Distributed Trip Selection Game for Public Bike System with Crowdsourcing

Jianhui Zhang*, Pengqian Lu, Zhi Li, Jiayu Gan
College of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018 China
*Corresponding author *e*-mail: jhzhang@ieee.org

***Abstract*—Public Bike Systems (PBSs) offer convenient and green travel service and become popular around the world. In many cities, the local governments build thousands of fixed stations for PBS to alleviate the city traffic jam and solve the last-mile problem. However, the increasing use of PBSs leads to new congestion problems in the form that users have, such as *no bike to rent* or *no dock to return* the bike. Further, users wish to receive assistance on deciding how to select bike trips with minimal time cost while taking congestion into account. Meanwhile, *crowdsourcing* attracted increasing attention in recent years. This paper applies it to help users share information and select bike trips before the bikes or docks are occupied. An interesting and important problem is how to help users select bike trips so that the time consumed on the trips can be minimized. We model the problem as a Bike Trip Selection (BTS) game which is shown to be equivalent to the symmetric network congestion game. This equivalence allows us to design a BTS algorithm by which the users can find at least one Nash Equilibria (NE) distributively. Furthermore, this paper evaluates the algorithm based on real datasets collected from the PBS of Hangzhou City in China. We also design a BTS system including an Android APP and a server to conduct the experiment for the distributed BTS algorithm in practice.**

***Index Terms*—Bike Trip Selection; Public Bike System; Crowdsourcing; Game Theory**

I. Introduction

As a green and convenient transportation mode, Public Bike System (PBSs) have been widely promoted by many governments around the world, such as in Chicago and New York of USA [1][2], many cities of China [3]. It brings great benefit to last-mile transportation in busy cities. At the same time, PBSs also lead to new and interesting research questions such as bike mobility prediction [3] and bike lane planning [4].

PBSs have some fixed bike stations, each of which has a limited amount of docks or bikes, in which users can rent or return bikes. As the use of PBS grows, congestion will occur in the form that a user may not be able to rent or return her bike after a walking or biking to the bike station. For example, in Figure 1, the bike stations A and C have no available bike and empty dock so that user cannot rent bike from A or return bike to C. One interesting and practical challenge is how user can find the bike trip to avoid getting into this congestion situation with no bike to rent or no dock to return bike. Some studies in the literature suggest to estimate the resource requirement for bikes and docks [3] or to redistribute the unbalanced bike resources by transporting bikes among stations with trucks [5][6]. These methods cannot help user find available bike and empty dock in time. Another challenge

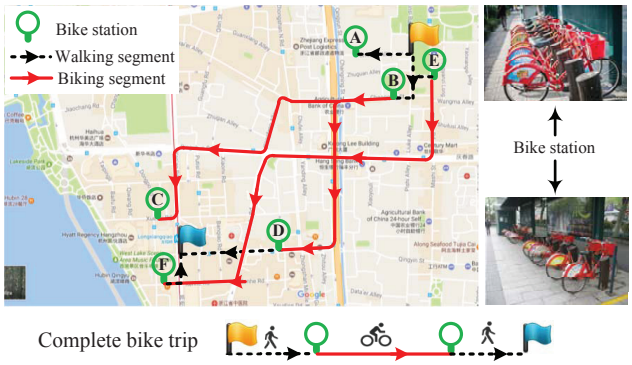


Fig. 1. An example for bike trip selection in PBS. A complete bike trip contains three segments.

is how users make the *bike trip selection* (BTS) to minimize the time cost on their trips so that they can enjoy the benefit of PBS. For example, there are two trips, one going through B and D and the other going through E and F in Figure 1. The later one is longer than the first one on distance but it takes less time because of its shorter walking distance and relative lower time cost. Yoon *et al.* present the personal journey advisor application to choose the pair of stations for users to travel to minimize the travel time with high probability [7]. But it needs the information from neighboring stations and seasonal trend to predict the bike available probability. However, users are usually willing to minimize their own time cost selfishly.

Benefiting from the wide applications of wireless network and smartphone, crowdsourcing can help users share more updated information of PBS and share it through the Internet [8][9][10]. This paper adopts crowdsourcing to help users share information of the PBS and then select bike trips before bikes and docks are occupied. It ensures users to know the information of PBS earlier than those system offering real-time information and to avoid the complex calculation such as on prediction as the previous works. When helping user select bike trip, this paper further introduces a complete bike trip as shown in Figure 1, which includes not only the segment from initial bike station to target one but also two additional walking segments, one from user's source location to the initial bike station and the other from the target bike station to her terminal location. In this way, our model of the complete bike trip is much closer to practice. We formulate the above challenges as

a BTS problem and solve it by mapping to a BTS game, and prove that the game is equivalent to the *symmetric network congestion game* [11][12]. This equivalence ensures the existence of at least one Nash Equilibrium (NE) and guarantees the convergence to NE in finite iterations. Furthermore, this paper develops a BTS algorithm for the game to find the NE and allows each user to select bike trip distributively. Real data collected from the PBS of Hangzhou City in China is used to evaluate the performance of the BTS algorithm. We also design a prototype crowdsourcing BTS system including an Android APP for smartphones and a server to help users select bike trips. The contributions of this paper is as follows:
1) Bike trip selection modeling. This paper considers the complete bike trip including three segments, which is closer to practice than existing works which only consider the trip between bike stations. We formulate the BTS problem so that user can select the bike trip in deterministic way instead of probabilistic one beforehand.
2) Game formulation and performance analysis. We formulate the BTS problem as the BTS game and map it to a symmetric network congestion game. So it has at least one pure NE and the Finite Improvement Property (FIP).
3) Distributed algorithm design. We propose the BTS algorithm for the game to find bike trips for users distributively. It is guaranteed to converge to at least one NE.
4) Real data and system based evaluation. We conduct two experimental evaluations. In the first one, the BTS algorithm is compared to one benchmark algorithm and real datasets collected from the PBS in Hangzhou City of China. The other invites some volunteers to implement the distributed BTS algorithm on our BTS system, which includes one server and Android APP, and is compared to the bike utilization with the aid of electronic map.

Most symbols used in this paper are summarized in Table I.

TABLE I
SYMBOL AND MEANING

Sym.	Description	Sym.	Description
b	Bike station	B	Bike station set
u	User/player	U	User set
r	Bike trip/strategy	T	Bike trip set
o	# of docks	n	# of bikes
M	# of users	N	# of stations
ω	Edge weight	l	Location/position
c	Cost	C	Overall cost
s	Strategy profile	$\mathbf{S}$	Strategy set

II. System Model and Problem Formulation

A. System Model

This paper considers a mobile crowdsoucing system that involves the collection of key state information for the PBS, including users' information such as their source and terminal locations, and the bike stations' information such as available bikes and docks. We propose a crowdsourcing platform, called BTS system, including an Android APP and a server. Once user installs the Android APP on her smartphone, she can use the map in the APP to look for available bike stations to rent and return bikes. The APP can also find a bike trip for her if she requests the server. Suppose that the PBS has N stations, b_k, $k = 1, \cdots, N$, which are also used to indicate their locations. There is a set U of users, u_j, $j = 1, \cdots, M$, who request the server to help them select bike trip.

Bike station. Each bike station b_k has a fixed number of docks. On each station there are some bikes and/or empty docks. User can rent bike from one station and return it to another. Denote the number of bikes and empty docks by n_k and o_k in each station respectively. Both bikes and docks in each bike station are limited resources, and it can cause the congestion when too many users rent or return bikes from or to the same bike station.

Bike trip. We draw a bike trip selection graph in Figure 2 to illustrate the bike trips of multiple users. A complete bike trip r composes of three segments: the first one from user's source location to an initial bike station, the second from the initial bike station to a target bike station, and the third from the target bike station to user's terminal location, which are denoted by e^1, e^2 and e^3, *i.e.*, $r = \{e^1, e^2, e^3\}$. Each bike trip includes the initial and target bike stations, *e.g.*, the stations B and D in Figure 1, and the user u's source and terminal locations. Denote the user's source and terminal locations by $l_i(u)$ and $l_t(u)$ as Figure 2. The trips that user u_j can select compose a trip set T_j. The trip sets of all users merge to a united trip set $\mathbf{S}$, $\mathbf{S} = \cup_{u_j \in U} T_j$.

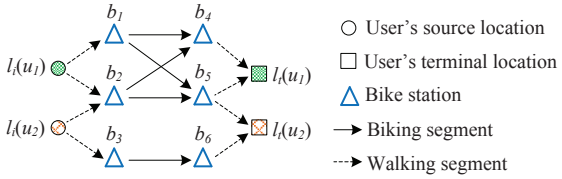


Fig. 2. Bike trip selection (BTS) graph.

Trip cost function. The overall time cost of a trip includes the time for both walking and biking. The time of a walking segment depends on the walking velocity and segment distance, while the time of a biking segment depends on the biking velocity and its distance. Each segment of all bike trips can be assigned with a weight ω, which is the time to travel the segment. Notice that there may be multiple paths between a pair of locations, such as between one user's source location to one initial bike station. User prefers the one with the lowest time cost in order to save time. So this paper supposes that there is only one path with the minimum distance between user source location to initial bike station. Similar to the location pairs from initial bike station to target bike station and from target bike station to user terminal location. Furthermore, the time cost of each bike trip is also determined by the status of stations, *i.e.*, n and o, on it. Each station b_k has two roles: the initial and target bike station, *i.e.*, bike renting and bike returning. Let $x_k\langle x_k^r, x_k^t\rangle$ denote the number of users to share b_k. When b_k is an initial bike station, $x_k = x_k^r$.

Otherwise, $x_k = x_k^t$. We define two functions: *rent cost function* and *return cost function* to capture the dependency on the station status. The rent cost function indicates that the trip cost is quite high and denoted by a huge value when there is no available bikes to rent. Otherwise, it's zero and given as the following equation.

$$g_r(x_k) = \begin{cases} 0, & x_k \leq n_k \\ \infty, & \text{otherwise} \end{cases} \tag{1}$$

Similarly, the return cost function indicates whether the bike station b_k has available empty docks to return bikes and is given as follows.

$$g_t(x_k) = \begin{cases} 0, & x_k \leq o_k \\ \infty, & \text{otherwise} \end{cases} \tag{2}$$

Summarizing the above discussions, the time cost of bike trip r includes the time to travel its three segments and those given by the rent and return cost functions as the following function:

$$c(r) = \sum_{e^i \in r} \omega(e^i) + g_r(x_k) + g_t(x_k) \tag{3}$$

B. Problem Formulation

This paper assumes that each user u_j acts selfishly and aims at choosing the trip r_j to minimize the time cost of her own bike trip. Given the set U of users, the set B of bike stations, each of which has certain numbers of bikes and docks, the problem is how each user $u_j \in U$ selects her bike trip r_j selfishly so that the overall cost of her trip, $c(r_j)$, can be minimized. This paper calls it as the BTS problem.

III. Bike Trip Selection Game

This section formulates the BTS problem as the BTS game, and proves that every BTS game is equivalent to a symmetric network congestion game [11][13]. The BTS game is then proved to have at least one NE and the FIP.

A. Symmetric Network Congestion Game

In the *congestion game*, players share resources, and the cost of each player depends on the resources she chooses and the number of players choosing the same resource [14]. Congestion game, like potential game, guarantees to have pure NE [8], and has been extensively applied in the context of networks, such as routing and spectrum allocation [15][16]. The *network congestion game* is the networked version of the congestion game, and has good properties such as the existence of NE and the FIP [12][13]. With the FIP, a game can reach at least one NE ultimately when players keep improving their strategies unilaterally where no other player changes her strategy at any given time. In the *network congestion game* [11], the resources are edges within a network. Each player selects a route from her source vertex to terminal vertex, and aims to minimize the cost paid for traversing congested edges. If all players use the same pair of source and destination, the network congestion game is *symmetric*, and *asymmetric* otherwise [17].

A *symmetric network congestion game* can be defined by a 5-tuples $(U, l_i, l_t, G, (d_e(\cdot))_{e \in E})$ formally, where U is a set of players; $G(V, E)$ is a directed graph; l_i and l_t are the common initial vertex and the common target vertex respectively and $d_e(\cdot)$ is a non-decreasing and non-negative cost function for each edge $e \in E$. A strategy for the symmetric network congestion game is a route from l_i to l_t. A strategy profile s is a collection of users' strategies when each user takes one strategy and can be written as $s = (r_1, \cdots, r_M)$. The overall cost of a player u_j within a strategy profile s is:

$$C_{u_j}(s) = \sum_{e \in \epsilon(r_j)} d_e(\psi_s(e)) \tag{4}$$

where $\epsilon(r_j)$ is the set of edges in route r_j and $\psi_s(e) = |\{u_j \in U : e \in \epsilon(r_j)\}|$ is the total number of players sharing the edge e under strategy s.

Definition 1 (Pure NE): A pure Nash Equilibrium (NE) is a strategy profile $s^* \in \mathbf{S}$, where any user cannot achieve lower time cost by taking unilateral strategy.

The symmetric network congestion game is equivalent to the congestion game. It preserves the FIP so that every sequence of players' asynchronous improvement steps is finite and converges to at least one pure NE. If a player can decrease her cost by changing her strategy while fixing other player strategies, it is called a *better response* [18].

Definition 2 (Best response): A best response $\{r_j, s_{-j}\}$ of a player u_j within the strategy profile s is the strategy which can decrease u_j's cost to the minimum possible value among all better responses given other players' strategies $s_{-j} = \{r_1, \cdots, r_{j-1}, r_{j+1}, \cdots, r_M\}$ in the current strategy profile s.

B. Bike Trip Selection Game

The symmetric network congestion game is similar to the BTS problem but cannot be applied to it directly. By comparing to the formulation of the BTS problem in Section II, it is different from the problem in the following aspects. 1) In a symmetric network congestion game, each player aims to minimize her cost as given in Equation (4). In the BTS problem, each user tries to minimize her trip cost in Equation (3). The two cost functions are different. 2) In the game, each player shares the edge capacity with others after she is assigned to it. In the problem, each bike trip is successful only if both the bike renting and returning are successful. In other words, the success of each complete bike trip is determined by both the initial and target bike stations. 3) In the game, a player's cost depends only on the edges of the route; whereas in the problem, a user's cost depends on both the bike stations and users' source and terminal locations. 4) In the game, the route of each player starts from the same source and ends at the same destination; whereas in the BTS problem, the users start and end at their own locations.

Converted BTS graph. In the following context, this section presents some steps to design our BTS game so as to eliminate the above difference. Firstly, we construct a new graph, called *converted BTS graph*, based on the BTS graph

in Figure 2. For each vertex in the bike station set B, we add a virtual one, and connect it to the real vertex with a link as the edge $e(b_1, b_1')$ in Figure 3. Each such link connecting with the initial bike station is assigned a weight with the number of available bikes in the station. Each link connecting with the target bike station is assigned a weight with the number of available empty docks in the station. Denote all virtual vertices by a set B'. All links connecting the initial bike stations to their virtual ones form an edge set E_i while those connecting the target bike stations to their virtual ones form an edge set E_r. For example, the edges $e(b_1, b_1') \in E_i$ and $e(b_4, b_4') \in E_r$ in Figure 3. The weight of edge $e(b_1, b_1')$ is set to be the number of available bikes in station b_1. The weight of edge $e(b_4, b_4')$ is set to be the number of available empty docks in station b_4. The cost of each edge in the set E_i is given by Equation (1) while that in the set E_r is given by Equation (2).

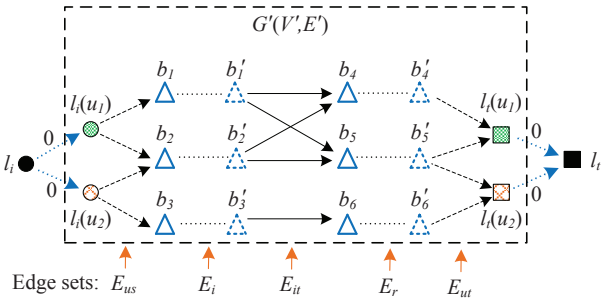


Fig. 3. The converted BTS graph attaching a common source vertex and a target vertex.

Let E_{us} denote the edge set of all links connecting the users' source locations to the initial bike stations. Let E_{ut} denote the edge set of those connecting the virtual target bike stations to the users' terminal locations. Let E_{it} denote the edge set of all links connecting the virtual initial bike stations to the target bike stations. Each edge in E_{us}, E_{ut} and E_{it} corresponds to one segment of bike trip in Figure 2. So the weight of edge is set to be the time to travel its corresponding segment. For example, the edges $e(l_i(u_1), b_1) \in E_{us}$, $e(b_4', l_t(u_1)) \in E_{ut}$, and $e(b_1', b_4) \in E_{it}$ in Figure 3. The weight of the edge $e(l_i(u_1), b_1)$ is the walking time to travel from $l_i(u_1)$ to b_1.

Denote the sets of all users' source and terminal locations by V_S and V_T respectively. Define a new vertex set V' to contain the vertex sets, B, B', V_S and V_T. Define a new edge set E' to contain the edge sets, E_i, E_r, E_{us}, E_{ut} and E_{it}. It leads to the converted BTS graph $G'(V', E')$ as shown in the dotted box of Figure 3.

BTS game. The converted BTS graph helps us develop the BTS game for the BTS problem. We then show that the game is equivalent to the symmetric network congestion game. It takes four steps to develop the BTS game, and we state them with the help of Figure 3 as follows.

1) Define the set U of users as the set of players of BTS game.

2) Add a virtual common source vertex l_i and a virtual common target vertex l_t. Let l_i connect to all users' source locations and l_t connect to all users' terminal locations. Set the weights of all edges connecting l_i or l_t as zero as the example in Figure 3.

3) Take the edge set E' as the resources, from which each player selects her strategy. Recall that a bike trip r_j of user u_j is a traveling from u_j's source location to her terminal location. A u_j's strategy contains her bike trip r_j, *i.e.*, one from l_i to l_t, and going through one vertex in each of the four sets V_S, V_T, B and B', and the edges in the set E'. Without confusion, the strategy is also denoted by r_j. Define the strategy profile $s = \{r_j, u_j \in U\}$, where r_j is the strategy of player u_j. All strategy profiles form the strategy space $\mathbf{S}$.

4) Define the cost function for each edge as follows. Let the cost function for each edge connecting with l_i and l_t as $d_e(x) = 0$, $\forall x$, where x is the number of players sharing the edge e. Define the cost function for each edge in E' as follows.

$$d_e(x) = \begin{cases} \omega(e), & e \in E_{us} \cup E_{ut} \cup E_{it}, \forall x \\ g_r(x), & e \in E_i, \quad \forall x \\ g_t(x), & e \in E_r, \quad \forall x \end{cases} \tag{5}$$

where $g_r(x)$ and $g_t(x)$ are given in Equation (1) and (2) respectively. $\omega(e)$ is the weight, *i.e.*, the time cost of the edge e. With the help of Equation (3), the overall cost received by u_j within a strategy profile s is given by the following equation:

$$C_{u_j}(s) = \sum_{e \in r_j} d_e(x_s^e) \tag{6}$$

where x_s^e is the number of users to share the edge e within the strategy profile s, and $r_j \in s$.

By the above steps, we have the player set U, the graph $G'(V', E')$, the cost function for each edge $d_e(\cdot)$, the virtual common source and terminal vertices l_i and l_t. This can lead to our definition for the BTS game as follows.

Definition 3 (BTS game): A *BTS game* is given by 5-tuple $(U, G', l_i, l_t, (d_e(\cdot))_{e \in E'})$ with each player u_j choosing a strategy $r_j \in \mathbf{S}$ so as to minimize the cost in Equation (6).

Recalling the formulation of the BTS problem, we can find that the problem is consistent with the BTS game. The natural questions are if the game has NE and the FIP or not.

C. Game Isomorphism

The symmetric network congestion game has NE and FIP [11][13]. We can find these properties for the BTS game by proving that it's equivalent to the symmetric network congestion game. Two games are equivalent if there is a weak isomorphism from one to the other [12][19]. So we have the following theorem.

Theorem 1: Every BTS game is equivalent to a symmetric network congestion game.

Proof: This proof shows that the symmetric network congestion game Γ_{net} and the BTS game Γ_{BTS} are isomorphic. Recall that the two games are given by the 5-tuples $\Gamma_{net} = (U, l_i, l_t, G, (d_e(\cdot))_{e \in E})$ and $\Gamma_{BTS} =$

$(U, G', l_i, l_t, (d_e(\cdot))_{e\in E'})$ respectively. We confirm the strong isomorphism from Γ_{net} to Γ_{BTS} according to its definition in the reference [19]. Basically, the two games have the same player set U. They have the common initial and target vertices and thus are symmetric. Secondly, Γ_{net} is played on the graph $G(V, E)$ according to its definition in Section III-A. In this paper, the graph can be instantiated to be the graph G' as the example in Figure 3 since no vertex in G' has affection on the time cost of any route. The game can play on the graph $G'(V', E')$ now. For Γ_{net}, we also associate each edge $e \in E'$ with the non-decreasing and non-negative cost function as the definition in Equation (5). Thirdly, each strategy in Γ_{net} is a route r from the common source l_i to the common target l_t and thus corresponds to a bike trip in Γ_{BTS}. Let $\mathbf{S}_{net}$ denote the set of all routes from l_i to l_t in Γ_{net}. In Γ_{net}, each route is a strategy. The total cost of player u_j to play the strategy r_j within the strategy profile s is $C^{net}_{u_j}(s) = \sum_{e\in\epsilon(r_j)} d_e(\psi_s(e))$ according to Equation (4). Fourthly, let $\mathbf{S}_{BTS}$ denote the set of all bike trips from l_i to l_t in Γ_{BTS}. Since two games play on the same graph G', each route in Γ_{net} corresponds to one bike trip. The cost of bike trip r_j in Γ_{BTS} within the strategy profile s is $C^{BTS}_{u_j}(s)$ according to Equation (6). So we have $C^{net}_{u_j}(s) = C^{BTS}_{u_j}(s)$. Obviously, there is a bijection: $\mathbf{S}_{net} \mapsto \mathbf{S}_{BTS}$ so as to convert the game Γ_{net} to Γ_{BTS}. ■

Since the symmetric network congestion game has NE and FIP and is equivalent to the BTS game, we have the following corollaries.

Corollary 2: Every BTS game has the FIP and at least one pure NE.

The corollary means that any local and global optimal solutions of the BTS game can result in at least one pure strategy for NE [12]. Furthermore, each player u_j's best response in the BTS game is the shortest path from l_i to l_t, which can be found in polynomial time by Dijkstras shortest path algorithm. u_j can find her best response given a strategy profile s if other players fix their strategies within s. So we can conclude this by the following corollary.

Corollary 3: A player in the BTS game can find her best response within polynomial time.

The above corollary indicates that players can reach a NE in polynomial time if they keep asynchronously updating their strategies according to their best responses.

IV. Bike Trip Selection Algorithm

The isomorphism between the BTS game and the symmetric network congestion game allows us to design algorithm to find NE conveniently. Although there are some existing algorithms, such as the min-cost flow [11][13], for the symmetric network congestion game, they are not close to the practical status of bike trip selection. This section designs the distributed BTS algorithm.

A. Algorithm Design

Corollary 2 guarantees the convergence of our BTS algorithm. The key idea behind the algorithm is to find a bike trip strategy and update it *asynchronously* until an NE is reached by Corollary 3. Recall that this paper uses the crowdsourcing technique so our algorithm involves the Android APP and the server. The algorithm runs distributively since each user makes decision in her Android APP, which needs only a little communication with the server. The distributed BTS algorithm composes of two parts, one for user and the other for the server in Algorithm 1 and Algorithm 2 respectively. By the distributed BTS algorithm, each user uploads her personal information: her source and terminal locations, to the server, and then obtains the information of the bike stations including their locations, available bikes and empty docks, and the number of users sharing the bike stations. The APP calculates and selects the bike trip for the user locally and sends the selected trip to the server, which then updates the status of bike stations on the bike trip once it receives the request messages. Server processes messages one by one.

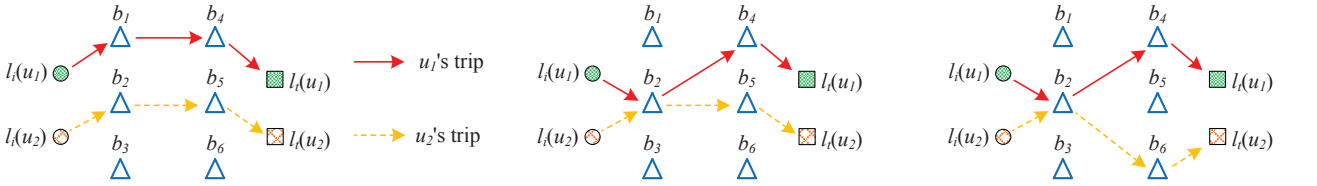
To initialize Algorithm 1, each user sends the request message to the server including the user's source and terminal locations (line 2), checks the information of bike stations including o_j, n_j and $x_j\langle x^r_j, x^t_j\rangle$, $\forall b_k \in B$ (line 3), calculates the time cost of the bike trip based on the walking and biking velocity and finds the shortest path with the distance information provided by the server (line 4). Suppose that Algorithm 2 receives a message containing a trip r, which goes through the initial bike stations b_k and the target bike station b_m. Line 5 of Algorithm 2 decreases one bike in b_k and one dock in b_m to update $x_j\langle x^r_j, x^t_j\rangle$, $\forall b_k \in B$. Algorithm 2 processes the messages of users one by one in order of arrival.

Algorithm 1 Distributed BTS Algorithm for user $u_j \in U$

Input: User u_j's personal source and terminal locations $l_i(u_j)$ and $l_t(u_j)$.
Output: Bike trip r_j.

1: **while** $\tau < \tau_{max}$ **do**
2: Request the server with a message including u_j's source and terminal locations;
3: Check the smartphone APP for the information of bike stations: o_k, n_k and $x_k\langle x^r_k, x^t_k\rangle$, $\forall b_k \in B$;
4: With the Dijkstras shortest path algorithm, find the bike trip $r_j \in T_j$ with the minimal cost;
5: Report the trip r_j including its time cost to the server;
6: $\tau += 1$.
7: **end while**

Notice that each user needs only request the available bike stations to rent or return her bike and the number of users sharing them from the server, and claims the bike trip she selects. There is no need that users negotiate with each other and the server calculates the bike trip for each user. In the BTS algorithm, each user updates her strategy distributively until a predefined iteration limit τ_{max}. In this paper, its value is set large enough so that the BTS algorithm can converge. Our experiment results in the next section hint that it is enough to set $\tau_{max} > 9$. The reason that each user can implement the



(a) Users select their initial trips

(b) u_1 updates to the best response while u_2 fixes her strategy

(c) u_2 updates to the best response while u_1 fixes her strategy

Fig. 4. An example to illustrate the distributed BTS algorithm. Users select their initial trips in Figure 4(a), and implement the best response updates asynchornously in Figure 4(b) and 4(c), in which u_2 is after u_1. They reach a NE after several rounds of updates.

Algorithm 2 Information Exchanging Algorithm for the server

1: **if** Recieve user request message **then**
2: Provide the user with the information of bike stations: the locations of bike stations, the numbers of bikes and empty docks in them, and the number of users sharing the bike stations.
3: **end if**
4: **if** Recieve user bike trip message **then**
5: Update the information of the bike stations on the trip.
6: **end if**

BTS algorithm distributively is that the server offers each user the number of users sharing the bike stations in the step 3 of Algorithm 1. Therefore, they can know the number, and then select their trips to avoid congestion with others.

Figure 4 shows an example to illustrate the distributed BTS algorithm converging to a NE in the asynchronous way. The algorithm allows users to do the best response update in the deterministic way. It thus saves the communication among users. Actually, the way that the server processes the messages in order of their arrival is a method to control users to do the asynchronous strategy update.

B. Bike Trip Pruning

In Algorithm 1, each user needs to collect the information of the bike station in B. We can implement the following reasonable pruning so as to reduce greatly the amount of the information that each user needs to process. Suppose that there is the shoretest path directly from $l_s(u_j)$ to $l_t(u_j)$ without going through any bike station. Denote the bike time cost for the path by $t(l_s(u_j), l_t(u_j))$ and call it *endurance time*. So each user needs only to look for the bike stations to rent bike within the range of the endurance time centered at $l_s(u_j)$. Denote those bike stations by a set B_j^r. Similarly, she can obtain another set B_j^t of bike stations when she returns bike. Let $B_j = B_j^r \cup B_j^t$ and obviously $B_j \subseteq B$. For example, the size of B is over 3000 while the size of B_j is 10 on average in the experiment of Section V. By the bike trip pruning, the great time on calculation can be reduced so that users can converge to NE quickly.

V. Experiment Evaluation

This section conducts two experiments to evaluate the performance of our algorithm. The first one adopts the real data collected from the PBS of Hangzhou City in China from April 1 to May 31, 2014 with 9.1 million records. The PBS of Hangzhou is the largest one around the world and has more than 3300 stations and over 84,000 bikes [3]. The primary fields in the PBS datasets is shown in Table II. In the second experiment, we design the real crowdsourcing system including one server and the Android APP. Some volunteers take part in the experiment and implement it in practice.

TABLE II
Primary fields in the PBS datasets

user_id	rent_netid	tran_date	tran_time
6132518	3088	20140420	000154
return_netid	return_date	return_time	bike_id
4264	20140420	000162	802347

A. Real Data based Evaluation

Firstly, we propose a benchmark algorithm, called Relaxed BTS algorithm (RTS), in which each bike station is supposed to have infinite bikes and docks so every user can find the shortest path. Secondly, the whole urban area of Hangzhou City is divided into 5 experimental subareas as shown in Figure 5. We randomly generate the users' source and terminal locations in the 5 regions for the algorithms BTS, RTS and RD. RD represents the real data collected from the PBS. Note that RD has only the information from the initial bike station to target one and no information of the walking segments. So the generated users' source and terminal locations are also applied to RD. In the real data, the bike trips arrive online. To obtain offline bike trip information, we extract the bike trips with moment by moment in the five subareas, such as 6:30 Am on April 20 in the subarea A, and read the status of the bike stations at the moment from the PBS datasets of Hanghzou. Each extracted trip corresponds to one user. The number of trips extracted at one moment in one area is packed into one set. It represents one user set U. U in each trial may be different from that in another. The time cost between each pair of bike stations is represented by the average value of those real biking times between them in the PBS datasets. Considering the bike trip pruning in Section IV-B, each user sets the endurance time a random value as given in Table III to represent the time cost of walking segment. In the following

TABLE III
EXPERIMENT SETTING

Factor	Setting
Endurance time	random(150,250) second
Region	A, B, C, D, E in Figure 5
Data sampling time	From 6:30 to 19:30 of every day with one hour duration in April and May. For example, 6:30, 7:30, 8:30, $\cdots$, 19:30.

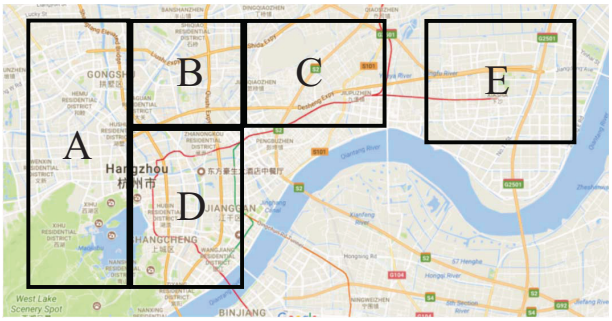
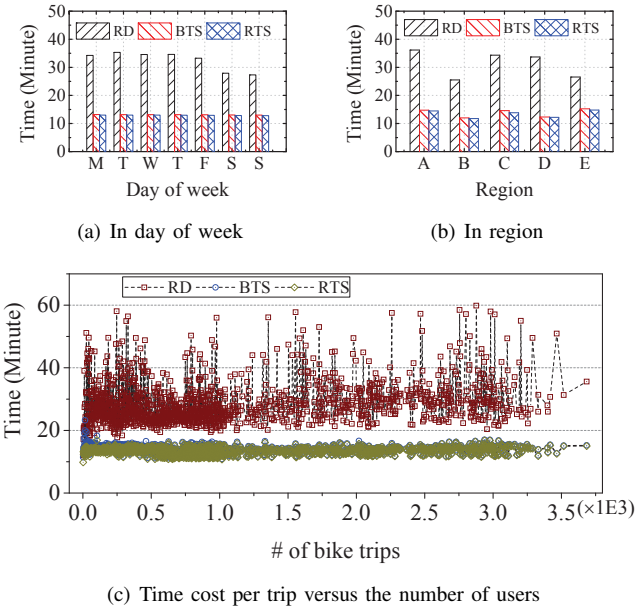


Fig. 5. Experiment areas includes five parts: A, B, C, D and E.



(a) In day of week (b) In region

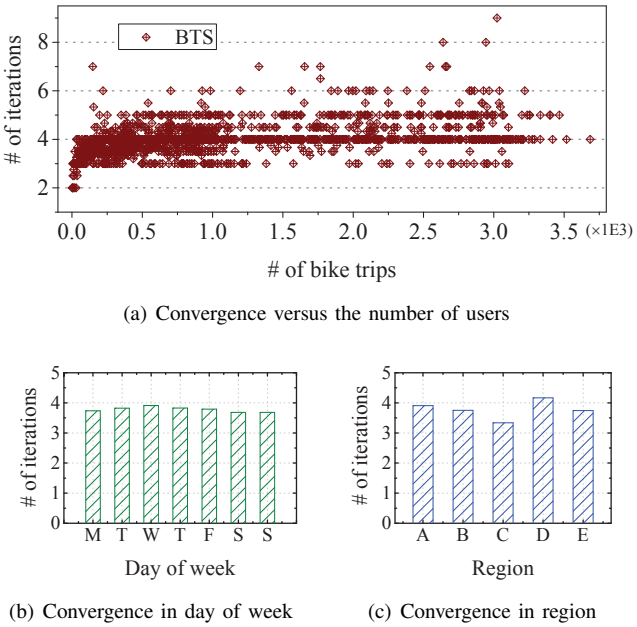
(c) Time cost per trip versus the number of users

Fig. 6. The time cost per trip of RD, BTS and RTS in three cases.

context, *the average time represents the time cost per bike trip*.

Figure 6 shows the average time of RD, BTS and RTS. We can see that the average times of the three algorithms vary a little as shown in Figure 6(a). The average time of RD 32.46 minutes while those of BTS and RTS are 13.12 and 12.93 minutes. BTS is 0.19 minutes higher than RTS on average so they are quite close to each other. The average time of BTS is 59.58% lower than that of RD, and 1.47% higher than RTS. The average time varies a little big among regions in Figure 6(b). RD is up to 38.18 minutes in the region A. The average time of BTS is from 12.01 to 15.22 minutes while that of RTS is from 11.77 to 14.80 minutes. We select 1422 sets of bike trips and each set contains a certain number of bike trips different from others. The average times of the three algorithms have different performance when the number of trips in each set increases as shown in Figure 6(c). It keeps stable under BTS and RTS and unstable under RD when the number of bike trips takes different values.

The convergence of BTS algorithm is also evaluated as shown in Figure 7. The numbers of convergent iterations increase with the number of bike trips in each set before the size of the set is less than 77. After the number, the convergent iteration keeps mainly between 3 and 6 in most cases no matter in the day of week and region. The maximal number of iterations is 9 in the figure. So the BTS algorithm converges fast and can keep stable performance. Since the users' source and terminal locations are randomly generated and the bike trip pruning is adopted, some users may not find bike trips. Figure 8 plots the successful ratio to select bike trip by the distributed BTS algorithm. It is at least 99.79% in day of week and 98.64% in region. Its average value is 99.48%. However, it is hard to count the successful ratio for RD since there is



(a) Convergence versus the number of users

(b) Convergence in day of week (c) Convergence in region

Fig. 7. The convergence of the distributed BTS algorithm.

no any related information left in the PBS datasets. The RTS has no resources limitation, its successful ratio is always full.

B. Real System based Evaluation

To evaluate the performance of the distributed BTS algorithm in practice, we implement our real experiment in Hanghzou City with our BTS system. The APP UI of the system is shown in Figure 9(a) and 9(b), through which user inputs her source and terminal locations. The APP displays her source

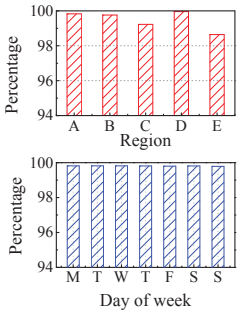
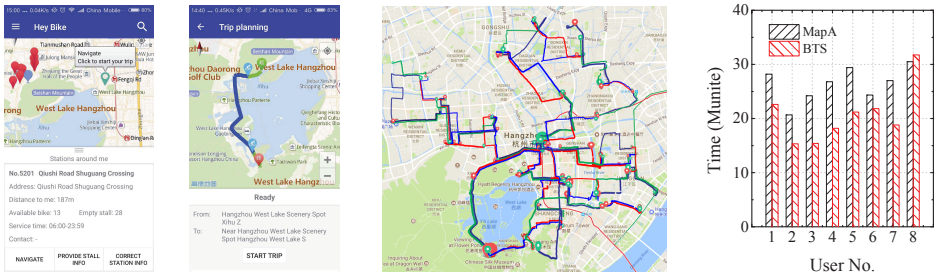


Fig. 8. Successful ratio to select trip of the BTS algorithm.



(a) Locating (b) Trip selection (c) Overlaid trips in color (d) Average time cost per trip

Fig. 9. APP UI, bike trips overlay graph and experimental results.

and terminal locations, the bike stations (in red bubble) near her source location in Figure 9(a), and a bike trip that the APP selects for her in Figure 9(b). Eight volunteers take part in the real experiment, which costs more than 40 hours and 200 kilometers in all. 41 pairs of source and terminal locations are randomly chosen in the urban area of Hangzhou City. For fairness, 7 volunteers choose randomly 5 pairs and the last one chooses 6 pairs. They travel each pair of source and terminal locations with twice trials: the distributed BTS algorithm and the map assisted traveling. The later trial is called "MapA". The volunteers can find the bike stations and obtain the navigation route with the help of electronic map, such as Baidu or Google map, but cannot obtain the complete navigation route from the source location to the terminal location. So the comparison between our algorithm and MapA is quite challenging.

The tracks of all experiment trips are recorded and are overlaid together in color, which is shown in Figure 9(c). Figure 9(d) shows the average time cost per trip of the 8 volunteers under both the distributed BTS algorithm and MapA. The average time cost per trip of the first 7 volunteers under the BTS algorithm is better than that under MapA. They save time 19.86%, 25.81%, 36.36%, 32.09%, 27.89%, 10.27% and 30.37% respectively. There could be room for the improvement of the distributed BTS algorithm because of some factors. For example, the volunteers are not familiar with the APP operation and the BTS APP is newly designed and has not been tested in wide versions of smartphones. The APP is not compatible well with the 8^{th} volunteer's smartphone. He restarts the APP several times during each trip trial and thus wastes much time as shown in Figure 9(d). But the average saved time over all volunteers is up to 22.32%.

VI. Related Works

A. PBS and Crowdsourcing

The public bike is a kind of green transportation tool and convenient for public travel in city. It increasingly appears in many cities, such as New York (Citi Bikes), Chicago (Divvy), San Francisco (SRFBikeShare), Washington D.C. (Capital Timeshare) [20], and Hangzhou in China [3][21][22].

Crowdsourcing has been applied to the PBS on some systems. *Ride with GPS*, *Map my Ride* and *Bikely* allow cyclists to map biking routes and to share them with others so as to facilitate cyclists to borrow bikes in cities [20][23]. Misra *et al.* propose the use of crowdsourcing mechanism to involve a large group of stakeholders in transportation planning and operations [24]. Motta *et al.* present IRMA, a cross-device and cross-platform system that enables users to manage a multimodal mobility [25]. Wu *et al.* analyze the accuracy of Google biking times using crowdsourced data from thousands of urban bike routes generated by the bike sharing system [20]. Torres *et al.* design a participatory sensing system, BeCity, which takes advantage of the collective knowledge of transportation cyclists to improve the quality of city cycling and provides refined city usage information for cycling associations [26].

Different from these crowdsourcing platform, this paper designs such a platform to help users select the bike trip, which can be close to the practical demand.

B. Research on PBS

To solve the problem, no bike to rent or no dock to return bike, an increasing number of works have been devoted to PBS, such as the redistribution, system prediction and bike trip planning [7][21]. Some works design routing mechanisms for truck to move bikes and incentive mechanisms for users to help with bike redistribution [6][27][28][29].

Fricker *et al.* compute the least rate for the bike redistribution by truck given quality of service, and suggest users to return to the least loaded stations [27]. Singla *et al.* design a crowdsourcing mechanism to provide the users with alternative stations by monetary incentive [5]. Liu *et al.* solve the large scale vehicle redistribution problem [29]. System prediction estimates bicycle availability based on the recent history of bikes in different bike stations, such as hourly prediction of rental bicycles [3], bike demand prediction [30][31] and the individual trip prediction [32].

Few works provide real time trip planning for users. Yoon *et al.* predict the availability of bike resources at every bike station and help users select the best pair of stations with the minimal time cost and the maximal probability to finish trip after giving the origin and destination [7]. Zhang *et al.* aim at finding the optimal trip route from the origin to the destination through several bike stations in the PBS by the

minimum-cost network flow algorithm [33]. These works did not fully consider the complete bike trip and the service quality of the system.

However, the redistribution and prediction of PBS are not very precise and timely for single user so it cannot tell users exact information of bike trips. Existing works on the trip planning did not consider the complete bike trip or are probability-based. So the planned trip is not timely too. This paper suggests the deterministic way to help user select bike trip with crowdsourced information and thus provides bike trip selection beforehand.

VII. Conclusion

This paper studies the BTS problem for the practical PBS by considering the complete three-segment bike trip. The problem is mapped to the BTS game, which is proved to be equivalent to the symmetric network congestion game. So the existence of NE and the FIP is ensured. The distributed BTS algorithm then is designed to find NE with finite iterations. We evaluate the algorithm with the PBS datasets of Hangzhou City in China. The experiment results show that the algorithm can save much time compared to the way in users' daily habit, *i.e.*, MapA. We also designed the BTS system including the Android APP and the server. Some volunteers take part in the experiment with the intensive trials in Hangzhou City. The experiment on the system indicates that BTS algorithm can much time.

Acknowledgement. This work is supported by the National Natural Science Foundation of China under Grants No. 61473109, 61572164 and 61671193, Key Research and Development Plan of Zhejiang Province under Grant No. 2018C04012.

References

[1] Peter Midgley. The role of smart bike-sharing systems in urban mobility. *Journeys*, 2(1):23–31, 2009.

[2] Ahmadreza Faghih-Imani and Naveen Eluru. Analysing bicycle-sharing system user destination choice preferences: Chicago's divvy system. *Journal of transport geography*, 44:53–64, 2015.

[3] Zidong Yang, Ji Hu, Yuanchao Shu, Peng Cheng, Jiming Chen, and Thomas Moscibroda. Mobility modeling and prediction in bike-sharing systems. In *ACM MobiSys*, pages 165–178, Singapore, Jun 26-30 2016.

[4] Jie Bao, Tianfu He, Sijie Ruan, Yanhua Li, and Yu Zheng. Planning bike lanes based on sharing-bikes' trajectories. In *ACM KDD*, accepted to appear. Halifax, Nova Scotia, Canada, Aug 13-17 2017.

[5] Adish Singla, Marco Santoni, Gábor Bartók, Pratik Mukerji, Moritz Meenen, and Andreas Krause. Incentivizing users for balancing bike sharing systems. In *AAAI*, pages 723–729, Austin, TX, USA, Jan 25-29 2015.

[6] Julius Pfrommer, Joseph Warrington, Georg Schildbach, and Manfred Morari. Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1567–1578, 2014.

[7] Won Yoon Ji, F. Pinelli, and F. Calabrese. Cityride: A predictive bike sharing journey advisor. In *IEEE MDM*, pages 306–311, Bengaluru, India, Jul 23-26 2012.

[8] Jianhui Zhang, Zhi Li, Xiaojun Lin, and Feilong Jiang. Composite task selection with heterogeneous crowdsourcing. In *Proceedings of IEEE SECON*, pages 379–387, San Diego, CA, USA, Jun 12-14 2017.

[9] Xinglin Zhang, Zheng Yang, Wei Sun, Yunhao Liu, Shaohua Tang, Kai Xing, and Xufei Mao. Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):54–67, 2016.

[10] Shibo He, Dong Hoon Shin, Junshan Zhang, Jiming Chen, and Phone Lin. An exchange market approach to mobile crowdsensing: Pricing, task allocation and walrasian equilibrium. *IEEE J-SAC*, PP(99):1–1, 2017.

[11] Berthold Vöcking. Congestion games: Optimization in competition. In *Proceedings of the Second ACiD Workshop on Algorithms and Complexity*, pages 9–20, Durham, UK, Sep 18-20 2006.

[12] Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

[13] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In *ACM STOC*, pages 604–612, Chicago, IL, USA, Jun 13-15 2004.

[14] Robert W Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[15] C Tekin, Mingyan Liu, R Southwell, and Jianwei Huang. Atomic congestion games on graphs and their applications in networking. *ACM/IEEE TON*, 20(5):1541–1552, 2012.

[16] Richard Southwell, Jianwei Huang, and Xin Liu. Spectrum mobility games. In *Proceedings of IEEE INFOCOM*, pages 37–45, Orlando, FL, USA, Mar 25-30 2012.

[17] Marios Mavronicolas, Igal Milchtaich, Burkhard Monien, and Karsten Tiemann. Congestion games with player-specific constants. In *MFCS*, pages 633–644, Ceský Krumlov, Czech Republic, Aug 26-31 2007.

[18] Richard Southwell, Yanjiao Chen, Jianwei Huang, and Qian Zhang. Convergence dynamics of graphical congestion games. In *International Conference on Game Theory for Networks*, pages 31–46, 2012.

[19] Joaquim Gabarró, Alina García, and Maria Serna. On the complexity of game isomorphism. *Mathematical Foundations of Computer Science*, pages 559–571, 2007.

[20] Mingsheng Wu and Vanessa Frias-Martinez. Crowdsourcing biking times. In *Adjunct Proceedings of UbiComp/ISWC*, pages 1123–1131, Osaka, Japan, Sep 7-11 2015.

[21] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. Traffic prediction in a bike-sharing system. In *ACM SIGSPATIAL*, pages 33:1–33:10, Seattle, WA, USA, Nov 3-6 2015.

[22] Yang Tang, Haixiao Pan, and Qing Shen. Bike-sharing systems in beijing, shanghai, and hangzhou and their impact on travel behavior. In *Transportation Research Board 90th Annual Meeting*, number 11-3862, Washington, D.C, USA, Jan 23-27 2011.

[23] Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, and Xing Xie. Searching trajectories by locations: an efficiency study. In *ACM SIGMOD*, pages 255–266, Indianapolis, IN, USA, Jun 6-11 2010.

[24] Aditi Misra, Aaron Gooze, Kari Watkins, Mariam Asad, and Christopher Le Dantec. Crowdsourcing and its application to transportation data collection and management. *Transportation Research Record: Journal of the Transportation Research Board*, (2414):1–8, 2014.

[25] Gianmario Motta, Daniele Sacco, Tianyi Ma, Linlin You, and Kaixu Liu. Personal mobility service system in urban areas: The irma project. In *IEEE SOSE*, pages 88–97, San Francisco Bay, USA, Mar 30 - Apr 3 2015.

[26] Salomon Torres, Felipe Lalanne, Gabriel Del Canto, Fernando Morales, Javier Bustos-Jimenez, and Patricio Reyes. Becity: sensing and sensibility on urban cycling for smarter cities. In *IEEE SCCC*, pages 1–4, Santiago, Chile, Nov 9-13 2015.

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

[28] Mauro Dell'Amico, Eleni Hadjicostantinou, Manuel Iori, and Stefano Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, 2014.

[29] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. Rebalancing bike sharing systems: A multi-source data smart optimization. In *ACM KDD*, pages 1005–1014, San Francisco, CA, USA, August 13-17 2016.

[30] Patrick Vogel, Torsten Greiser, and Dirk Christian Mattfeld. Understanding bike-sharing systems using data mining: Exploring activity patterns. *Procedia-Social and Behavioral Sciences*, 20:514–523, 2011.

[31] Patrick Vogel and Dirk Mattfeld. Strategic and operational planning of bike-sharing systems by data mining–a case study. *Computational Logistics*, pages 127–141, 2011.

[32] Jiawei Zhang, Xiao Pan, Moyin Li, and S Yu Philip. Bicycle-sharing system analysis and trip prediction. In *IEEE MDM*, volume 1, pages 174–179, Porto, Portugal, Jun 13-16 2016.

[33] Jiawei Zhang and S Yu Philip. Trip route planning for bicycle-sharing systems. In *IEEE CIC*, pages 381–390, Pittsburgh, PA, USA, Nov 1-3 2016.