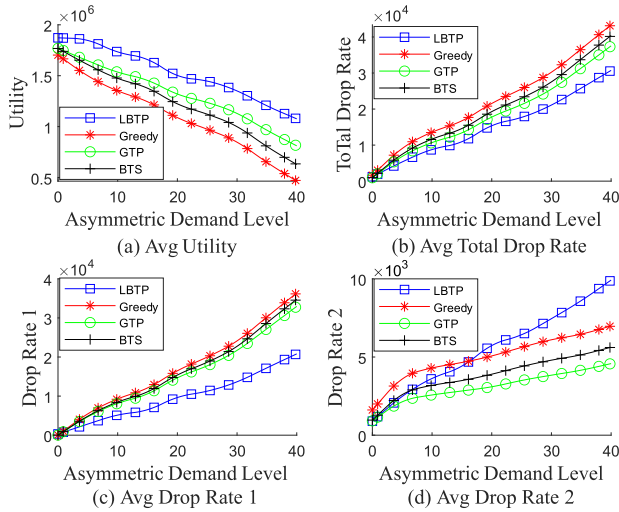


Fig. 7. Performance with fairness guarantee under the weekday circumstance.

midnight. Moreover,  $e = 500$  and  $\hat{\rho} = 0.4 + 0.3 * \eta$  are finally used, and the simulated results of the system utility and the user drop rate are shown in Fig. 7.

The performances of both the system utility and the total user drop rate of LBTP are a little weaker than without fairness guarantee, which is not hard to understand, for the additional guarantee of fairness may affect the performance. The influence significantly reflects in the performance regression of drop rate 1. As a tradeoff, the performance of drop rate 2 improves a little.

To further explore the fairness degree (i.e., the average actual optimization rate) of users, we simulate the LBTP policy under the weekday circumstance and set  $\eta = 0.3$ , which is a perfect circumstance with a proper station congestion scale and user type differences. Furthermore, to stress the user type diversity and focus on the actual optimization rate variances among different types of users, we set  $\hat{\rho} = 0.5$  and assume that 60% of the trips caused by asymmetric demand (i.e., the additional trips added for simulating asymmetric demands) are allocated to one group of users, while 60% of the trips caused by normal demands (i.e., the trips of normal demands) are allocated to another group of users, and these two groups of users will never overlap. Above settings distinguish the working users from the normal users during weekdays, where the normal users are more likely to detour for the working users to alleviate the station congestion areas. The fairness among such two kinds of users is under our major concern.

The final results are shown in Fig. 8. Fig. 8(a) and 8(b) represent the distribution of the average actual optimization rate of users under the no-fairness-guarantee circumstance and the fairness-guarantee circumstance separately. The values of the average actual optimization rate  $\tilde{\rho}$  in Fig. 8(b) appreciably gathered around 0.5 when comparing with Fig. 8(a), and the amount of users with poorer performance on average actual optimization rate ( $\tilde{\rho} \geq 0.6$ ) in Fig. 8(b) is reduced by about one-third. The improvement is especially significant when focusing on the user whose actual optimization rate is extremely poor ( $\tilde{\rho} \geq 0.8$ ). This achieves our goal that the

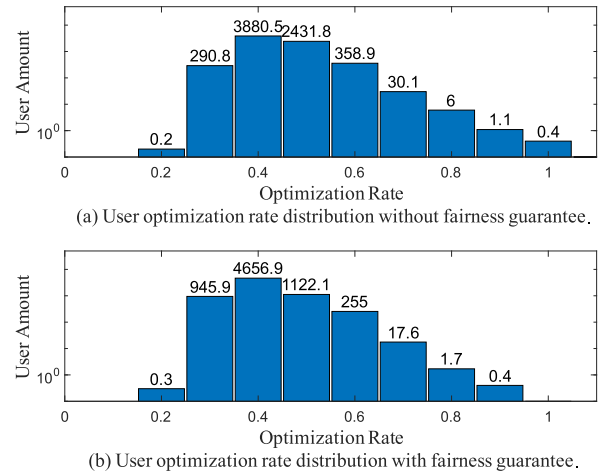


Fig. 8. User optimization rate distribution.

TABLE III  
COMPUTATIONAL TIME

User Number $N$	LBTP (ms)	GSS (ms)	GTP (ms)	BTS (ms)
170	3,110	183	1,367	2,733
510	4,550	593	9,467	8,810
816	8,648	917	22,681	15,289

number of the users with extremely poor experience can be reduced significantly to improve the user experience of the BSS.

#### F. Computational Time Evaluation

The test platform is a quad-core PC compatible of the Intel(R) Core(TM) i5-5200U CPU, and each core processor of the CPU is 2.20GHz. The algorithm is programmed in Python language, and we select PyCharm as IDE.

We randomly generate 170, 510 and 816 sequential trip demands in 10 slots during the simulation to evaluate their computational time. The first 170 sequential trip demands are generated during the off-peak hours, when the asymmetric demand level is 0%. The second 510 sequential trip demands are generated during the peak hours, when the asymmetric demand level is 0%. The third 816 sequential trip demands are generated during the peak hours, when the asymmetric demand level is 60%. The results are shown in Table III. From the results in Table III, the GSS algorithm, which is based on greedy algorithm, has a clear advantage on the computational time. But remember that GSS has a poor performance in both the system utility and total drop rate of users. On the contrary, as mentioned before, the GTP algorithm sacrifices running time efficiency for a nearly-best solution. The LBTP and BTS algorithms are both distributed solutions, and each one has tolerable time efficiency even under high demand level.

The LBTP policy is a distributed policy, which can be divided into the server-side algorithm running on the server, and the user-side algorithm running on user's cellphone. The computational time of LBTP policy could be improved, as we run both the server-side algorithm and the user-side algorithm together during the simulation.

## VII. CONCLUSION

In this paper, we have proposed a new real-time BTP policy, named Lyapunov-based Bike Trip Planning (LBTP) policy. The LBTP policy can dynamically maintain an efficient and stable BSS, while achieving self-organizing redistribution without requiring any historical usage information or extra budget. Moreover, the LBTP policy is waiting-time-aware that allows users to queue at stations, and the longest-waiting-time guarantee and other general user experience requirements are all taken into account. Finally the extra travel time fairness guarantee among users is considered for the practical deployment requirements. We evaluate our policy under different demand patterns with diverse traffic scales and asymmetric demand levels. The results exhibit the superior performance of the LBTP policy on both the system utility and the service rate of users when running in a computationally efficient manner, especially under the serious station congestion situations.

Regarding possible extensions of the proposed policy, the efficiency of the operation of our policy combined with other relocation schemes will surely be investigated. Using the regular daily flow pattern given by the prediction methods, the truck-based redistribution scheme can be applied to do the redistribution work at a macro-level in advance, while the proposed policy can deal with the unpredictable events in the real time as a supplement. Even though the truck-based schemes seem to require extra budget, it could reduce the average travel time for users and the amount of unsatisfied users. In the future, we will focus on a comprehensive work combines our LBTP policy with the truck-based redistribution scheme.

## REFERENCES

- [1] P. DeMaio, "Bike-sharing: History, impacts, models of provision, and future," *J. Public Transp.*, vol. 12, no. 4, pp. 41–56, Dec. 2009.
- [2] X. Yao, X. Shen, T. He, and S. H. Son, "Demand estimation of public bike-sharing system based on temporal and spatial correlation," in *Proc. 4th Int. Conf. Big Data Comput. Commun. (BIGCOM)*, Aug. 2018, pp. 60–65.
- [3] F. Huang, S. Qiao, J. Peng, and B. Guo, "A bimodal Gaussian inhomogeneous Poisson algorithm for bike number prediction in a bike-sharing system," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 8, pp. 2848–2857, Aug. 2019.
- [4] J. Zhang, P. Lu, Z. Li, and J. Gan, "Distributed trip selection game for public bike system with crowdsourcing," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 2717–2725.
- [5] J. W. Yoon, F. Pinelli, and F. Calabrese, "Cityride: A predictive bike sharing journey advisor," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage.*, Jul. 2012, pp. 306–311.
- [6] C. Contardo, C. Morency, and L.-M. Rousseau, *Balancing a Dynamic Public Bike-Sharing System*, vol. 4. Montreal, QC, Canada: Cirrelet, 2012.
- [7] J. Pfrommer, J. Warrington, G. Schilbach, and M. Morari, "Dynamic vehicle redistribution and online price incentives in shared mobility systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1567–1578, Aug. 2014.
- [8] J. Wu, "Challenges and opportunities in algorithmic solutions for rebalancing in bike sharing systems," *Tsinghua Sci. Technol.*, vol. 25, no. 6, pp. 721–733, Dec. 2020.
- [9] R. J. Szczerba, P. Galkowski, I. S. Glickstein, and N. Ternullo, "Robust algorithm for real-time route planning," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, no. 3, pp. 869–878, Jul. 2000.
- [10] P. Sanders and D. Schultes, "Engineering fast route planning algorithms," in *Proc. Int. Workshop Experim. Efficient Algorithms*. Berlin, Germany: Springer, 2007, pp. 23–36.
- [11] H. Bast *et al.*, "Route planning in transportation networks," in *Algorithm Engineering*. Cham, Switzerland: Springer, 2016, pp. 19–80.
- [12] J. E. Froehlich, J. Neumann, and N. Oliver, "Sensing and predicting the pulse of the city through shared bicycling," in *Proc. Morgan Int. Joint Conf. Artif. Intell. (IJCAI)*, 2009, pp. 1420–1426.
- [13] J. Zhang and P. S. Yu, "Trip route planning for bicycle-sharing systems," in *Proc. IEEE 2nd Int. Conf. Collaboration Internet Comput. (CIC)*, Nov. 2016, pp. 381–390.
- [14] T. Preisler, T. Dethlefs, and W. Renz, "Self-organizing redistribution of bicycles in a bike-sharing system based on decentralized control," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Oct. 2016, pp. 1471–1480.
- [15] J. Xu, J. Xu, G. Cao, H. Xu, M. Xu, and N. Zheng, "Towards optimal free-of-charge trip planning in bike-sharing systems," in *Proc. 18th IEEE Int. Conf. Mobile Data Manage. (MDM)*, May 2017, pp. 92–101.
- [16] Z. Li, J. Zhang, J. Gan, P. Lu, and F. Lin, "Large-scale trip planning for bike-sharing systems," in *Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2017, pp. 328–332.
- [17] J. Liu *et al.*, "Station site optimization in bike sharing systems," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 883–888.
- [18] S. Ruffieux, E. Mugellini, and O. A. Khaled, "Bike usage forecasting for optimal rebalancing operations in bike-sharing systems," in *Proc. IEEE 30th Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2018, pp. 854–858.
- [19] H. I. Ashqar, M. Elhenawy, M. H. Almanna, A. Ghanem, H. A. Rakha, and L. House, "Modeling bike availability in a bike-sharing system using machine learning," in *Proc. 5th IEEE Int. Conf. Models Technol. Intell. Transp. Syst. (MT-ITS)*, Jun. 2017, pp. 374–378.
- [20] C. Latinopoulos, N. Daina, and J. W. Polak, "Trust in IoT-enabled mobility services: Predictive analytics and the impact of prediction errors on the quality of service in bike sharing," in *Living in the Internet of Things: Cybersecurity of the IoT*. London, U.K.: IET, 2018, pp. 1–7.
- [21] L. Caggiani and M. Ottomanelli, "A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems," *Procedia-Social Behav. Sci.*, vol. 87, pp. 203–210, Oct. 2013.
- [22] D. Chemla, F. Meunier, and R. W. Calvo, "Bike sharing systems: Solving the static rebalancing problem," *Discrete Optim.*, vol. 10, no. 2, pp. 120–146, May 2013.
- [23] T. Raviv, M. Tzur, and I. A. Forma, "Static repositioning in a bike-sharing system: Models and solution approaches," *EURO J. Transp. Logistics*, vol. 2, no. 3, pp. 187–229, Aug. 2013.
- [24] B. Legros, "Dynamic repositioning strategy in a bike-sharing system; how to prioritize and how to rebalance a bike station," *Eur. J. Oper. Res.*, vol. 272, no. 2, pp. 740–753, Jan. 2019.
- [25] L. Caggiani, R. Camporeale, M. Ottomanelli, and W. Y. Szeto, "A modeling framework for the dynamic management of free-floating bike-sharing systems," *Transp. Res. C, Emerg. Technol.*, vol. 87, pp. 159–182, Feb. 2018.
- [26] C. Fricker and N. Gast, "Incentives and redistribution in bike-sharing systems with stations of finite capacity," *Springer EURO J. Transp. Logistics*, vol. 5, no. 3, pp. 1–31, 2013.
- [27] A. Wasserhole, V. Jost, and N. Brauner, "Pricing techniques for self regulation in vehicle sharing systems," *Electron. Notes Discrete Math.*, vol. 41, pp. 149–156, Jun. 2013.
- [28] Y. Luo, W. Dou, H. Yan, L. Liu, and S. Lu, "An automated planning and scheduling method of shared bikes based on reward and punishment mechanism," in *Proc. 17th Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci. (DCABES)*, Oct. 2018, pp. 276–279.
- [29] D. Gimon, "Bike commuters contribution to balance shared bike systems during peak load," in *Proc. IEEE Int. Smart Cities Conf. (ISC)*, Sep. 2018, pp. 1–2.
- [30] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.
- [31] E. Leonardi, M. Mellia, M. A. Marsan, and F. Neri, "Optimal scheduling and routing for maximum network throughput," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1541–1554, Dec. 2007.
- [32] M. J. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 2915–2934, Jul. 2006.
- [33] M. J. Neely, "Delay-based network utility maximization," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 41–54, Feb. 2013.
- [34] A. Sinha and E. Modiano, "Optimal control for generalized network-flow problems," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 506–519, Feb. 2018.
- [35] J. Zhang, A. Sinha, J. Llorca, A. Tulino, and E. Modiano, "Optimal control of distributed computing networks with mixed-cast traffic flows," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 1880–1888.

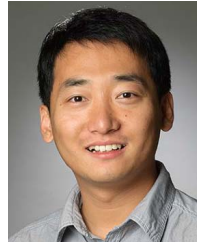
- [36] M. Lu, S.-C. Hsu, P.-C. Chen, and W.-Y. Lee, "Improving the sustainability of integrated transportation system with bike-sharing: A spatial agent-based approach," *Sustain. Cities Soc.*, vol. 41, pp. 44–51, Aug. 2018.
- [37] Z. Liu, X. Jia, and W. Cheng, "Solving the last mile problem: Ensure the success of public bicycle system in Beijing," *Procedia-Social Behav. Sci.*, vol. 43, pp. 73–78, Jan. 2012.
- [38] Q. Li, R. Fan, and N. Li, "Research on big data driven large-scale bike-sharing systems and associated bike rebalancing," *Chin. J. Appl. Probab. Statist.*, vol. 38, no. 5, pp. 499–518, 2016.
- [39] M. A. Razzaque and S. Clarke, "Smart management of next generation bike sharing systems using Internet of Things," in *Proc. IEEE 1st Int. Smart Cities Conf. (ISC)*, Oct. 2015, pp. 1–8.
- [40] S. Garcia-Perez *et al.*, "Alternative mobility system using IoT and smart-cities approaches through the use of the bike for a sector of the city of bogota within a simulated environment," in *Proc. Smart City Symp. Prague (SCSP)*, Jun. 2020, pp. 1–6.
- [41] C. Bo, T. Jung, X. Mao, X.-Y. Li, and Y. Wang, "SmartLoc: Sensing landmarks silently for smartphone-based metropolitan localization," *EURASIP J. Wireless Commun. Netw.*, vol. 2016, no. 1, pp. 1–17, Dec. 2016.
- [42] A. Fabrikant, C. Papadimitriou, and K. Talwar, "The complexity of pure Nash equilibria," in *Proc. 36th Annu. ACM Symp. Theory Comput. (STOC)*, 2004, pp. 604–612.
- [43] B. Vöcking and R. Aachen, "Congestion games: Optimization in competition," in *Proc. Algorithms Complex. Durham Workshop (ACiD)*, 2006, pp. 9–20.
- [44] M. J. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1728–1736.
- [45] Y. Li *et al.*, "MP-coopetition: Competitive and cooperative mechanism for multiple platforms in mobile crowd sensing," *IEEE Trans. Services Comput.*, early access, May 16, 2019, doi: [10.1109/TSC.2019.2916315](https://doi.org/10.1109/TSC.2019.2916315).
- [46] Y. Li *et al.*, "PTASIM: Incentivizing crowdsensing with POI-tagging cooperation over edge clouds," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4823–4831, Jul. 2020.
- [47] Y. Li, F. Li, S. Yang, P. Zhou, L. Zhu, and Y. Wang, "Three-stage stackelberg long-term incentive mechanism and monetization for mobile crowdsensing: An online learning approach," *IEEE Trans. Netw. Sci. Eng.*, early access, Feb. 5, 2021, doi: [10.1109/TNSE.2021.3057394](https://doi.org/10.1109/TNSE.2021.3057394).
- [48] I.-L. Wang and C.-W. Wang, "Analyzing bike repositioning strategies based on simulations for public bike sharing systems: Simulating bike repositioning strategies for bike sharing systems," in *Proc. 2nd IIAI Int. Conf. Adv. Appl. Informat.*, Aug. 2013, pp. 306–311.
- [49] B. Chandra and M. M. Halldórsson, "Greedy local improvement and weighted set packing approximation," *J. Algorithms*, vol. 39, no. 2, pp. 223–240, May 2001.
- [50] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, May 1996.



**Junheng Wang** received the B.E. degree in computer science and technology from the Beijing University of Technology, Beijing, China, in 2018. He is currently pursuing the M.S. degree with the School of Computer Science, Beijing Institute of Technology, Beijing. His research interests include mobile crowd sensing and game theory.



sensing, crowd sensing, and mobile computing. She is a member of ACM. Her papers won best paper awards from the IEEE MASS in 2013, the IEEE IPCCC in 2013, the ACM MobiHoc in 2014, and the Tsinghua Science and Technology in 2015.



computing, and network function virtualization.

**Fan Li** (Member, IEEE) received the B.Eng. and M.Eng. degrees in communications and information system from the Huazhong University of Science and Technology, China, the M.Eng. degree in electrical engineering from the University of Delaware, Newark, DE, USA, and the Ph.D. degree in computer science from the University of North Carolina at Charlotte, Charlotte, NC, USA. She is currently a Professor with the School of Computer Science, Beijing Institute of Technology, China. Her current research interests include wireless networks, smart

**Song Yang** (Member, IEEE) received the Ph.D. degree from the Delft University of Technology, The Netherlands, in 2015. From August 2015 to July 2017, he worked as a Postdoctoral Researcher with the EU FP7 Marie Curie Actions CleanSky Project, Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany. He is currently an Associate Professor with the School of Computer Science, Beijing Institute of Technology, China. His research interests include data communication networks, cloud/edge



**Youqi Li** received the Ph.D. degree in computer science and technology from the Beijing Institute of Technology, Beijing, China, in 2020. He is currently a Postdoctoral Researcher with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include mobile crowd sensing, privacy, game theory, and adversarial machine learning.



**Yu Wang** (Fellow, IEEE) received the B.Eng. and M.Eng. degrees in computer science from Tsinghua University, and the Ph.D. degree in computer science from the Illinois Institute of Technology, Chicago, IL, USA. He is currently a Professor with the Department of Computer and Information Sciences, Temple University. He has published more than 200 articles in peer-reviewed journals and conference papers. His research interests include wireless networks, smart sensing, and mobile computing. He has been an ACM Distinguished Member since

2020. He was a recipient of the Ralph E. Powe Junior Faculty Enhancement Awards from the Oak Ridge Associated Universities in 2006 and the Outstanding Faculty Research Award from the College of Computing and Informatics, University of North Carolina at Charlotte, in 2008. He has served as the General Chair, the Program Chair, and the Program Committee Member for many international conferences, such as IEEE IPCCC, ACM MobiHoc, IEEE INFOCOM, IEEE GLOBECOM, and IEEE ICC, and an Editorial Board Member for several international journals, including IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.