TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214
DOI: 10.26599/TST.2020.9010002
Volume xx, Number x, xxxxxxx 20xx

Challenges and Opportunities in Algorithmic Solutions for Re-Balancing in Bike Sharing Systems

Jie Wu *

Abstract: In recent years, the booming of the bike sharing system (BSS) has played an important role in offering a convenient means of public transport. The BSS is also viewed as a solution to the first/last mile connection issue in urban cities. The BSS can be classified into dock and dock-less. However, due to imbalance in bike usage over spatial and temporal domains, stations in the BSS may exhibit overflow (full stations) or underflow (empty stations). In this paper, we will take a holistic view of the BSS design by examining the following four components: (1) system design, (2) system prediction, (3) system balancing, and (4) trip advisor. We will focus on system balancing, addressing the issue of overflow/underflow. We will look at two main methods of bike re-balancing: with trucks and with workers. Discussion on the other three components that are related to system balancing will also be given. Specifically, we will study various algorithmic solutions with the availability of data in spacial and temporal domains. Finally, we will discuss several key challenges and opportunities of the BSS design and applications as well as the future of dock and dock-less BSS in a bigger setting of the transportation system.

Key words: Algorithmic solutions, bike re-balancing, bike sharing system (BSS), data analytics

1 Introduction

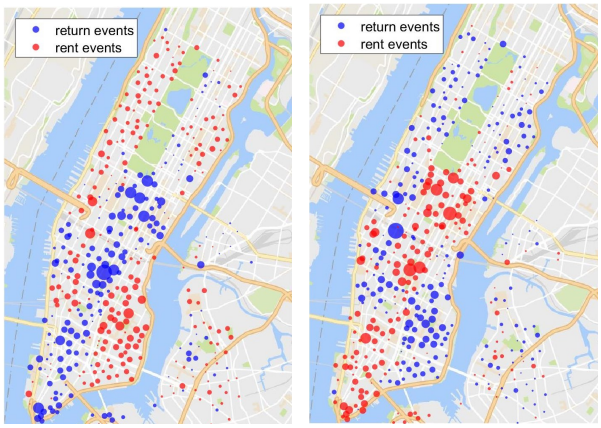
In recent years, the booming of the *bike sharing system* (BSS) has played an important role in offering a convenient means of publication transport. The BSS is also viewed as a solution to the *first/last mile connection issue*, getting people between public transport hubs (such as subway stations and bus stops) and home, in large urban cities, such as New York City (NYC) and Shanghai. As an integral part of smart city, BBS offers healthy lifestyle for citizens and green transportation in urhan cities. A recent survey has showed that 40% of BSS users drive less [1], which is very desirable in a crowed urban city.

The BSS can be classified into *dock* and *dock-less*. Sample dock BSSs include Citi Bike (NYC), Capital Bikeshare (DC), Indego (Philadelphia), GoBike (Bay Area), public bicycles (Shanghai, Beijing, and Hangzhou), BikeMi (Milan), BuBi (Budapest), and EBI (Esztergom). Dock-less BSSs include LimeBike, Spin, JUMP (bikes and electric scooters); Bird (electric scooters) in the U.S.; Mobike, ofo, and Hellobike in China; and U-Bicycle and OV-fiets in Europe [2]. As of May 2018, more than 1,600 bike-sharing programs were in operation worldwide, providing more than 18 million bicycles for public use for transport [3] with China, Italy, Unuted States, Germany, and Spain being the top-5 counties that use BSSs.

In a typical BSS, there are several stations scattered in a given region. Each station has a capacity limit in terms of the number of slots used to hold bikes. Users make use of the BSS through a pair of activities: renting a bike from one station and returning the bike to the same or another station. Because of variations in bike demand, imbalance may occur in bike usage at stations over spatial and temporal domains. Fig. 1 shows bike rent and return distributions in NYC across two domains: spatial (different locations in Manhattan)

• Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, 19122, USA. Email: jiewu@temple.edu.

* To whom correspondence should be addressed.
Manuscript received: 29-Dec-2019; Accepted: 03-Jan-2020



(a) AM rush hours: 8:00 - 10:00 AM (b) PM rush hours: 5:00 - 7:00 PM

Fig. 1 The users' demands in morning and evening rush hours in Manhattan.

and temporal (morning and evening). The usage discrepancies at different stations in the BSS may result in *overflow* (full stations that exceed the station capacity to hold a return bike) or *underflow* (empty stations with no bike to rent). Therefore, bike re-balancing - moving bikes from an overflow station to a underflow station - is needed (Fig. 2). Note that Fig. 2 shows bike re-balancing in a dock BSS. Overflow/underflow also occurs in dock-less BSSs. The author witnessed a phenomenon of overflow/underflow of a dock-less BSS in Shanghai in the summer of 2018. This overflow/underflow frequently occurs at subway entrances with heavy traffic flows at different time slots of a day (as shown in Fig. 3).

In this paper, we will take a holistic view of addressing bike re-balancing by looking at four components: (1) system design, (2) system prediction, (3) system balancing, and (4) trip advisor. The focus is on system balancing, addressing the issue of overflow/underflow. Discussion on the other three components that are related to system balancing will also be given. In system balancing, we will consider two groups of solutions for bike re-balancing: the first uses tracks for bike re-balancing and the other recruits workers with incentive. We will discuss various algorithmic solutions related to bike re-balancing. The focus will be on some open challenges.

The remainder of the paper is organized as follows: Section 2 discusses some key design issues related to bike-balancing at each design stage of four components. Section 3 focuses on bike re-balancing using tracks and Section 4 studies bike re-balancing by recruiting workers with incentive. Section 5 discusses extensions from

Fig. 2 Bike re-balancing between an overflow and underflow station.

solutions for one dimensional domain to two dimensional domains. Section 6 presents some future directions on challenges and opportunities in BSSs, with a focus on bike re-balancing. Section 7 discuss the role of bike sharing in a bigger setting of the transportation system. Section 8 concludes the paper.

2 Four System Components

We first discuss four components of a BSS, focusing on issues related to bike re-balancing.

2.1 System design

System design [4–6] includes the selections of the number of stations, station location, station capacity, and the number of bikes in circulation. Clearly, with the increase in the number of stations and their capacity, the need for bike re-balancing will be reduced. However, such an increase will incur cost for BSS operators. Therefore, sensible balance is needed on cost and effect in the system design phase.

At first glance, system design resembles the classic *facility location problem* [7]: it consists of a set of potential facility sites (i.e., stations in a BSS) and a set of demand points (i.e., users in a BSS) that must be serviced. The goal is to pick a subset of facilities to open (i.e., the number and locations of stations in a BSS) to minimize the sum of the distance from each bike demand location to its nearest facility and plus the total cost of the stations. However, the dynamic of stations where bikes are returned makes the BSS system design more challenging. This is because the capacity of each station varies even if we can accurately predict rent locations, as in the classic facility location problem. Bike re-balancing can be viewed as an external means to make the service level of a BSS more predictable and accountable.

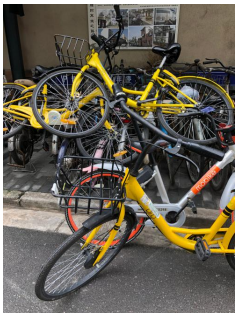


Fig. 3 Overflow in a dock-less BSS in Shanghai (June 2018).

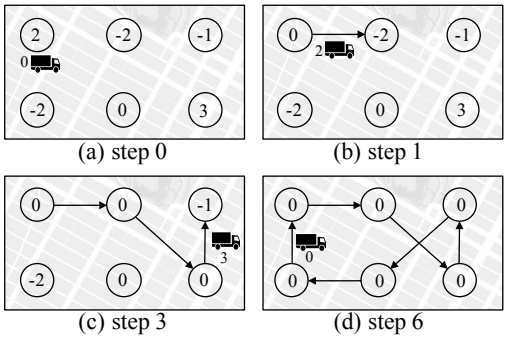


Fig. 4 A sample bike re-balancing using a truck sweeping along a given Hamiltonian circle.

2.2 System prediction

System prediction deals with data collection and prediction of bike demands over spatial and temporal domains. The prediction process is rather complex. It involves user mobility modeling [8] and traffic prediction [9] across both spatial and temporal domains. To reduce complexity, some researchers used the cluster-based approach [10], which groups similar stations into clusters to reduce the computation complexity.

Bike usage prediction deals with not only more predictable common contextual factors, such as time and space, but also opportunistic contextual factors, such as special social and traffic events. Some events propagate along the physical vicinity (e.g., a special social event at a physical location) as well as the logical one (through a social network). In this case, the traffic prediction at different stations may be correlated. All of these will bring uncertainty in traffic prediction, making bike re-balancing more challenging.

2.3 System balancing

System balancing deals with bike re-balancing over both spatial and temporal domains. One common approach is to partition the two-dimensional domain into a sequence of *slices* of the one-dimensional domain. For example, if the two-dimensional domain is sliced based on the time, then each slice deals with only the spatial domain, i.e., balancing bikes across the physical space with a fixed time slot.

System balancing usually has two approaches: *dedicated truck service* [11–17] or *incentive-based worker recruitment* [18–21]. In dedicated truck service, one or more trucks are used to move around stations to pick-up and drop-off bikes at different stations according to their overflow and underflow situations. In incentive-based worker recruitment, an incentive mechanism is used to recruit individual workers to re-balance bikes on a per-bike basis. One particular interesting model is to recruit workers from BSS users who are willing to go through a small detour in their regular bike journey.

2.4 Trip advisor

Trip advisor [22], operated by the BSS operator, can serve two purposes. On one hand, it provides BSS users some guidance on the availability of nearby stations for rent/return. It can also give advice on route selection to avoid traffic or to avoid the distributed trip selection game among users.

Trip advisor can also suggest bike selection to BSS users to balance individual bike usage at different stations [23]. For example, when two bike return stations are equivalent to a bike user, the trip advisor can recommend the user to return the bike to the station with fewer bikes to increase the overall system utility and enhance the long-term service level [23]. On the other hand, trip advisor can also recommend choices that may lead to bike re-balancing, provided there is little or no deviation from the optimal choice made for the user.

3 Bike Re-balancing Through Trucks

In dedicated truck service, one or more trucks are used to move around stations to pick-up and drop-off bikes at different stations according to their overflow and underflow situations. To simplify our discussion, we assume that one truck is used with a capacity of l at a particular time slice. A station with $+m$ ($-m$) stands for overflow (underflow) by m slots as shown in Fig. 4. $-l \leq m \leq l$ is always true; otherwise, we can split a station into multiple adjacent stations in such a way that $-l \leq m \leq l$ holds for each new station. Overall, overflow and underflow stations are balanced, i.e., the total amount of $+$ values equals that of $-$ values.

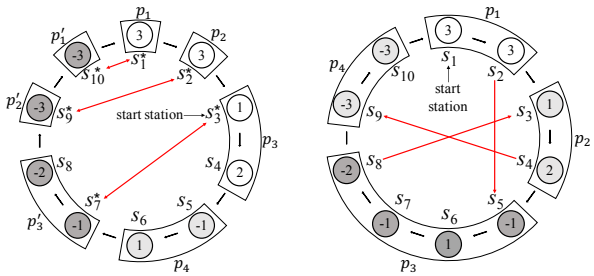
3.1 Constructing a legitimate Hamiltonian circle

A typical solution involves first finding a certain Hamiltonian circle among overflow/underflow stations as a starting point. One sweep is used to visit each station to reset each station value to 0 through bike pick-up for a positive station or bike drop-off for a negative station as shown in Fig. 4. This problem is more challenging than the classic Hamiltonian circle problem, as during the routing process (i.e., finding a Hamiltonian circle), the truck itself cannot be negative (the number of bikes carried by the truck falls below zero) or exceed the truck capacity (i.e., more than l). For example, if $l = 4$ in Fig. 4, then if the truck goes to the station with 3 bikes after completing its first move with 2 bikes, then 2 + 3 exceeds the truck capacity, generating an illegitimate route.

Among existing approaches, two methods partition a given Hamiltonian circle into *positive-pieces*, *negative-pieces*, and *zero-pieces*. The positive-pieces and negative-pieces are used alternatively in the visit sequence to keep the load within the truck capacity. To identify these pieces, a construction process starts from a given node, called start node, which can be either positive or negative node (zero nodes are not involved), and then connects these pieces following the clockwise direction of the given Hamiltonian circle to form a legitimate Hamiltonian circle that satisfies the truck's capability constraint. The following is the construction process that identifies each piece and its label: if the start node is a positive $+$ (negative $-$), follow the circle to find the next station to connect until the load summation reaches a predefined threshold $l^{'}$ $(-l^{'})$ or reaches a negative (positive) value. In the former case, the process is complete and the corresponding piece is called a positive (negative) piece, and in the latter case, location is either shifted back one station or the current station is partitioned into two virtual stations in such a way that the load summation becomes zero by connecting one virtual station; such a piece is called *positive zero* (*negative zero*). Positive and negative zero pieces can be simply called zero pieces if their signs do not play any role in a solution.

3.2 MATCH method

In the method proposed in [24], called MATCH here, $l^{'}$ is set to $l/2$. Fig. 5 (a) shows such a partition with s_3 as the start node, assuming that $l^{'} = 3$. The given Hamiltonian circle is then partitioned into positive-, negative-, and zero-pieces in one sweep. MATCH performs a



(a) MATCH (b) GREED

Fig. 5 Constructing a legitimate Hamiltonian circle by partitioning a given circle into positive (white), negative (dark), and zero (gray) pieces.

minimum-weight perfect matching between positive and negative pieces (zero pieces are not included). Here, positive and negative pieces form matching pairs, p_i and $p_i^{'}$ for $i = 1, 2, 3$, in which the distance of each pair defined is based on the geographic distance between two closest nodes (called bridge nodes denoted by $*$), one each from the pair. The path is constructed beginning from the start node and following the given circle. The only constraint is that when a piece is visited, its matching piece is co-visited at the same time. This is done through traversing the matching pair using two gateway nodes on the pair. However, the selected start node may not be legitimate, as load summation at the truck may become negative. It is proved that $l^{'}$ is defined in such a way that we can always find a legitimate start node with the same circle constructed from the random start node.

In Fig. 5 (a) with the initial start node s_3, the visitation sequence generated by MATCH is $(s_3, s_7, s_8, s_4, s_5, s_6, s_9, s_2, s_{10}, s_1, s_3)^*$. The number of bikes on the truck (i.e., load summation) after visiting each station is $(1, 0, -2, 0, -1, 0, -3, 0, -3, 0)$. This sequence is not legitimate as it contains negative values, with -3 being the smallest. A legitimate start node can be found by shifting the initial start node on the newly constructed circle from s_3 to s_1 based on the smallest value -3 (i.e., shifting back one station on the new circle). The resulting sequence becomes $(s_1, s_3, s_7, s_8, s_4, s_5, s_6, s_9, s_2, s_{10}, s_1)$. The load summation sequence becomes $(3, 4, 3, 1, 3, 2, 3, 0, 3, 0)$, which is clearly legitimate with all non-negative values and all values being less than l, the truck capacity. The complexity of MATCH is $O(n^3)$, where n is the number of

*The traversal between matching pairs p_3 and $p_3^{'}$ through two gateways s_3 and s_7 generates the following visitation order: s_3, s_7, s_8, s_7, s_3, and s_4. By the keeping only first visit record, we have the following sequence: (s_4, s_7, s_8, s_5).

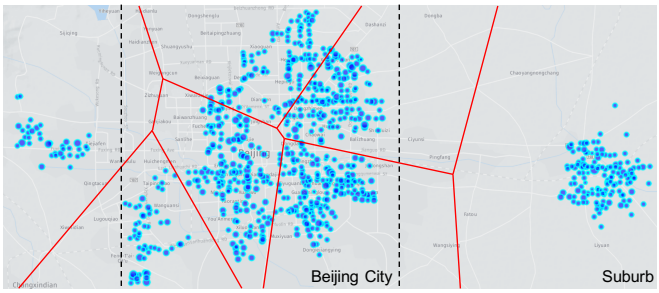
stations. MATCH guarantees an approximation ratio of 6.5 [24], compared to the optimal solution to a problem that is NP-hard.

3.3 GREED method

In another method in [25], called GREED here, a different approach is used to connect pieces, where l' is set to l $(= 6)$ in Fig. 5 (b). The legitimate Hamiltonian circle starts with any positive station as a start node which will generate either a positive piece or a positive zero piece. If a positive piece is generated, the process finds the closest negative station along the circle to connect. If that negative station ends up with a negative piece, the process will find the closest positive station and the same process continues. If the negative station ends up with a negative zero piece, the closest negative station is sought again. This process continues by alternating positive and negative pieces. Note that positive zero pieces will not change the load summation of the truck. In the above process, once a piece is visited, it is removed from the circle so that the remaining non-visited stations still form a Hamiltonian circle with an updated adjacency relationship. For example, once p_1 finds the matching negative piece p_3 in Fig. 5 (b), both p_1 and p_3 are removed and the remaining pieces p_2 and p_4 are then connected forming a positive zero piece with start node s_3. The visiting sequence generated by GREED is $(s_1, s_2, s_5, s_6, s_7, s_8, s_3, s_4, s_9, s_{10}, s_1)$. The load summation sequence is $(3, 6, 5, 6, 5, 3, 4, 6, 3, 0)$. The complexity of GREED is lower which is $O(n^2)$. However, the algorithm does not guarantee any approximation ratio. To find a good start node, multiple start nodes can be used and then select the best start node is selected based on the GREED result.

3.4 HYBRID method

Comparing MATCH with GREED, it is shown in [25] that MATCH outperforms GREED in a relatively sparse mode (i.e., a small number of stations) as its primary factor or a small geographical area as its secondary factor while GREED is better in a dense mode or a large geographical area. One possible extension is to combine MATCH and GREED to form a two-level hybrid solution. In general, a BSS may not have a uniform distribution of stations. Instead, stations in a dense population area are clustered. In the hybrid solution, called HYBRID, MATCH is used for intra-cluster bike re-balancing and GREED is used for inter-cluster bike-balancing.



(a) A sample distribution of dock stations in Beijing [26]

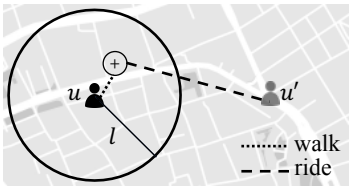
	MATCH	GREED	HYBRID
City	2.064	1.108	0.881
City+Suburb	3.016	1.923	1.080
City (Sparse)	1.435	1.781	1.342
City + Suburb (Sparse)	2.597	2.575	1.827

(b) MATCH, GREED, vs HYBRID

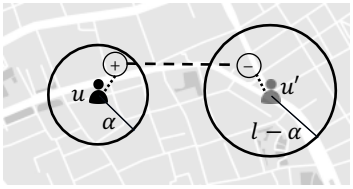
Fig. 6 Performance comparison.

Fig. 6 shows a performance comparison among MATCH, GREED, and HYBRID based on the a real set of data (i.e., distribution of bike stations in Beijing). In Fig. 6 (a), it is shown that stations in the center city are unevenly distributed among roughly five clustered centers. When including the suburbs, there are seven clusters. The map is first divided by a mesh. The size of each square is 0.4 (longitude) $\times$ 0.3 (latitude) (in km). The threshold is set as 1, i.e., a grid has a station if there is at least one station. As a result, 550 stations are found. A sparse sampled map is generated with a mesh size of 0.8×0.6 and a threshold of 4. As a result, 76 stations are included in the sparse sampled data set. The capacity of the truck is set as 20. The re-balancing target of each station is randomly set by following Poisson distribution with a mean of 7. The sign of the re-balancing target ($+$ or $-$) is randomly decided with equal possibility. The last station's target is set to guarantee that the sum of targets among all stations is 0.

Fig. 6 (b) shows the results in terms of total travel distance of bike re-balancing, one for center-city (Beijing) and the other for the whole city, by including both center city and its suburbs. The simulation results show that HYBRID has 57.3% (6.48% for sparse sampled map) and 20.5% (24.6%) less distance in the center city and 64.2% (29.6%) and 43.8% (29.0%) less distance in the entire city compared to MATCH and GREED, respectively. The number in each entry represents the average travel distance in km for per re-balancing bike, where the total number of re-balancing bikes is a summation of absolute values in all re-balancing targets divided by 2. Obviously, travel distance is relatively large for



(a) Source incentive



(b) Source and destination incentive

Fig. 7 Incentive worker recruitment with monetary rewards based on walking distance.

cases with suburbs, and more trucks should be used to cover different geographic regions, as will be discussed in Section 6.

4 Bike Re-balancing Through Workers

Bike re-balancing through workers with incentive focuses on recruiting individual workers, which can be BSS users. As each worker can only move one bike at a time, we use $+/-$ sign to denote an overflow/underflow spot (for dock-less BSS) or a station (for dock BSS). Usually, the BBS operator gives monetary incentives for workers. Here, we focus on approaches where workers are BSS users who plan to use bikes on their journey anyway and are willing to take a detour with either monetary reward or one or multiple free-rides. We use u and $u^{'}$ to represent a user's current location and his/her intended destination.

4.1 Incentive method for individual workers

In [21], a monetary incentive approach is studied in a dock-less BSS (and also can be used in a dock BSS), where a bike user is asked to walk to a location that is within l distance from u. The monetary award is based on the walking distance from u, with the amount as either a linear or quadratic function of the distance. The following global optimization program is formulated: given a fixed amount of total budget and bike demands over spatial and temporal domains, how do we set the amount of the monetary award for each location to maximize the service level of the system? It is assumed that for each walking distance upper bounded by l (as shown in Fig. 7 (a)), users set their monetary award threshold for each distance in advance. Users will accept the request if the monetary award meets the threshold. The pricing distribution is calculated through reinforcement learning based on past data. This approach can be extended by providing incentives at both source and destination [27] as shown in Fig. 7 (b), as long as the total walking distance at source (α) and destination ($l - \alpha$) is bounded by l, where α is a tunable parameter. In fact, this extension is more powerful than the original approach. Instead of solving the overflow problem as in [21], the extension basically recruits a worker to solve a pair of overflow and underflow spots. The only difference is that in the extension reinforcement learning, the pricing at both source and destination needs to be considered.

4.2 Incentive method for a group of workers

In [28], a more general incentive approach is studied to address bike re-balancing among overflow and underflow stations in a dock BSS (and also in a dock-less BSS). Here, the approach deals with multiple workers through matching, rather than each individual worker assignment. Fig. 8 shows a simple example with one user and two overflow stations and two underflow stations. The user walks from u to rent a bike at one of two overflow stations, returns at one of two underflow stations, and finally walks to his/her intended destination $u^{'}$. Obviously detour occurs, compared to a straight line between u and $u^{'}$, i.e., the shortest distance between u and $u^{'}$. In this case, there are four possible choices for the user: (s_1, s_3), (s_1, s_4), (s_2, s_3), and (s_2, s_4). When there are multiple users, the problem becomes *3-dimensional perfect matching* among users, overflow stations, and underflow stations. For example, if there are two users, u_1 and u_2, the number of possible matching becomes $2^3 = 8$. In general, multiple-dimensional perfect matching is an NP-hard problem. In [28], a two-round of the perfect bipartite matching is used to approximate the 3-dimensional perfect matching problem with a 3-approximation using the geometric properties of locations. This approximation is done by first matching overflow and underflow stations and then by matching users to overflow-underflow pairs.

5 Complexity in the Spatial and Temporal Domains

So far, our discussion has focused on one slice that deals with the spatial domain only. Adding the temporal domain will significantly increase the complexity of the problem.

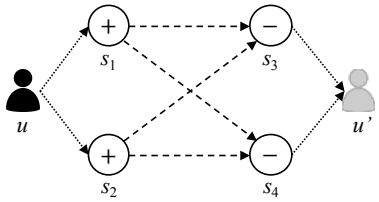


Fig. 8 Incentive worker recruitment through minimizing total detour using 3-dimensional perfect matching.

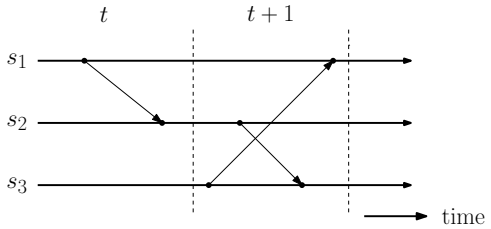


Fig. 9 A time-space view of bike re-balancing with slanted arrow lines representing bike re-balancing activities between pairs of stations.

5.1 Time-space view

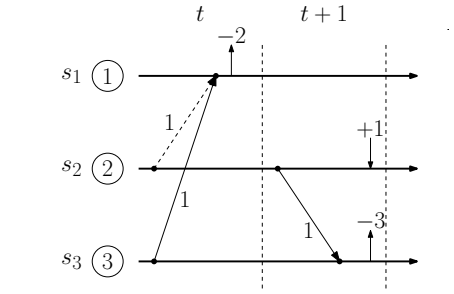
Let us first take a *time-space view* of the system from the classic distributed computer system [29]. Fig. 9 shows such a view with three stations. Each station has its value (i.e., the number of bikes) as its local state. A bike rented from a station and the corresponding bike returned to another station completes a bike journey, even if the bike can be returned to the same station. The classic distributed computer system modeling and analysis can still be applied here, where global state (total number of bikes) equals the summation of local stations (all bikes available for rent), plus bikes in transit (on road by either normal bike users, on trucks, or being ridden by workers for bike re-balancing). Our bike re-balancing scheme can also be represented using a similar view. As shown in Fig. 9, each slanted arrow line corresponds to a bike re-balancing event between two stations. Each slice corresponds to a time period represented by a vertical dotted line. Ideally, each re-balancing event does not go across two slices (called a "cut" in the distributed computing community). If we set the re-balancing activity granularity to a relatively long period, say one or two hours, each re-balancing effort can be done within that time frame for a relatively small region under the study period. Thus, the cut issue can be mitigated and hence ignored.

5.2 Reducing re-balancing frequency through look ahead

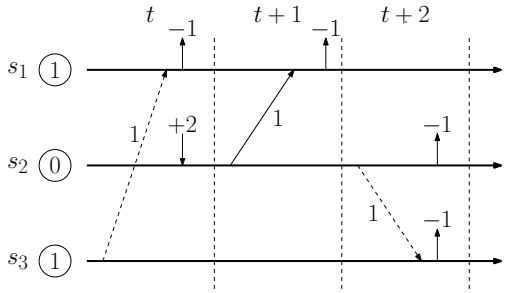
One main challenge in extending the single slice solution to cover multiple slices lies in the global determination of re-balancing frequency and the corresponding target for each re-balancing. The idea used here is to reduce the re-balancing frequency. For example, setting frequency to k means that re-balancing is applied at every k slices. That is, the system conducts re-balancing in the current slice, but with the k-slice view (called *k-hop look ahead*), such that once the target is set and done in the current slice, it can last at least k slices; otherwise, the frequency k needs to be reduced either uniformly for all slices or at each re-balancing activity. However, this one-policy-fits-all approach does not work well since demand varies across time and space. Another approach called *greedily look-ahead*, which is inspired by a method in [11], uses look-ahead data to make the best target move at the current slice such that the target configuration will last the longest in the number of slices, given that all future data is known. This greedily look-ahead approach may still be outperformed in some cases as shown later. Note that the complexity of both k-hop look ahead and greedily look ahead is $O(kn \log n)$, where n is the number of stations and k is the number of look ahead slices.

Fig. 10 (a) illustrates why in k-hop look ahead, it is better in general to have a larger k, even if all moves are done at the current slice. We assume that all new updates (i.e., load re-balancing) are completed in the current slice before bike rent $(-)$ and bike return $(+)$ occur at the end of the current slice. Rent and return activities are represented by a short outward vertical arrow line for bike rent and a short inward vertical arrow line for bike return in the figure. Suppose the initial state of three stations is (s_1, s_2, s_3) = (1, 2, 3). The current *activity vector* at slice t is $(-2, 0, 0)$, meaning 2 bikes will be rented out at station s_1 at the end of slice t. Clearly, either s_2 or s_3 should move one bike to station s_1. However, when $k = 1$ without the look ahead feature, s_3 has the same position as s_2 in terms of the priority. If s_3 is selected to move one of its three bikes and the activity vector at slice $t + 1$ turns out to be $(0, 1, -3)$, s_2 has to move one of its bikes to s_3 at slice $t + 1$. On the other hand, when $k = 2$ with one slice look ahead, it is clear that it is better to move one bike from s_2 to s_1 to save one move.

On the other hand, look ahead may not always generate a better result. Fig. 8 (b) shows another example using the greedily look ahead approach, which makes a re-balancing move at the current slice so that



(a) An example of 2-hop look ahead outperforming 1-hop look ahead



(b) An example of 1-hop look ahead outperforming greedily look ahead

Fig. 10 Illustration of various look ahead schemes.

it can last with the maximum number of slices without re-balancing, given that all future activity vectors are known a priori. In Fig. 10 (b), station state is initially (1, 0, 1), with activity vectors being $(-1, 2, 0)$, $(-1, 0, 0)$, and $(0, -1, -1)$ for slices t, $t+1$, and $t+2$, respectively. Using the greedily look ahead approach, the re-balancing involves moving one bike from s_3 to s_1 so that the resulting configuration can survive two slices. However, at slice $t + 2$ with state (0, 2, 0), s_2 has to move at least one bike to s_3 to meet the demand. If we look at only one slice, i.e., $k = 1$, no action is needed at slice t, one bike moves from s_2 to s_1 at slice $t + 1$, and no action is needed at slice $t + 2$.

5.3 Extensions to look ahead

To develop more sophisticated greedy solutions, the k in k-hop look ahead can be dynamically adjusted. When the re-balance at the current slice can only last $k^{'}$ $(< k)$ slices, re-balancing is needed after $k^{'}$. If k is the value either used in a feasible k-hop look ahead or derived from the greedily look ahead, we can consider *greedily look and act ahead*. The main difference in this approach is that it can not only look ahead but also act ahead in the next k slices instead of limiting actions within the current slice only. In the example of Fig. 10 (b), the algorithm can pre-assign actions at future slices, for example, moving one bike from s_2 to s_1 for slice $t + 1$ when it starts its view for slice t. However, the complexity of such an algorithm will increase since actions at multiple slices need to be decided jointly with dynamic programming as a possible solution. More work is needed to gain insights on solution complexity as well as cost-effective trade-offs.

6 Challenges and Opportunities

This section studies future challenges and opportunities associated with BSSs, focusing on bike re-balancing. Finally, a comparison is drawn between dock and dock-less BSS in terms of their applications and future developments.

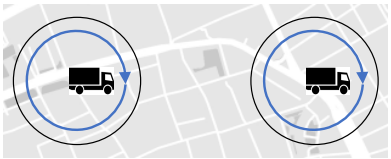
6.1 Model extensions

So far, the models we have discussed have various constraints. For example, in Fig. 10 we assume that all bike re-balancing activities can be done in one slice without any "cut". In addition, normal bike usage may not be completed in one slice (as shown in Fig. 10) with different global state values after each slice: 3 after slice t, 2 after slice $t + 1$, and 0 after slice $t + 2$. In reality: there are at least four models that can be constructed depending on the activity completion time, before or beyond the current slice, for each bike re-balancing activity and for each normal bike usage activity.

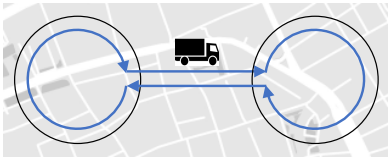
The capacity of trucks and workers will also affect the availability of bikes at each station when re-balancing activities go beyond one slice. In fact, such capacity will affect the value of the global state, making the service level less predictable from slice to slice. In addition, it is still open as to how to partition spatial and temporal domains so that other efficient solutions, other than slicing, can be explored.

6.2 Scalable design

As the size of a BSS increases in density and in the spatial domain, it is natural to study the scalability issue of different solutions. Solutions based on worker recruitment are scalable by design so we focus on solutions based on the number of trucks used. Results in Fig. 10 show scalability issues for a large coverage area and for a large number of stations. Fig. 11 shows another simple example to illustrate the challenges in scalable design. Suppose there are two populated regions separated by a given distance. Two trucks are used to individually cover regions. The question is whether to keep two regions separate with two trucks or merge two regions with one truck. Obviously, the former will save travel distance, resulting in a larger service frequency to



(a) Two individual circles



(b) One merged circle

Fig. 11 A simple example to illustrate subtle decisions with clustering.

stations, while the latter will reduce the cost (of using only one truck). This problem has been addressed in [30] on a UAV application under the ocean for sensor data collection applications. The problem in the BSS is newer, with a different setting and objective, and hence, worthy of further study.

The partition approach is a viable solution to address the scalability issue. One approach is based on geometric partitioning [31], assuming that stations and traffic are uniformly distributed, and more importantly, rent and return pairs exhibit locality so that most pairs fall within individual partition grids. A more effective method is clustering [32], such as k-means or balanced k-means, which can deal with non-uniform station distribution and non-uniform traffic distributions.

6.3 Gaming and incentive among BSS operators and workers

Game theoretical approaches [33] can be explored, especially for worker recruitment approaches. In general, BSS operators and workers form a Stackelberg game, while homogeneous and heterogeneous workers can form different sub-games with a Nash equilibrium. In game theoretical analysis, pricing [20] plays an important role on gaming analysis, especially the pricing design in relation to detour distance.

The incentive mechanism is a key in worker recruitment. Using the problem in [28] as an example as illustrated in Section IV B, Fig. 12 shows how the solution for detour minimization on one slice can be extended to multiple slices. Note that the ability of detour minimization also depends on the number of available workers. The more available workers, the more likely it is for the BSS operator to find suitable matches between workers and stations so that detour distance can be further reduced. Given a fixed size of the worker pool W, we assume that each worker has a probability p of joining the crowdsourcing activity for bike re-balancing. If it is assumed that each bike re-balancing activity will receive a fixed monetary or non-monetary (e.g., one or multiple free rides) award for each worker, then detour distance will play an important role in a user's willingness to participate (represented by p). That is, the p value changes over the rounds (moving slices along the time in Fig. 12), and a small detour at slice t will increase the value p for slice $t+1$, which in turn increases the worker pool $p \cdot |W|$ at slice $t+1$. A larger worker pool will generate a better set of matchings between workers and stations, resulting in smaller detours at slice $t+1$. Further investigation of the impact of this reinforcement incentive is another direction for future research.

6.4 Algorithmic solutions vs. ML with data analytic

This paper focuses on class algorithmic solutions. The main purpose is to gain more insights from these solutions. There are many other approaches, for example, various optimization approaches, including integer and linear programming [14, 16, 34, 35], which is usually more powerful; and machine learning [11, 13, 21], which is more effective with the support of a large data set.

One future challenge is how to integrate merits from different approaches. Currently, the machine learning (ML) approach is widely used in many subareas of bike re-balancing, including the determination of pricing in the incentive approaches [18, 21] and user mobility and traffic prediction [8, 9]. Many ML approaches use a blackbox approach without providing much insights that can benefit future design. One possible direction is to apply algorithmic and ML co-design to address a complex problem. For example, ML can be used for future data and traffic prediction while algorithmic solutions are applied using data and traffic information.

Another important issue is the *robustness* of a solution. By robustness, we mean the degree of deviation from the intended performance when there is perturbation of data. One solution can be low efficient but robust, while another solution is efficient but non-robust. Quantification of robustness for both algorithmic and ML solutions for bike re-balancing remains under-exploited, which deserves more research.

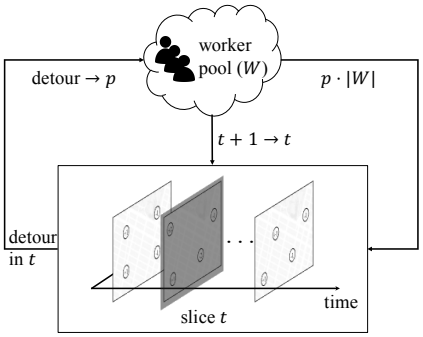


Fig. 12 Reinforcement incentive through slice iterations.

6.5 Integration of different system components

There are three other system components that affect the performance of bike re-balancing in addition to system balancing. They are system design, system prediction, and trip advisor. We will discuss here design issues that affect the performance of bike re-balancing.

System design includes deciding the location and capacity of stations (which usually indirectly determine the total number of bikes in circulation) in addition to the number of stations in a dock BSS. The number and capacity of stations as well as the number of bikes determines the cost of a BSS. More stations, capacity per station, and bikes will reduce the frequency of bike re-balancing will increase the cost of maintaining stations and bikes. However, having too many stations without increasing the number of bikes will also inadvertently increase the frequency of underflow per station. Given a fixed amount of expenditure, there is a trade-off between the cost on stations, the cost on bikes, and the cost on bike re-balancing.

System prediction clearly plays a key role in system design as well as system balancing. Almost all algorithmic and ML approaches rely on data available through either data collection or data predictions. The effectiveness of bike re-balancing depends on the accuracy of prediction. For example, a typical look ahead solution uses look ahead data that may or may not be accurate. In general, the quality of data deteriorates over time, especially ones used for multi-slice look ahead. The question is how to incorporate this data accuracy decay in the algorithmic design. The discount factor used in a typical reinforcement learning [21] can be applied. As data collection and prediction both incur cost, it is important to know the data granularity for each solution to achieve a good balance between its cost and effectiveness.

Trip advisor can act as an assistant to bike re-balancing with little or no service level degradation for the user. For example, when a user can pick up or return a bike from two equivalent sites, the advisor can make a judicial decision for the user that will benefit the BSS performance, e.g. rent a bike from an overflow station and return a bike to the underflow station. However, giving the user advice that may hurt the user's interest is more subtle, as it involves moral and legal issues, unless some kind of incentive is applied to compensate the user's loss.

6.6 Dock vs. dock-less BSS

Currently, both dock and dock-less BSSs exist in different urban cities in various countries. The U.S. has more dock BSSs while China has mostly of dock-less BSSs. It is clear that the dock-less BSS is more convenient to the user since they can pick up and drop off at any location, provided a sufficient number of bikes is available. However, the dock-less BSS creates various challenges to BSS operators in terms of management and maintenance. For example, it is reported in [36] that the bike-share oversupplies in China, resulting in huge piles of abandoned and broken bicycles (similar to the one shown in Fig. 3).

Here we focus on technical issues related to the co-existence of BSSs - dock and dock-less. One possible collaboration occurs among different BSS operators. For example, one truck can be used for bike re-balancing among different companies. This problem poses some unique challenges as each company owns different stations. Bikes from one company must be returned to the station of the same company. However, the truck can serve multiple companies and the corresponding Hamiltonian circle would consist of stations from different companies. Truck load can be shared among bikes from different companies as well.

7 A Bigger Picture

This section starts with a new classification of the transportation system and provides personal predictions on some future trends. We also include transportation systems with the goal of *shared mobility*, which includes all modes of travel that offer short-term access to transportation on an on-needed basis [37].

7.1 Classification

We first classify the transportation system into *active* and *passive* modes. An active system, such as a bus, taxi, or autonomous shuttle, moves around even when

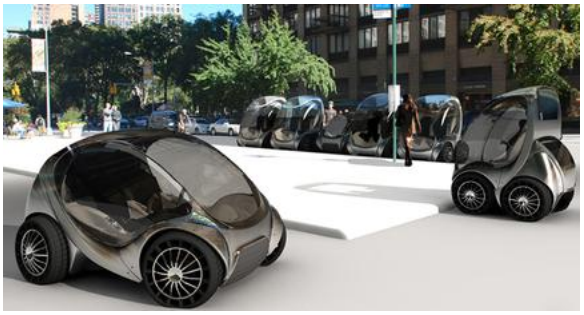


Fig. 13 The vision of folded cars in the MIT's CityCar project [40].

there is no demand from a user. The movement trajectory can be either *fixed* (such as a subway, bus, or autonomous shuttle [38]), *on-demand* (such as taxi for ride-hailing and Uber and Lyft for ride-sharing), or *hybrid*. There exist several hybrid modes, either in the actual system or in the research project. One mode is a restricted version of on-demand movement where the vehicle travels only along a subset of routes, e.g., every other street and avenue in Manhattan to save overall travelling distance at the expense of the user who may have to travel one street/avenue at both source and destination. Another mode is a flexible version of a bus where the vehicle follows a fixed route with controlled deviation based on demand [39]. A passive system, such as a ZipCar, bike, or scooter, remains motionless when there is no demand from a user. In a typical passive system, the vehicle is operated by the user, with and without power, such as e-scooters vs. regular scooters and motorcycles/e-bikes vs. bicycles.

The BSS belongs to the passive mode of the transportation system. the ZipCar, bike, and scooter systems are similar in terms of their functions. Therefore, re-balancing issues and their solutions discussed in this paper also apply to both ZipCar and scooters. However, while bikes and scooters address the first/last mile issue, the ZipCar solves the first/last ten-mile issue. These issues are different in scale and quantity.

7.2 Future of BSSs

Because of various government regulations, various forms of transportation systems aiming for shared mobility exist. For example, many cities put caps on the number of total vehicles/bikes a company can provide. Several Chinese cities have implemented license plate control. The chance is high for BSS to last for a long time. Traditional bikes are likely to be replaced by small-sized scooters (which are popular in Germany and France). Man-powered bikes and scooters will be superseded by e-bikes or two-wheeled e-scooters. Therefore, future BSSs may well be called scooter-sharing systems (SSSs). In the case of dock and dock-less BBSs, dock-less BSSs have largely disappeared in some cities in the US, including Washington, D.C. They are still going strong in the bicycle kingdom - China, but Ofo, the largest dock-less BSS in China, has recently suffered financially due to challenges in bike management and fierce competition [41].

In order for BSSs to flourish long-term, the following two issues need to be addressed: (1) *shared responsibility* and (2) *safety and regulation*. It is important that the user acts responsibly when placing a bike in a dock-less BSS. The question is how to encourage people to share responsibility? One possible solution is using a system that maintains personal credit scores. Higher ratings correspond to preferential access to services while lower ratings entail higher costs or blocked access to services. When BSSs are used together with other transportation systems, safety is an important issue. In the current system, there are three different lanes on a road: sidewalk, bike lanes, and car lanes. Some issues have been reported, such as when scooter riders blocked the sidewalk with parked scooters and rided on the sidewalk rather than using bike lanes. The first fatality involving an e-scooter was reported in September 2018 [42]. As BSS companies scale up their service to mini-cars, like the ones in MIT's CityCar project [40], will a fold-able mini-car be considered a regular car in a car lane or a bike in the bike lane, or will new regulation be introduced for such mini-cars? These problems certainly give food for thought in the future development of BSSs.

Looking forward, both active and passive modes of the transportation system will likely co-exist for a long period of time. Among the passive mode, BSSs in form of ZipCar, bikes, or scooters will certainly play an important role. The approaches discussed in this paper on re-balancing will still be relevant in managing such systems.

8 Conclusions

In this paper, we discussed various solutions to improve bike re-balancing in both dock and dock-less bike sharing systems (BSSs). We focused on algorithmic solutions to problems that span both time and space. Similar to the classic time-space view of a distributed com-

puting system with a set of communicating processes, a BSS can be represented as a set of stations with a given capacity. Bike re-balancing among stations can be represented as a slanted arrow line from one station to another station, similar to process communication in a distributed computing system. Several unique challenges in truck-based and worker-based solutions were examined, with discussion of various solutions and some open problems. Finally, we discussed several challenges and opportunities associated with bike re-balancing, including scalable design, gaming, and incentive among BSS operators and workers, algorithmic solutions vs. machine learning with data analytic, integration of different system components, and dock vs. dock-less BSSs.

Acknowledgement

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and CNS 1439672. Author would like to thank Yubin Duan who helped to collect data and run the simulation. A special thanks is due to Calton Pu who gave good suggestions on improving the vision part of this paper.

References

[1] S. A. Shaheen, E. W. Martin, A. P. Cohen, N. D. Chan, and M. Pogodzinski, "Public bikesharing in north america during a period of rapid expansion: Understanding business models, industry trends & user impacts, mti report 12-29," 2014.

[2] Wikipedia contributors. (2019) List of bicycle-sharing systems — Wikipedia. Available: https://en.wikipedia.org/w/index.php?title=List_of_bicycle-sharing_systems&oldid=877207320.

[3] F. Richter. Bike-sharing clicks into higher gear. Available: https://www.statista.com/chart/14542/bike-sharing-programs-worldwide/.

[4] L. Chen, D. Zhang, G. Pan, X. Ma, D. Yang, K. Kushlev, W. Zhang, and S. Li, "Bike sharing station placement leveraging heterogeneous urban open data," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 571–575.

[5] J. C. García-Palomares, J. Gutiérrez, and M. Latorre, "Optimizing the location of stations in bike-sharing programs: A gis approach," *Applied Geography*, vol. 35, no. 1-2, pp. 235–246, 2012.

[6] J. Liu, Q. Li, M. Qu, W. Chen, J. Yang, H. Xiong, H. Zhong, and Y. Fu, "Station site optimization in bike sharing systems," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 883–888.

[7] Wikipedia contributors. (2018) Facility location problem — Wikipedia. Available: https://en.wikipedia.org/w/index.php?title=Facility_location_problem&oldid=875216995.

[8] Z. Yang, J. Hu, Y. Shu, P. Cheng, J. Chen, and T. Moscibroda, "Mobility modeling and prediction in bike-sharing systems," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2016, pp. 165–178.

[9] Y. Li, Y. Zheng, H. Zhang, and L. Chen, "Traffic prediction in a bike-sharing system," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2015, p. 33.

[10] L. Chen, D. Zhang, L. Wang, D. Yang, X. Ma, S. Li, Z. Wu, G. Pan, T.-M.-T. Nguyen, and J. Jakubowicz, "Dynamic cluster-based over-demand prediction in bike sharing systems," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 841–852.

[11] J. Liu, L. Sun, W. Chen, and H. Xiong, "Rebalancing bike sharing systems: A multi-source data smart optimization," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1005–1014.

[12] S. Ghosh, M. Trick, and P. Varakantham, "Robust repositioning to counter unpredictable demand in bike sharing systems," 2016.

[13] Y. Li, Y. Zheng, and Q. Yang, "Dynamic bike reposition: A spatio-temporal reinforcement learning approach," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2018, pp. 1724–1733.

[14] J. Schuijbroek, R. C. Hampshire, and W.-J. Van Hoeve, "Inventory rebalancing and vehicle routing in bike sharing systems," *European Journal of Operational Research*, vol. 257, no. 3, pp. 992–1004, 2017.

[15] G. Erdoğan, M. Battarra, and R. W. Calvo, "An exact algorithm for the static rebalancing problem arising in bicycle sharing systems," *European Journal of Operational Research*, vol. 245, no. 3, pp. 667–679, 2015.

[16] M. Rainer-Harbach, P. Papazek, B. Hu, and G. R. Raidl, "Balancing bicycle sharing systems: A variable neighborhood search approach," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2013, pp. 121–132.

[17] C. Contardo, C. Morency, and L.-M. Rousseau, *Balancing a dynamic public bike-sharing system*. Cirrelt Montreal, 2012, vol. 4.

[18] A. Singla, M. Santoni, G. Bartók, P. Mukerji, M. Meenen, and A. Krause, "Incentivizing users for balancing bike sharing systems." in *AAAI*, 2015, pp. 723–729.

[19] C. Fricker and N. Gast, "Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity," *Euro journal on transportation and logistics*, vol. 5, no. 3, pp. 261–291, 2016.

[20] A. Waserhole and V. Jost, "Pricing in vehicle sharing systems: Optimization in queuing networks with product forms," *EURO Journal on Transportation and Logistics*, vol. 5, no. 3, pp. 293–320, 2016.

[21] L. Pan, Q. Cai, Z. Fang, P. Tang, and L. Huang, "A deep reinforcement learning framework for rebalancing dockless bike sharing systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 1393–1400.

[22] J. Hu, Z. Yang, Y. Shu, P. Cheng, and J. Chen, "Data-driven utilization-aware trip advisor for bike-sharing systems," in *Data Mining (ICDM), 2017 IEEE International Conference on*. IEEE, 2017, pp. 167–176.

[23] P. Cheng, J. Hu, Z. Yang, Y. Shu, and J. Chen, "Utilization-aware trip advisor in bike-sharing systems based on user behavior analysis," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[24] M. Charikar, S. Khuller, and B. Raghavachari, "Algorithms for capacitated vehicle routing," *SIAM Journal on Computing*, vol. 31, no. 3, pp. 665–682, 2001.

[25] Y. Duan, J. Wu, and H. Zheng, "A greedy approach for vehicle routing when rebalancing bike sharing systems," in *IEEE GLOBECOM*, 2018.

[26] O. O'Brien. Bike share map: Beijing. Available: http://bikes.oobrien.com/beijing/.

[27] Y. Duan and J. Wu, "Optimizing rebalance scheme for dock-less bike sharing systems with adaptive user incentive," in *Proc. of the 20th IEEE International Conference on Mobile Data Management (IEEE MDM 2019)*, 2019.

[28] Y. Duan and J. Wu, "Optimizing the crowdsourcing-based bike station rebalancing scheme," in *Proc. of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019)*, 2019.

[29] J. Wu, *Distributed System Design*. CRC press, 1998.

[30] H. Zheng, N. Wang, and J. Wu, "Minimizing deep sea data collection delay with autonomous underwater vehicles," *Journal of Parallel and Distributed Computing*, vol. 104, pp. 99–113, 2017.

[31] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[32] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[33] J. Zhang, P. Lu, Z. Li, and J. Gan, "Distributed trip selection game for public bike system with crowdsourcing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2717–2725.

[34] C. Kloimüllner, P. Papazek, B. Hu, and G. R. Raidl, "Balancing bicycle sharing systems: an approach for the dynamic case," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2014, pp. 73–84.

[35] I. A. Forma, T. Raviv, and M. Tzur, "A 3-step math heuristic for the static repositioning problem in bike-sharing systems," *Transportation research part B: methodological*, vol. 71, pp. 230–247, 2015.

[36] A. Taylor. The bike-share oversupply in china: Huge piles of abandoned and broken bicycles. Available: https://www.theatlantic.com/photo/2018/03/bike-share-oversupply-in-china-huge-piles-of-abandoned-and-broken-bicycles/556268/.

[37] What is shared mobility? Available: https://sharedusemobilitycenter.org/what-is-shared-mobility/.

[38] (2019) Rtd launches the first on-road deployment of an autonomous shuttle (ez10) in denver. Available: http://www.easymile.com/rtd-launches-the-first-on-road-deployment-of-an-autonomous-shuttle-ez10-in-denver/.

[39] J. Wu, S. Yang, and F. Dai, "Logarithmic store-carry-forward routing in mobile ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 735–748, June 2008.

[40] Wikipedia contributors. (2018) Citycar — Wikipedia. Available: https://en.wikipedia.org/w/index.php?title=CityCar&oldid=854799564.

[41] C. Campbell. (2018) The trouble with sharing: China's bike fever has reached saturation point. Available: http://time.com/5218323/china-bicycles-sharing-economy/.

[42] Wikipedia contributors. (2019) Scooter-sharing system — Wikipedia. Available: https://en.wikipedia.org/w/index.php?title=Scooter-sharing_system&oldid=880835966.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless net- works, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.