BRAVO: Improving the Rebalancing Operation in Bike Sharing with Rebalancing Range Prediction

SHUAI WANG and TIAN HE, University of Minnesota
DESHENG ZHANG, Rutgers University
YUANCHAO SHU, Microsoft Research Asia
YUNHUAI LIU, Peking University
YU GU, IBM
CONG LIU, The University of Texas at Dallas
HAENGJU LEE and SANG H. SON, Daegu Gyeongbuk Institute of Science and Technology

Bike sharing systems, which provide a convenient commute choice for short trips, have emerged rapidly in many cities. While bike sharing has greatly facilitated people's commutes, those systems are facing a costly maintenance issue – rebalancing bikes among stations. We observe that existing systems frequently suffer situations such as no-bike-to-borrow (empty) or no-dock-to-return (full) due to existing ad hoc rebalancing practice. To address this issue, we provide systematic analysis on user trip data, station status data, rebalancing data, and meteorology data, and propose BRAVO – the first practical data-driven bike rebalancing app for operators to improve bike sharing service while reducing the maintenance cost. Specifically, leveraging experiences from two-round round-the-clock field studies and comprehensive information from four data sets, a data-driven model is proposed to capture and predict the *safe* rebalancing range for each station. Based on this safe rebalancing range, BRAVO computes the optimal rebalancing amounts for the full and empty stations to minimize the rebalancing cost. BRAVO is evaluated with 24-month data from Capital, Hangzhou and NiceRide bikeshare systems. The experiment results show that given the same user demand, BRAVO reduces 28% of the station visits and 37% of the rebalancing amounts.

CCS Concepts: • **Human-centered computing → Empirical studies in ubiquitous and mobile computing**;

Additional Key Words and Phrases: Bike sharing, bike rebalancing system, demand prediction, urban data

ACM Reference Format:
Shuai Wang, Tian He, Desheng Zhang, Yuanchao Shu, Yunhuai Liu, Yu Gu, Cong Liu, Haengju Lee, and Sang H. Son. 2018. BRAVO: Improving the Rebalancing Operation in Bike Sharing with Rebalancing Range Prediction. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 44 (March 2018), 22 pages. https://doi.org/10.1145/3191776

1 INTRODUCTION

Bike sharing has become a popular choice for commuters to take a short travel, especially for the first/last mile trips. Currently more than 700 bike sharing systems have been launched in more than 50 countries in the world,

This work is supported by the National Science Foundation under grant CNS-1446640 and DGIST Research and Development Program (CPS Global Center) funded by MSIP.
Authors' addresses: Shuai Wang; Tian He, University of Minnesota, shuaiw@cs.umn.edu; Desheng Zhang, Rutgers University; Yuanchao Shu, Microsoft Research Asia; Yunhuai Liu, Peking University; Yu Gu, IBM; Cong Liu, The University of Texas at Dallas; Haengju Lee; Sang H. Son, Daegu Gyeongbuk Institute of Science and Technology.

2474-9567/2018/3-ART44 $15.00
https://doi.org/10.1145/3191776

operating approximately 806,200 bicycles at 37,500 stations [17]. These bike sharing systems have scored a great success in enhancing the efficiency of people's daily commutes. For example, the Capital Bikeshare members in Washington D.C. reduce their driving by 4.4 million miles per year [1].

On the other hand, the existing bike sharing systems face several practical operational issues. One of the most challenging issues is that the bike sharing service providers have to rebalance the bikes among hundreds or thousands of stations to ensure consistent bike/dock availability. This rebalancing operation is challenging in practice because of the following reasons. First, bike usage patterns are highly imbalanced among stations because of one-way trips. Second, numbers of rebalancing vehicles and human operators are limited while the rebalancing demand and cost are extremely high. For example, thousands of bikes need to be rebalanced every day in the Vélib' bike sharing system while the cost of rebalancing *one* bike is *three* dollars [8].

Since bike rebalancing is a critical issue in bike sharing systems, many studies have been proposed to solve different aspects of the problem, including demand analysis [7, 9, 11, 12, 19, 20], service analysis [15, 16], and rebalancing route [13, 14]. Most of the previous designs are either based on (i) limited sampling data collected through surveys and censuses, which are often dated and incomplete, or (ii) the released one-sided historical data. These two limitations result in unsatisfactory performance in rebalancing designs, such as long empty bike/dock duration, redundant or untimely rebalancing and excessive operating cost. For example, existing works only use the station status data including the available bikes and docks, to explore the usage pattern and provide guidance to bike rebalancing. However, we find that the *fluctuation of the available bikes and docks is not only caused by the user but also by the operators' own rebalancing activities*. In this case, the user demand analysis based on only one data set is inaccurate and lead to inefficient rebalancing operations.

Since almost all the bike sharing systems are equipped with sensors and communication devices, large-volume real-time data sharing for the bike sharing service becomes feasible. On the other hand, the first version of open data standard for bikeshare is adopted by North American Bike Share Association (NABSA) board in Nov. 2015 [3]. More and more bike sharing systems (i.e., at least 169 systems [4]) release API URLs for real-time data accessing [5].

To leverage this advantage, this paper proposes BRAVO – the first data-driven bike rebalancing app for operators. The BRAVO system not only collects the real-time status of each station (e.g., the available bikes and docks), but also collects (i) all the bike riders' trip information (e.g., the start/end station and the travel time), (ii) operators' rebalancing activity, and (iii) the real time meteorology data. Leveraging upon the four data sets, BRAVO (i) figures out the issues in current rebalancing activity, (ii) analyzes and estimates the safe rebalancing range, and (iii) computes the next targeted station and rebalancing amount. The computational results are sent to the BRAVO app and the operators rebalance the bikes among stations. In summary, our work's contributions are as follows:

• To the best of our knowledge, this work proposes the first practical data-driven bike rebalancing app. The app is built upon open data standard and thus can benefit hundreds of existing bike sharing systems with tiny changes.

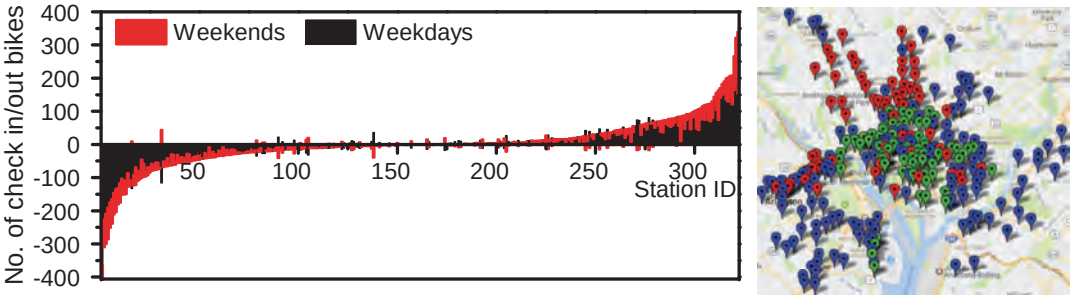
• Leveraging upon the comprehensive information of the four multi-event data sets, we build a novel model to capture and predict the safe rebalancing range for each station. Motivated by two-round daylong field studies across on weekdays and weekends, we find that a new rebalancing amount formulation is a real need, instead of the traditional rebalancing routing formulation. Specifically, the rebalancing amount formulation aims to minimize the total rebalancing cost while guaranteeing the safe rebalancing range. An optimal solution with low computational complexity is proposed to this problem.

• Our evaluation effort is comprehensive. We evaluate our design with 24-month data from three bike sharing systems. The evaluation results show that our rebalancing system reduces 28% of station visits and 37% of rebalancing amounts, given the same user demand.

The rest of the paper is organized as follows. Section 2 gives the motivation. Section 3 presents the design overview. Section 5, 6 and 7 introduce detail designs on demand prediction, rebalancing range and rebalancing amount calculation. Section 8 introduces implementation details. Section 9 provides evaluation results. Section 10 reviews related works. Finally, Section 11 concludes the paper.

2 MOTIVATION

In this section, we motivate our rebalancing design by revealing bike usage/demand imbalances in bike sharing systems. With empirical data, we statistically summarize the rebalancing activities and show the dilemma that (i) on one hand, the operators in existing bike sharing systems indeed pay great efforts in rebalancing, and (ii) on the other hand, there are still a large number of instances of no-bike-to-borrow or no-dock-to-return. To understand the root causes, we then inspect the rebalancing operations in details, showing their inefficiencies and the opportunities for improvements.



(a) System-wide accumulated numbers for check in/check out bikes in two weeks (b) Station distribution

Fig. 1. Imbalanced user demand in Capital bikeshare system

2.1 Demand Imbalances

In bike sharing systems, the operators rebalance bikes to meet user demand on available bikes/docks due to inherent imbalances of renting and returning amounts at various stations. Figure 1(a) shows the system-wide accumulated number of outgoing bikes (i.e., y-axis) in two weeks in 322 Capital bikeshare stations (i.e., x-axis). From the figure, we can clearly see that some stations have more rentals than returns (e.g., the first 100 stations with negative accumulated value of outgoing bikes in Figure 1(a)) while some stations have more returns than rentals (e.g., the last 100 stations with positive accumulated value of outgoing bikes in Figure 1(a)). Figure 1(b) shows the corresponding distribution of stations: the stations marked "red" have more rentals than returns while the stations marked "green" have more returns than rentals. We also find that some stations have opposite demand trends on weekdays and weekends. These inherent imbalances are caused by the users' trip demand. For example, there are more rentals than returns in residential district in the morning because many people borrow bikes as a commute tool to go to work. Correspondingly, there are more returns in downtown in the morning since many people drop their borrowed bikes there. To satisfy user demand subject to the imbalances, the operator needs to drive a truck to rebalance the bikes among the stations.

2.2 Current Rebalancing Amount and Result

Existing bike sharing systems face a dilemma that system operators pay great efforts in rebalancing bikes among stations to meet the users' demand while many users still complain that they cannot borrow/return a bike since the station is empty/full. This dilemma is reflected by the statistical result from the data sets. Figure 2 shows

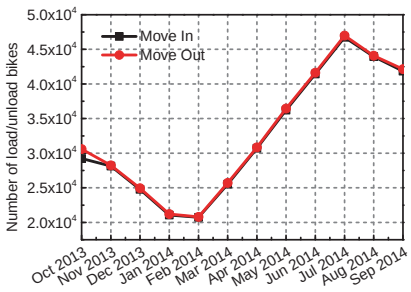


Fig. 2. Rebalancing Amounts

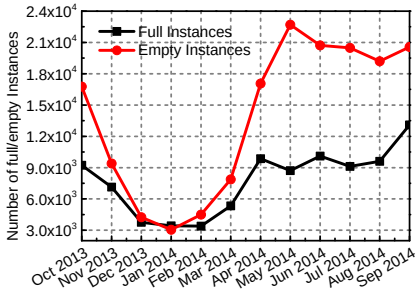


Fig. 3. Full/Empty Instances

the number of moved in/out bikes in one month from Oct. 2013 to Sep. 2014 in Capital bikeshare system. From the figure, we can see that the operators pay significant efforts in rebalancing the bikes among stations. On average, the operators move in 32,533 bikes and move out 32,803 bikes every month which costs about 200,000 dollars/month. It is the most expensive item among all maintenance expenses. On the other hand, we find that the system still faces severe empty bike/dock issues. Figure 3 shows the number of full/empty station instances in each month during the same period of time. On average, there are 7,732 full station instances and 13,880 empty station instances in each month.

2.3 Inefficiencies of Rebalancing Operations

Up to now, we know that the current bike sharing system faces an awkward situation that the current service cannot fully satisfy the users' demand though the rebalancing amount is extremely large. We thus perform an intensive study about the full/empty station situation as well as the rebalancing operation. In the following, we first reveal the inefficiencies in existing systems with typical examples, and then show the statistic results.

In Figure 4 and 5, we show typical examples of inefficiencies of the rebalancing operation in Capital bikeshare system. The figures plot the variation of the number of bikes in every one minute in one day. In both figures, we use gray line to show the capacity of the station. For example, the capacity of station "19th St & Pennsylvania Ave NW" is 15. The variation of the number of bikes caused by the "Move In" and "Move Out" rebalancing operations are plotted using the red short-dash and green short-dot line separately. We use the black line to represent the normal fluctuation, which is caused by users' "Borrow" and "Return" activities.

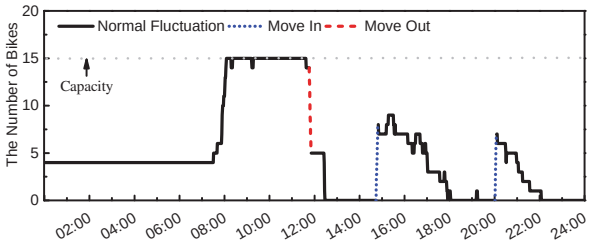


Fig. 4. No. of bikes in Station–19th St & Penns. Ave NW

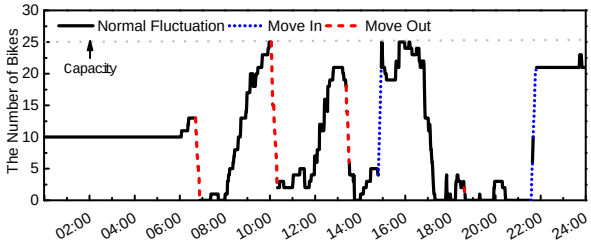


Fig. 5. No. of bikes in Station–Washi. & Indep. Ave SW/HHS

• **Slow rebalancing response:** The response to the empty/full station situation is slow and thus the stations suffer a long duration of empty bike/dock situation. From Figure 4, we can find that the station goes to full at around 8 A.M. since many users return bikes at this station. The full station status lasts about 4 hours before

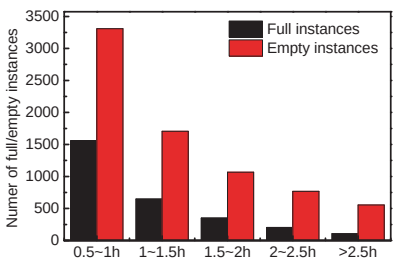


Fig. 6. Full/empty cases by diff. intervals

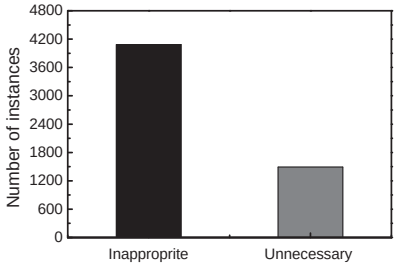


Fig. 7. Inefficiencies Amounts

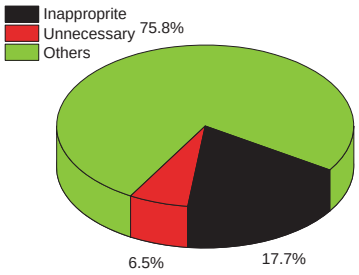


Fig. 8. Percentage

the rebalancing operator takes an action. After that, the station goes to empty at about 12:30 P.M., 6 P.M. and 10 P.M. In either case, the operator does not take a rebalancing operation within 2 hours. Figure 6 shows the system-wide full/empty instances for different response time intervals in Oct 2015. From the figure, we can see that there are about 3310 empty instances and 1560 full instances in time interval $[0.5h, 1h]$ in one month. And there are 4097 empty instances and 1312 full instances whose response time intervals are greater than one hour. From this statistic results, we can conclude that the current rebalancing response is not quick enough to meet the user demand.

• **Inappropriate rebalancing amount:** From Figure 4, we find that the operator moves out too many bikes at 12 P.M.. The station thus goes to empty soon at about 12:30 P.M. and the operator needs to send some bikes back to this station. But the amount of "Move In" bikes at 2:40 P.M. and 8 P.M. is not enough and the station goes to empty. The operator needs to take another trip to the station to move in some bikes. Figure 7 plots the statistics of inappropriate moving instances in Capital bike sharing system for Aug. 2014. Therefore, we define inappropriate rebalancing instance as the event that an empty/full instance occurs in the time interval between adjacent same types of rebalancing (i.e., both rebalancing are loading/unloading) in half-day. From the figure, we can see that there is a large amount of inappropriate moving instances, i.e., 4,084 instances in one month, which occupies 17.7% of the total number of rebalancing (Figure 8). These operations cause the station goes to empty/full and rebalancing operators need to go to that station again to load/unload bikes.

Root cause of inappropriate rebalancing: The inappropriate rebalancing phenomenon occurs because operators lack the knowledge of the suitable rebalancing amount for a station and they load/unlaod bikes simply based on their experience.

• **Unnecessary rebalancing activities:** Figure 5 shows a typical example of unnecessary rebalancing. From the figure, we can see that the operator moves out some bikes at 1:30 P.M. Then the station goes to empty and the operator moves back those bikes at 3 P.M. If the operator does not make the rebalancing at 1:30 P.M. and 3 P.M., the system actually works as well as the system after the rebalancing operation. Figure 7 shows the statistics of unnecessary rebalancing instances in Capital bike sharing system for Aug. 2014. We define the unnecessary rebalancing instance as the event that an empty/full instance occurs in the time interval between adjacent different types of rebalancing (i.e., one rebalancing is loading and the other is unloading) in half-day. From the figure, we find that there are many unnecessary moving instances, i.e., 1,490 instances in one month, which is about 6.5% of the total number of rebalancing (Figure 8). Those unnecessary rebalancing operations increase the maintenance cost while the full/empty situation is not alleviated.

Root cause of unnecessary rebalancing: Our field study indicates that unnecessary rebalancing is caused by the "robbing Peter to pay Paul" activity. When a operator finds that there are no bikes/seats in the truck during the rebalancing, instead of going back to the depot, she/he will load/unload some bikes from nearby stations, leading to unnecessary rebalancing at those nearby stations.

3 SYSTEM OVERVIEW

To address the rebalancing inefficiencies and improve the service quality, this paper proposes BRAVO – a data-driven bike rebalancing app for operators. In the following of this section, we first introduce how existing rebalancing systems work. Then, we introduce the system requirement and challenges, followed by the architecture of BRAVO system.

3.1 Current Rebalancing Systems

In existing bike sharing systems, operators drive trucks to rebalance bikes among stations to avoid empty/full instances every day. From the collaboration with existing bike sharing systems, we learn that rebalancing operators obtain the real-time information of (nearly) full/empty stations through either instructions from the dispatching center or carry-on devices such as laptop. For example, NiceRide bikeshare system adopts the Oobrien system [4], which provides 169 bike sharing systems with real-time full/empty station information. Based on the full/empty station map, operators determine the rebalancing route as well as the rebalancing amount through their experience.

3.2 System Requirement

To help existing bike sharing systems improve the rebalancing efficiency, this paper proposes BRAVO, which is designed to meet the following requirements. Except for providing basic functions, i.e., (i) displaying all the stations in a map with station status (i.e., the number of bikes/docks in a station), and (ii) highlighting (nearly) full/empty stations, BRAVO should automatically determine and display (i) the next targeted station and (ii) rebalancing amount to reduce the rebalancing cost based on the operator's location. Based on our field study, BRAVO should be designed to work smoothly when (i) operators need to suspend the rebalancing task to deal with emergency events such as failed borrow/return or bike damage report, and (ii) data are not updated in time.

3.3 Challenges

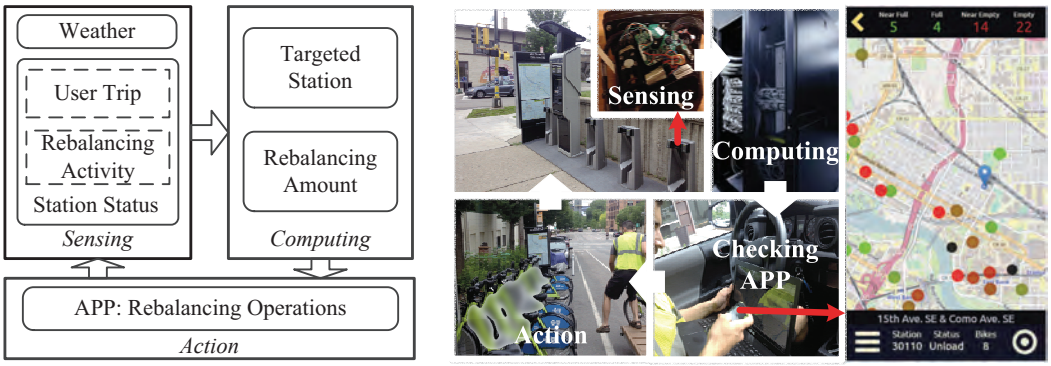
To meet these requirements, three major challenges need to be addressed in developing BRAVO.

- As mentioned in motivation, there are a large amount of inappropriate rebalancing operations in existing bikeshare systems and the rebalancing cost is extremely high. In order to address these inefficiencies, BRAVO needs to accurately predict the rebalancing amount for full/empty stations.
- Most of bike stations are located near narrow roads. The operators thus have to drive small trucks with limited capacity. Through our field rebalancing with operators, we find that it is quite common that there are no enough bikes/seats in the truck and the operators have to load/unload some bikes to/from nearby (transfer) stations. BRAVO needs to avoid the "robbing Peter to pay Paul" problem to eliminate the unnecessary rebalancing activities.
- In order to display the result in real time, BRAVO needs to efficiently calculate the next targeted station as well as rebalancing amount for bike sharing systems with hundreds or thousands of stations.

3.4 System Architecture

Keeping the above challenges in mind, we aim to build an efficient data-driven bike rebalancing system to help operators improve bike sharing service and reduce rebalancing cost. Figure 9 shows the architecture of BRAVO system which consists of three components.

• **Sensing Component:** The sensing component in each dock (Figure 9(b)) serves as the data feeding module, which collects the four data sets: (i) the user trip data, (ii) the station status data, (iii) the operators' rebalancing data, and (iv) the weather data. These sensing data are sent to the server in the dispatching center in real time and used as the input of the computing component.



(a) System Diagram (b) Physical Infrastructure

Fig. 9. The architecture of BRAVO system

• **Computing Component:** This component is responsible for computing next targeted station and rebalancing amount. The results are sent to the client, i.e., the BRAVO app in operators' phone.
• **Action Component:** After receiving the results from the server, the the BRAVO app displays the rebalancing amount and next targeted station to operators who will drive trucks to the station and load/unload the designated number of bikes. Note that rebalancing operators' behavior will impact station status. The operators' rebalancing information is thus collected by the sensing component and sent to the central server for future analysis.

4 THE KEY IDEA OF BRAVO

The key idea of BRAVO is to figure out *safe rebalancing range* for each station which will (or try the best to) ensure that the station will not face full/empty instances before the next rebalancing. Safe rebalancing range is defined as follows.

Definition 4.1. (**Safe Rebalancing Range**) Given the demand on bikes/docks and the current station status, the safe rebalancing range is the range of the initial number of bikes will be load/unload by the rebalancing operator to satisfies the demand without running out bikes/docks at stations.

For full/empty stations, safe rebalancing range is used to determine the final rebalancing amount. For non-full/non-empty stations, this range is used to determine the nearby transfer station to avoid the "robbing Peter to pay Paul" problem.

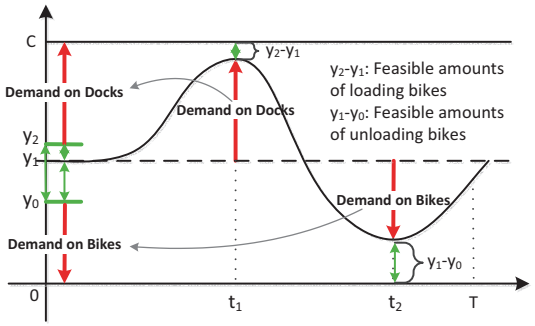


Fig. 10. The overview of the proposed approach

Figure 10 illustrates how we obtain safe rebalancing range. In this figure, the X-axis represents the time and the Y-axis represents the number of bikes in a station. Let the time stamp '0' in the figure be the starting instant of one rebalancing activity and 'T' be the the starting instant of the next rebalancing activity. The curve in the figure shows the variation of bikes in a station in time interval $[0, T]$. From the figure, we can see that in time interval $[0, t_1]$, the number of bikes in the station increases since a few users return bikes to the station. In time duration $[t_1, t_2]$, some users borrow bikes from this station and the number of bikes in the station decreases. From these users' borrow/return activities, we get the user demand on bikes and docks in time interval $[0, T]$, which is shown with read arrows in Figure 10.

Given the demand on bikes and docks, we find that (i) the station would not face full instances even if $y_2 - y_1$ bikes were loaded to the station, and (ii) the station would not face empty instances even if $y_1 - y_0$ bikes were unloaded from this station. In other words, the station will not go to full/empty states when there are y_0 to y_2 docked bikes at time 0, given the demand on bikes and docks in duration $[0, T]$. After obtaining feasible ranges of the numbers of bikes at all station, we now can decide the next targeted station and rebalancing amount.

To achieve the above idea, we need to address the following technical problems. First, we need to capture the demands on bikes and docks from the four multi-event data sets, and predict them accurately in advance to reduce the empty/full instances (Section 5). Based on the bike/dock demand, we need to figure out the safe rebalancing range for each station, given the perdition error (Section 6). Based on the safe rebalancing range, we need to design an efficient practicable rebalancing algorithm to determine the next targeted station and rebalancing amount (Section 7) in real time.

5 DEMAND ANALYSIS

In this section, we introduce how to utilize the data sets to predict the demand on bikes/docks for each station. We first introduce how to obtain the prediction target, followed by our proposed prediction algorithm and its prediction accuracy analysis.

5.1 Demands on Bikes and Docks

We now introduce how to calculate the prediction target, i.e., the demand on bikes/docks at each station, based on the user trip data. Given a time period T and a station S_i, we extract both borrow and return records related to station S_i. We divide the time period T into small time slots. In the calculation, the time slot length is set to be as short as possible, i.e., one minute, to keep as much demand information as possible. For each time slot t_i, we compute the net demand d_{t_i} in this short time duration. In detail, we set the initial net demand value be zero. The net demand value will be deducted by one if there is a borrow activity and the demand value will be added by one if there is a return activity.

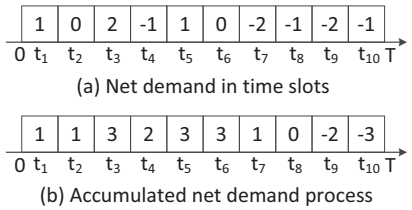


Fig. 11. An example of user demand calculation

Figure 11(a) shows an example of computing the net demand in each slot. In this example, the time period T is divided into ten slots. The value in each time slot is the net demand. For example, the value in slot t_3 is two,

which represents that there are two more returns than borrows in this time slot and the net demand d_{t_3} is two. Based on the net demand d_{t_i} in each time slot, we then calculate the accumulated net demand process, which is defined as follows:

Definition 5.1. (**Accumulated Net Demand Process**) Given a time duration, let the series of the time frames be $\{t_1, t_2, \ldots, t_m\}$. Then accumulated net demand process is the sequence of the accumulated net demand – $D = \{D_{t_1}, D_{t_2}, \ldots, D_{t_m}\}$, where $D_{t_i} = \sum_{j=1}^{i} d_{t_j}$.

Figure 11(b) shows the corresponding accumulated net demand process of the net demand in Figure 11(a). In Figure 11(b), each value D_{t_i} in time slot t_i is the sum of the net demand from time t_1 to t_i. After obtaining the accumulated net demand process, we are able to compute the demand on bikes/docks for each station through the following lemma.

Lemma 5.2. *Let the accumulated net demand process in time period T be $D = \{D_{t_1}, D_{t_2}, \ldots, D_{t_m}\}$. Then the demand on bikes is $max\{0, -min(D)\}$ and the user demand on docks is $max\{0, max(D)\}$.*

Proof. Given an accumulated net demand process D, the demand on bikes in *all* the slots will be satisfied if the amount of bikes $max\{0, -min(D)\}$ is provided. Similarly, if the amount of docks $max\{0, max(D)\}$ is provided, then the demand on docks in *all* the slots will be satisfied as well. We thus have Lemma 5.2. □

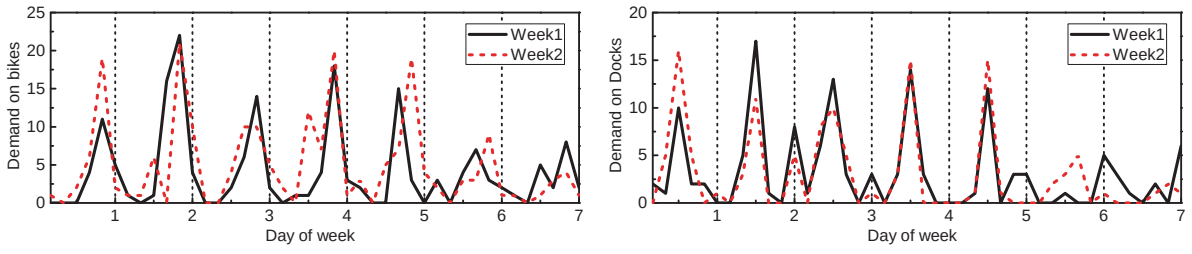
Based on Lemma 5.2, we now can calculate the demand on bikes/docks for each station. Take the case in Figure 11 as an example. Here the accumulated net demand process is $D = \{1, 1, 3, 4, 3, 3, 1, 0, -2, -3\}$. We have $max\{0, -min(D)\} = 3$ and $max\{0, max(D)\} = 4$. Therefore, the demand on bikes is three and the demand on docks is four in this example.

5.2 Demand Prediction

In previous section 5.1, we introduce how to calculate user demand on bikes and docks, given historical data. Although the best predictor of future behavior is past behavior, we cannot naively predict the demand using Historical Average (HA) or Auto-regressive Model (AR), because a few other external factors (e.g., weather condition, time of the day, and day of the week) affects the rebalancing demand and operations significantly.

In this section, we thus apply a statistical learning technique – gradient boosted regression trees (GBRT) to predict the demand on bikes/docks based on historical user trip, corresponding time and meteorology data.

In the following, we first introduce the extracted features in Section 5.2.1, followed by the GBRT model in Section 5.2.2. Then, we show the prediction error of GBRT along with other classic prediction algorithms in Section 5.2.3.



(a) Demand on bikes (b) Demand on docks

Fig. 12. Demands on bikes/docks at one station in two weeks in NiceRide bike sharing system

5.2.1 Feature Extraction. **Time Features:** The demand on bikes/docks is closely related to the time factor. Figure 12 shows the demand on bikes and docks at one station in NiceRide bike sharing system in different time and on different days. From the figure, we can see that the demand on bikes and docks on weekdays have similar temporary patterns. The station has (i) high demands on bikes on afternoon rush hours (Figure 12(a)), (ii) high demands on docks on morning rush hours (Figure 12(b)) and (iii) has relatively low demands in noon time and night time. Besides, the demand on weekends or holidays is relatively lower than that on weekdays. We extract three features: the time of a day, the day of a week, and holiday, for the prediction of the demand on bikes/docks. **Meteorology Features:** The meteorology condition has great impact on people's commute choices, especially for the choice of bikes. We thus collect meteorology data and extract features for the prediction of the demand on bikes/docks. We identify three meteorology features: weather, temperature, and wind speed. Among these features, the weather feature is divided into three categories: sunny (or cloudy), rainy, and snowy. Figure 13 shows the demand on bikes/docks at one station in one day from 2015/5/4 to 2015/5/4 in NiceRide bike sharing system. From the figure, we can see that there are some anomalous points in the red dotted box on 2015/5/5, 2015/5/7 and 2015/5/10 which are rainy days.

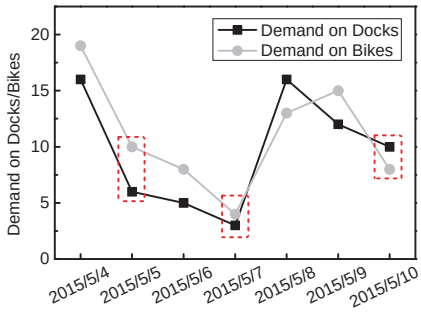


Fig. 13. Influence of weather: Demands on bikes/docks at one station in NiceRide bike sharing system

Online Demand Feature: The demand on bikes/docks may change dramatically although the time and meteorology features are the same. For example, when an event, e.g., a football game, happens, it's likely that the demand on bikes in nearby stations will increase dramatically. In contrast, the demand drops dramatically when the docks or station kiosks malfunction. Such events usually last a period of time and the demand in current time window is highly relevant to the demand in the next time window. Overall, the one line demand feature reflects the comprehensive situation of the factors including time, meteorology, and accident event, which is helpful information to infer the demand in the next time window.

5.2.2 The GBRT Model. After identifying the features, we are able to build learning models to predict the demand on bikes/docks. In our design, we apply the gradient boosted regression trees (GBRT) model which is one of the most effective machine learning models for prediction. GBRT provides an excellent fit of the predicted values to the observed values, even if the specific nature of the relationships is very complex (e.g., nonlinear).

The general idea of GBRT is to compute a sequence of simple regression trees, $d_1(x), d_2(x), \ldots, d_r(x)$, where each successive tree is built to predict the residual of the preceding trees, as shown by Eq. 1 and Eq. 2:

$$d_i = argmin_g \sum_{t=1}^{N} L(y_T - D_{i-1}(x_T), d(x_T)). \tag{1}$$

$$D_{i-1}(x) = \sum_{l=1}^{i-1} d_l(x). \tag{2}$$

Here, L is a loss function and $\{x_T, y_T\}_{t=1}^{N}$ is the training data set. Predictions are made by combining decisions of the regression trees – $d_1(x), d_2(x), \ldots, d_r(x)$, as shown by Eq. 3:

$$D(x) = d_1(x) + d_2(x) + \ldots + d_r(x). \tag{3}$$

In the prediction of demand on bike/docks, the variables x_T are features corresponding to period T, which have significant influence on the demand; y_T is the ground truth, i.e. the actual demand in period T. With the time and meteorology features, we can obtain a historical data set to train a GBRT model for online prediction.

5.2.3 Prediction Error. In order to confirm the effectiveness of our model, we conduct experiments to compare our methods with two baseline, i.e., Historical Average (HA) and Auto-regressive Model (AR). The HA model uses the average of historical observations for the same time and location to forecast the future data [10]. To predict the demand for a specific time period with the HA model, we first find out historical days with same day/time values, and then average the demand results from these periods. The second baseline, i.e., the AR model is widely used for time series prediction and is adopted to estimate the number of borrowing/returning information for bike sharing system [18]. This model leverages the demand information of the most recent p time windows for future prediction. Parameters are determined using historical data and the least squares method. The order of AR model (i.e., p) is set to five.

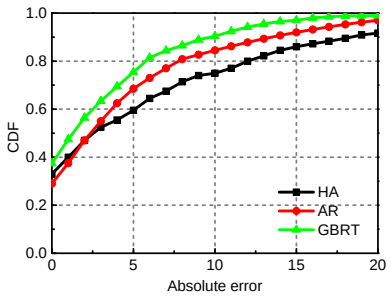


Fig. 14. The CDF of absolute error

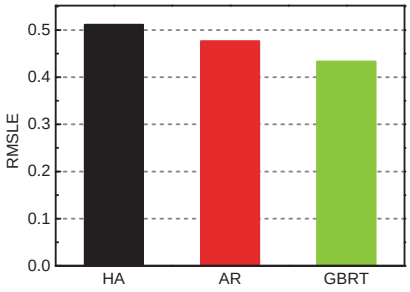


Fig. 15. The RMSLE of demand prediction

Figure 14 presents the overall prediction performance across all stations. We adopt the first 20 days of each month to train the GBRT model, and predict the demand on bikes/docks in remaining days. Compare the prediction results with ground truth, we can obtain the CDF of the absolute error (i.e., the difference between the predicted value and ground truth). From Figure 14 we can see that our approach outperforms the HA and AR models. For example, 90% of the absolute error is less than 9 in our approach while the corresponding value for the AR model is 14. It is even worse for the HA model whose absolute error is 19.

We also evaluate the prediction performance with Root Mean Squared Logarithmic Error (RMSLE), which is computed as $RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(\hat{x}+1) - \log(x+1))^2}$. Here $\hat{x}$ and x are the values of prediction and ground truth respectively while n is the number of predictions. Figure 15 shows the RMSLE of HA, AR and GBRT methods. The RMSLE of GBRT is as low as 0.433 while the RMSLEs of HA and AR are 0.511 and 0.476 respectively. Both absolute error and RMSLE results prove the effectiveness of the proposed GBRT model which takes time, meteorology and online demand features into consideration.

6 SAFE REBALANCING RANGE

In previous section, we introduce how to predict the demand on bikes/docks. In this section, we introduce how to compute the *safe rebalancing range* of bikes using this demand prediction information.

Lemma 6.1. *Let the station capacity be C and the number of bikes in the station be b. Then the rebalancing range is given by $[b - (C - \lceil\widehat{D_d} + \varepsilon\rceil), b - \lceil\widehat{D_b} + \varepsilon\rceil]$, where $\widehat{D_b}$ and $\widehat{D_d}$ are the predicted demands on bikes and docks separately. ε is the prediction error.*

Proof. To obtain the rebalancing range, we first calculate the feasible interval of bikes in a station that satisfies the demands on both bikes and docks, which is $[\lceil\widehat{D_b} + \varepsilon\rceil, C - \lceil\widehat{D_d} + \varepsilon\rceil]$. After obtaining the interval of the feasible amount of bikes, we calculate the rebalancing range by deducting the feasible amount of bikes from the current number of bikes in the station (i.e., the current station status). We thus get Lemma 6.1. □

We use the example in Figure 11 to illustrate how to calculate rebalancing range. We assume that the station capacity in this example is ten. Then we predict the demands on bikes and docks which are three and four separately. We thus have the feasible amount of bikes – $[3, 6]$, which represents that the demands on both bikes and docks will be satisfied if there are three to six bikes at this station. Given the interval of the feasible amount of bikes $[3, 6]$, if the station has less than three bikes, say one bike, then the operator needs to rebalance $[-5, -2]$ bikes, i.e., *loading* two to five bikes to the station. If the number of bikes in the station is greater than six, say eight, and the operator needs to rebalance $[2, 5]$ bikes, i.e., *unloading* two to five bikes from the station. There is no need to rebalance bikes if $0 \in [b - (C - \lceil\widehat{D_d} + \varepsilon\rceil), b - \lceil\widehat{D_b} + \varepsilon\rceil]$.

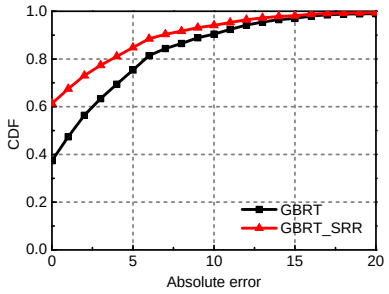


Fig. 16. The CDF of absolute error

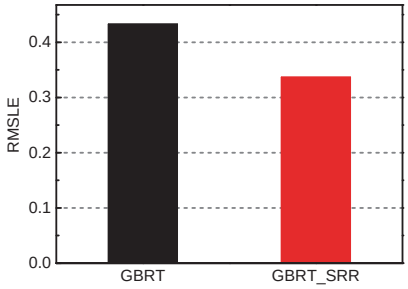


Fig. 17. The RMSLE of range prediction

Note that the safe rebalancing range design reduces the impact of demand prediction errors introduced by the GBRT model in previous section since as long as the predicted range sits *within* the range of ground truth, the prediction result satisfies rebalancing demand without causing empty bikes/docks in a station. In addition, the error compensation (i.e., ε) further helps the rebalancing range prediction sit in the range of ground truth. Figure 16 shows the CDF of the absolute error of the safe rebalancing range prediction (marked as "$GBRT_SRR$"). From the figure, we can see that 90% of the absolute error is less than 6. Figure 17 shows that the RMSLE of the safe rebalancing range prediction (i.e., *GBRT_SRR*) reduces to 0.337. Both Figure 16 and Figure 17 prove that the safe rebalancing range design reduces the impact of demand prediction errors.

In practical, it is possible that (i) the lower bound of the rebalancing range is greater than the station capacity, i.e., $b - (C - \lceil\widehat{D_d} + \varepsilon\rceil) > C$, (ii) the upper bound of the rebalancing range is less than the opposite number of the station capacity, i.e., $b - \lceil\widehat{D_b} + \varepsilon\rceil < -C$, and (iii) the lower bound of the rebalancing range is greater than the upper bound of the rebalancing range, i.e., $b - (C - \lceil\widehat{D_d} + \varepsilon\rceil) > b - \lceil\widehat{D_b} + \varepsilon\rceil$. These three situations mean that no matter how we set the initial number of bikes, it cannot satisfy the demand on bikes and docks in the time duration T. At this condition, our design tries the best to satisfy the demand on bikes/docks by setting the rebalancing amount (i) C, (ii) -C, and (iii) $(b - (C - \lceil\widehat{D_d} + \varepsilon\rceil) + b - \lceil\widehat{D_b} + \varepsilon\rceil)/2$.

7 TARGETED STATION & REBALANCING AMOUNT

In the previous section, we introduce how to estimate the safe rebalancing range in one station. In bike sharing systems, there may have dozens/hundreds of full/empty stations. We thus need to provide an efficient design to satisfy the rebalancing amount in all the full/empty stations while reducing the rebalancing cost as much as possible.

7.1 Targeted Station Calculation

From the collaboration with existing bike sharing systems, we find that operators may not adopt a complete pre-calculated route. This is because that operators also have to deal with events such as failed returns/borrows or bike damage during the rebalancing task. Operators also may not follow the specific path for the targeted station not only because they have their own habitual pathes but also because of constrains from roads and parking position (to fit with a truck with trailer). Therefore, in BRAVO, we only provide the operator the next targeted station's ID (address).

Since the problem of rebalancing bikes among full/empty stations is essentially the traveling salesman problem (TSP) [14].Our BRAVO design simply runs the approximate algorithm for TSP where each path's weight is the Euclidean distance. Our design then picks the first station from the sequence of returns as the targeted station. In practical rebalancing tasks, it is common that there is no enough bikes/docks in the truck because of its capacity limitation. At this situation, instead of guiding the operator going back to the depot, BRAVO selects the nearest station with the condition that the amount of bikes still sits in the safe rebalancing range after loading/unloading the required number of bikes.

7.2 Rebalancing Amount Calculation

After figuring out the next targeted station, we aim to compute the optimal rebalancing amount for these full/empty stations to reduce rebalancing cost. This is meaningful in practice since an optimal rebalancing amount design not only (i) reduces the total amount of loading/unloading bikes but also (ii) avoids empty-bike/empty-seat situations in the rebalancing truck and thus reducing the cases of grabbing bikes from nearby stations which may cause the "robbing Peter to pay Paul" problem.

In BRAVO, the rebalancing cost is defined as T – the total amount of loading/unloading bikes for these full/empty stations. In the following, we formally formulate the rebalancing amount calculation problem. The notations used in the formulation are shown as follows:
(i)S: the union set of the depot and all full/empty stations $S = \{S_0, S_1, \ldots, S_{|S|-1}\}$.
(ii)S_0: we use S_0 to represent the depot since it can be seen as a big station with zero rebalancing demand. Let V be the union of S_0 and the set of stations with rebalancing demands. V is a subset of S, i.e., $V \subseteq S$.
(iii)b_i: the number of bikes at station i before the rebalancing operation starts.
(iv)C_i: the capacity of station i.
(v)C_t: the capacity of the rebalancing truck.
(vi)$A[|S|]$: the array of full/empty stations which follows the sequence computed in targeted station calculation. $B[|V|]$ the corresponding station array which includes full/empty stations along with possible transfer stations.
(vii)$[L[|V|], U[|V|]$: the corresponding rebalancing range of station array $B[|V|]$.
(viii) x_i, the number of loading/unloading bikes at station S_i. When x_i is a positive value, it represents that the operators unload bikes from stations and move the bikes to the truck. When x_i is a negative value, it represents that the operators move the bikes from the truck and load bikes to stations.

$$min(\sum_{i=1}^{|V|} |x_i|)$$

s.t.

$$L[i] \leq x_i \leq U[i], \forall i \in [1, |V|]$$

$$0 \leq \sum_{i=1}^{j} x_i \leq C_t, \forall j \in [1, |V|]$$

The objective function minimizes the total amount of loading/unloading operations for all full/empty stations. The first constraint represents that the rebalancing amount x_i in each station will not exceed the safe rebalancing range. The second constraint guarantees that the total number of bikes in the truck is greater than or equals to zero and less than or equals to the truck capacity C_t after the rebalancing operation in each station.

ALGORITHM 1: Rebalancing Amount Algorithm

```
1  Initialize the rebalancing cost table T[|V| + 1][C_t + 1] = ∞;
2  Flip the array B[|V|], L[|V|], U[|V|];
3  for each i = 1 to |V| + 1 do
4      for each j = 1 to |C_t| + 1 do
5          if i == 1 then
6              Cost Table Update Preprocessing;
7              for k ∈ [L(i), U(i)] do
8                  tempT = |k|;
9                  Update table if tempT < T[i][j]; x_i = k
10         else if i == |V| + 1 then
11             Cost Table Update Preprocessing;
12             tempT = (j − 1) + T[i − 1][j];
13             Update table if tempT < T[i][j];
14         else
15             Cost Table Update Preprocessing;
16             for k ∈ [L(i), U(i)] do
17                 tempT = |k| + T[i − 1][j ± |k|];
18                 Update table if tempT < T[i][j]; x_i = k
19 Return x_i and min(T);
```

Optimal Solution: In this rebalancing amount calculation problem, we observe that *the total rebalancing cost of n stations equals to the sum of rebalancing cost at station i and the rebalancing cost of n − 1 stations which is an overlapped sub-problem of the original one.* We thus propose an optimal algorithm using dynamic program. The pseudo-code of the algorithm is shown in Algorithm 1. We initialize the value in the rebalancing cost table to be infinity (line 1). The algorithm flips the array $B[|V|]$, $L[|V|]$, and $U[|V|]$, and thus the algorithm traverses the stations in a backward way (line 2). Then for each station (line 3), the algorithm checks all the possible number of coming bikes in the rebalancing truck (line 4). The algorithm first updates the rebalancing cost table for the last station (line 5–9) since the rebalancing activity in this station will not impact others. Starting from this optimal substructure, the algorithm attempts to update the rebalancing cost table for other stations and the depot (line 10 – 18).

Before updating the rebalancing cost table, the algorithm executes the *rebalancing cost table update preprocessing* in the stations and depot (line 6&11&15), which has two steps. First, the algorithm checks that the station needs

to load or needs to unload bikes. This information can be inferred from the safe rebalancing range. Second, the algorithm checks whether there are enough bikes or seats in the truck to load or unload the bikes. If there are no enough bikes or seats in the truck, a punitive rebalancing cost will be added.After the rebalancing cost table update preprocessing, the algorithm computes the rebalancing cost for other stations based on the optimal substructure computed in the last station. In detail, the algorithm computes the rebalancing cost with all possible rebalancing amount within the safe rebalancing range (line 16–17). The rebalancing cost table will be updated if a shorter rebalancing cost is found (line 18). Finally, the algorithm returns the minimized rebalancing cost (line 19). The complexity of Algorithm 1 is $O(C_t^2|V|)$ since the algorithm needs to traverse all the stations with all possible initial number of bikes in the truck and all possible rebalancing amounts.

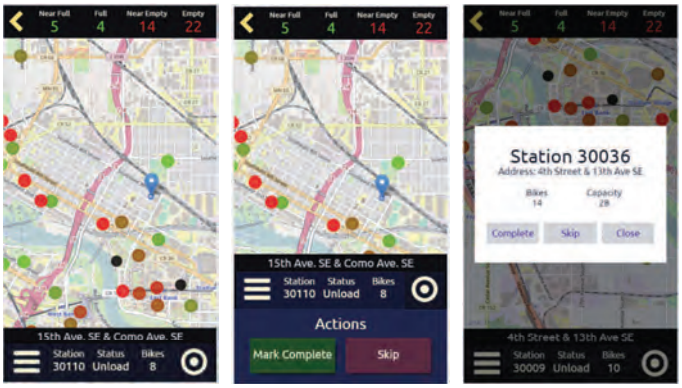
8 IMPLEMENTATION AND FIELD TEST

BRAVO is designed as a webapp due to the benefit of being system-agnostic. BRAVO can be deployed in a traditional desktop web browser as well as on iOS or Android devices as a hybrid app that acts the same as a standard app without any significant changes.

BRAVO is written using javascript. We use 'Node.js' to build the back-end server that serves the front-end app through an https connection. The back-end server sets up secure sockets layer (SSL) connection for accessing operators' location information in real-time. In addition, the back-end server is responsible for data processing, e.g., computing the rebalancing route and rebalancing amount.

On the front-end, the 'Leaflets' web-mapping library is used to develop the user interface (UI). The request functions in the app make 'GET' requests to the API URLs (i.e., the General Bikeshare Feed Specification (GBFS) feed) provided by NiceRide bikeshare system in every 15s to guarantee a timely access of station status data. Note that in BRAVO's working environment, auto mobile WiFi and car charger are provided, network usage and energy consumption are not considered in current version of BRAVO.

In our field tests, BRAVO is installed on Android 5.0.2 on BLU Vivo Air LTE smart phone. The application file has a size of 396 KB[1] and takes 424 KB storage on the phone after installed. The back-end server has a file-size of 1.93 MB and takes 2.66 MB on disk.



(a) Default Screen (b) New Footer (c) New Overlay

Fig. 18. The User Interface (UI) of BRAVO.

The user interface (UI) of BRAVO is friendly. Figure 18(a) shows the default screen of BRAVO. As we can see from the figure, BRAVO features the map front-and-center and indicates the location of the operator. The

[1]The file size is very small due to it being a hybrid app which leverages the default browser on the phone to do the bulk of the work.

stations are marked as circles in the map with the color from green (full status) to red (empty status). The full/empty stations's icon is designed to be sparkling to attract attentions. At the top of the screen, BRAVO shows operators' most concerned information, i.e., the number of (nearly) full and (nearly) empty stations. The less-than-sign button at the top right corner is designed to retract the bar. The expandable bar on the bottom provides information about the operator's current task. In this bar, BRAVO shows the next station (including both station id and station address) the operator should go as well as the amount of bikes should be loaded/unloaded. BRAVO updates the rebalancing task based on operator's location as well as the station status in real time. In other words, the system automatically judges whether the operator finishes the task at the targeted station based station status and prompts the operator the next task if the previous one is done. Overall, our BRAVO app rarely needs interaction and otherwise acts mostly as a guide.

The bar at the bottom of the screen is expandable. The operator can click the hamburger button at the left bottom to manually mark a completed task or skip a task, as shown in Figure 18(b). This design is meaningful under certain situations (e.g., data update delay). The UI is scaled so that it can be easily interacted with on-the-go. The circled dot button at the bottom right corner is designed to enlarge the map. In addition, the operators can check any stations' information by click the station icon. BRAVO will show the number of bikes and docks in this station, as shown in Figure 18(c).

We have tested the BRAVO design in our local NiceRide bikeshare system. The field test results show that our BRAVO system friendly guides multiple rebalancing trucks (i.e., two trucks with one in Minneapolis District and one in Saint Paul District) simultaneously in real time. The field test demo can be found in the YouTube link [2]. In the field test, we are unable to run both Oobrien (previously used bike rebalancing design in NiceRide bikeshare system) and BRAVO bike rebalancing designs. In addition, it is difficult to run our BRAVO system on a real-world bikeshare platform for a long time due to system maintenance issues. In the next section, we evaluate the performance of BRAVO with long-term data driven simulation based on data sets from three real-world bike sharing systems.

9 LONG-TERM DATA DRIVEN EVALUATION

This section reports data-driven evaluation results based on 24-month data sets (where we can run different rebalancing designs under the same user demand) from three bike sharing systems. We compare our design with current bike rebalancing designs in three aspects:

(1) **Station Visit:** the total station visit times by the rebalancing truck in a day.
(2) **Rebalancing Amount:** the total number of loading/unloading bikes in a day.
(3) **Rebalancing Time:** the total time of multiple rebalancing operators (trucks) spending on rebalancing and transportation in one day.

	Capital	Hangzhou	Local
Stations/Bikes	~430/~3700	~3400/~84000	~190/~1700
Collection Periods	10/01/10 - Now	01/01/13 – Now	06/10/10 - Now
Data Size	3.3GB	5.4GB	0.2GB
Record Number	2.8×10^8	1.1×10^8	2.2×10^6

Fig. 19. The basic information of data sets

9.1 Data Sets

The data sets are collected from three bike sharing systems, i.e., (i) Capital bikeshare system in Washington, D.C., U.S., which has more than 3,700 bikes and 400 stations, (ii) Hangzhou bikeshare system in Hangzhou, Zhejiang, China, which has more than 84,000 bikes and 3,400 stations, and (iii) NiceRide bikeshare system. The basic information of data sets from the three systems is shown in Figure 19.

User Trip Data Format		Station Status Data Format	
Trip Duration	Start & End Time	Time Stamp	Station ID
Start & End Station	Bike ID	Station Name	Lat. & Long.
User ID	Member Type	No. of Bikes	No. of Docks
Operator Rebalancing Data Format		**Meteorology Data Format**	
Rebalancing Time	Station ID	Time Stamp	Temperature
Operator ID	Bike ID	Humidity	Wind Speed
Rebalancing Type: Loading/Unloading		Visibility	Weather

Fig. 20. Detailed data format

For Capital bike sharing system, operators' rebalancing data are not released. For Hangzhou and NiceRide bike sharing systems, we collect all the four data sets, i.e., user-trip data, station status data, operators' rebalancing data and meteorology data. The detailed format of the four data sets is shown in Figure 20.

9.2 Evaluation Methodology

Since the data sets from the Capital bikeshare system and the Hangzhou bikeshare system are different, the evaluation methodology is slightly different as well.

• **Capital bikeshare system data set:** The data sets from the Capital bikeshare system do not include the operators' rebalancing data. We exploit the station visits and rebalancing amount from the user trip and station status data. The station status data are recording in every one minute. We thus track the station status in every one minute. If the station status changes, we check the user trip data to see whether there are a borrow/return activity from the user. If there are no user activities, we consider that the station status change is caused by the operator. With this method, we can capture the rebalancing operators' station visits and rebalancing amounts from the data sets. The rebalancing time can not be captured from the current data sets since there are multiple trucks in the system and we can not exact recover their trajectories. In the following evaluation, we compare our design with the Capital bike rebalancing scheme on station visits and rebalancing amounts only.

• **Hangzhou&NiceRide bikeshare system data sets:** Both Hangzhou and NiceRide bikeshare systems record the rebalancing operators' data which include the rebalancing trucks' IDs, the visited stations' IDs, the rebalancing amount, as well as the rebalancing time stamp. We thus can obtain the three performance metrics – station visits, rebalancing amount, and rebalancing time, directly from this data set. Due to privacy reasons, we are allowed to select one year of the four data sets in Xiasha district as a sample for evaluation purposes. There are 184 stations in this district.

For our design, we first capture and predict the user demand using the user trip and meteorology data with the method introduced in Section 5. In the user trip data, as shown in figure 20, it records user ID, user selected bike ID, start and end time, and start and end station. The evaluation thus considers the usage bias caused by users and truly reflects the real-world user demand. Combining the demand information and the station status data, we compute the rebalancing range for each station using the method in Section 6. Finally, based on the rebalancing range in each station, we compute the targeted station and the rebalancing amount with the algorithm introduced in Section 7.

In the following evaluation, the performance metrics of compared designs are obtained under the same user demand, which is calculated from the user trip data using the method introduced in Section 5.1. We compare our system (labeled with "BRAVO") with Oobrien [4], i.e., the rebalancing scheme used in NiceRide bike sharing system (labeled with "Oobrien"). We also compare BRAVO with the bike rebalancing schemes used in Capital bikeshare system (labeled with "CapitalSys") and Hangzhou bikeshare system (labeled with "HzSys") on station visit, rebalancing amount, and rebalancing time.

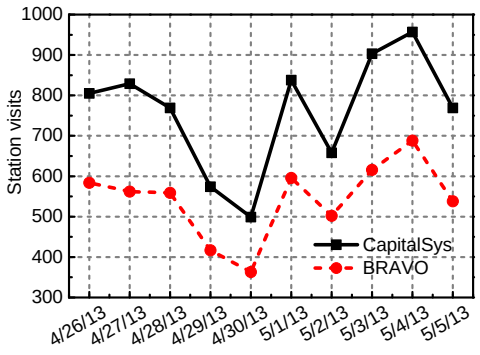


Fig. 21. Station visits in CapitalSys

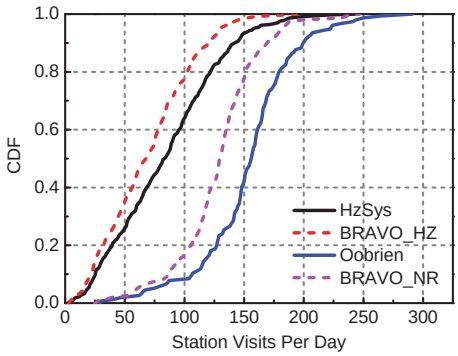


Fig. 22. Station visits in HzSys & NiceRide

9.3 Station Visits

In this subsection, we evaluate BRAVO via station visit in three bike sharing systems. Figure 21 shows the number of station visits from Capital bike sharing system from 4/26/13 to 5/5/13. On average, the station visit of Capital bike chairing system is 760 per day while BRAVO is 543, which introduces a 28% reduction. Figure 22 plots the CDF of station visits per day in Hangzhou (in Xiasha district) and NiceRide bike sharing systems. From the figure, we can see that the number of station visits in these two bikeshare systems changes dramatically, i.e., (i) from 3 to 243 station visits per day in Hangzhou bikeshare system and (ii) from 29 to 291 station visits per day in NiceRide bikeshare system. This is because that the number of station visit is decided by the number of full/empty stations. And station status is changed because of users' borrow/return activity which is significantly impacted by ride comfort factors such as weather condition. On average, operators visit about 154 times per day to rebalancing bikes among stations in NiceRide bike sharing system while BRAVO (denoted as "*BRAVO_NR*" in Figure 22) reduces the station visits to 130, saving 16% station visits.

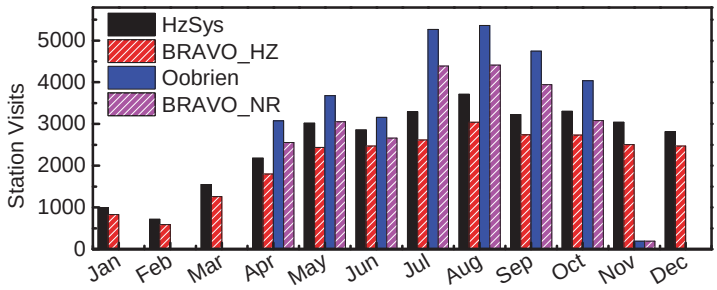


Fig. 23. Long-term evaluation: station visits per month in Hangzhou and NiceRide bikeshare systems.

Figure 23 reports the long-term experimental results of station visit times per month in Hangzhou and NiceRide bikeshare systems. From the figure, we can clearly see that August is the busiest month in these two systems, which has 3,712 and 5,362 station visits per month respectively. In NiceRide bikeshare system, there are very few station visits on November and no station visits on January, February, March, and December. That's because that NiceRide bikeshare system only opens from April 1st to November 1st due to cold weather. In both systems, about 20% reductions on station visits are observed over the whole year. The reduction of station visits in these bike sharing systems is mainly because that our design avoids inappropriate and unnecessary rebalancing operations.

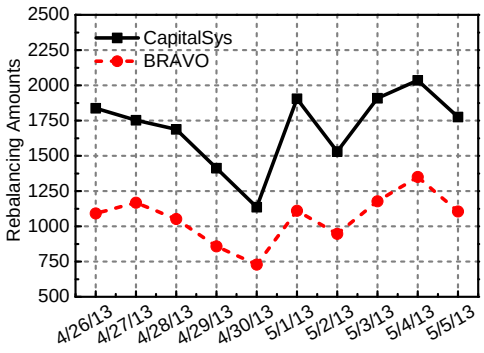


Fig. 24. Rebalancing amount in CapitalSys

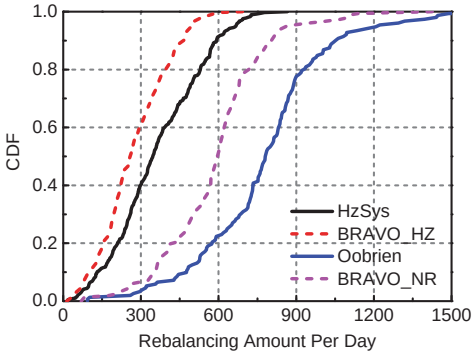


Fig. 25. Rebalancing amount in HzSys & NiceRide

9.4 Rebalancing Amounts

In this subsection, we evaluate the performance of our design in terms of rebalancing amounts. The results from Capital bike sharing system are shown in Figure 24 while Figure 25 plots the rebalancing amounts from Hangzhou (in Xiasha district) and NiceRide bike sharing systems. In the Capital bikeshare system, as shown in Figure 24, the rebalancing amount of our design is about 62% Capital bikeshare systems' rebalancing amount. Figure 26 plots one-year's rebalancing amounts of the Hangzhou and NiceRide bike rebalancing systems and our design. As we can see, the rebalancing amount of original Hangzhou bikeshare system ranges from 4,959 to 13,845 per month and the average rebalancing amount is 11,034. When the BRAVO design is adopted, the average rebalancing amount reduces to 8,233. The result shows that our design reduces 25.4% of the rebalancing amount in Hangzhou bikeshare system.

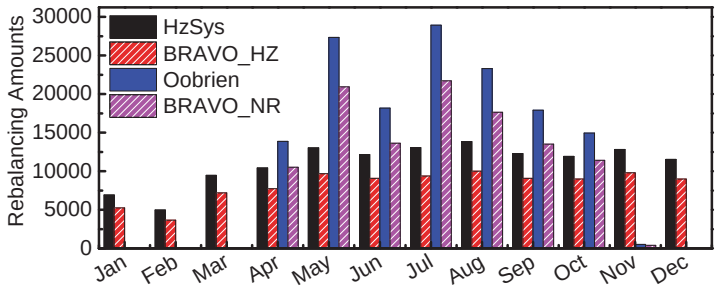


Fig. 26. Long-term evaluation: rebalancing amount per month in Hangzhou and NiceRide bikeshare systems.

The improvement of our design comes from two aspects. First, the precise demand analysis in our design help us clearly capture the rebalancing amount. Second, our design avoids unnecessary rebalancing operations, which greatly reduces the rebalancing amounts.

9.5 Rebalancing Time

In this subsection, we compare the rebalancing time of our design with rebalancing systems used in Hangzhou and NiceRide bikeshare. Figure 27 plots the twelve-month rebalancing time with and without BRAVO in Hangzhou bikeshare system. From the figure, we can see that the rebalancing from January to February is relatively low. The rebalancing time starts to increase from March and keeps in a high value since May. On average, the rebalancing

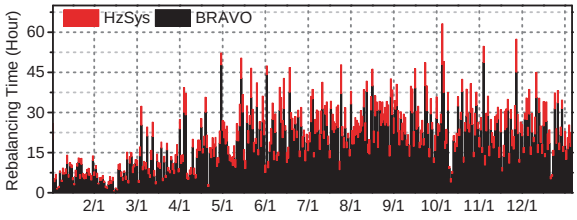


Fig. 27. Rebalancing Time in HzSys

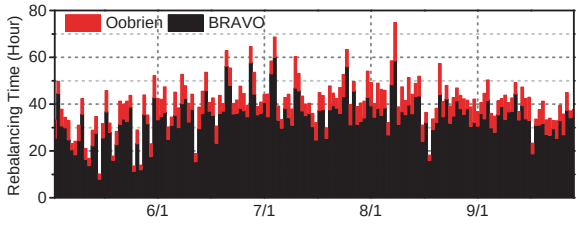


Fig. 28. Rebalancing Time in NiceRide

time of Hangzhou bike rebalancing system is about 22 hours, while our design is around 18 hours. Our design saves about 18.2% of the rebalancing time. Similarly, Figure 28 plots the five-month rebalancing time with BRAVO in NiceRide. In NiceRide bike sharing system, the average rebalancing time is 41 hours (operators in NiceRide rebalancing bikes using one to three trucks in three shifts, i.e., from 7 A.M. to 12 P.M., 1 P.M. to 6 P.M. and 7 P.M. to 12 A.M.) while that of BRAVO is 34 hour, make a 17.1% reduction. These improvements comes from two aspects. On the one hand, our design reduces the station visit times, which reduces the transportation time. On the other hand, our design reduces the rebalancing amounts and saves lots of time on loading/unloading bikes.

10 RELATED WORK

In this section, we review related works on user demand analysis, operator service analysis and rebalancing schedule design separately.

10.1 Demand Analysis

The demand analysis is helpful to (i) understand the user behavior, (ii) predict the future demand, and (iii) provide guidance to managerial decision such as expanding the system or making rebalancing. Most of the demand analysis works for bike sharing system are based on either the station status data set [9, 11, 20] or the user trip data set [7, 12, 19].

Based on the station status data, Froehlich et al. [9] and Kaltenbrunner et al. [11] explore the bike station usage pattern using clustering techniques and predict the number of available bikes/docks for each bike station. Yoon et al. [20] utilize the station status data to predict the available bikes/docks at each station with the autoregressive integrated moving average (ARIMA) model.

Based on the user trip data, Li et al. [12] propose a hierarchical prediction model to predict the number of bikes in each station cluster. They first cluster bike stations into groups using a bipartite clustering algorithm. Then they predict (i) the total number of rent bikes using the learning algorithm – gradient boosting regression tree and (ii) forecast the rent proportion across clusters using a multi-similarity based inference model.

Different from previous works, this paper focus on the prediction of rebalancing demand (i.e., the interval of bikes need to be rebalanced) which is depend on the weather conditions, user behavior, and station status. We thus collect multi-event data sets including the meteorology data, user trip data, and station status data at the same period of time from three existing bike sharing systems. Based on these three data sets, we propose a new model to predict the rebalancing demand interval.

10.2 Service Analysis

The existing works on service analysis mainly focus on the service levels in bike sharing systems, i.e., the service requirements for available bikes/docks. Nair et al. [15] propose a dual-bounded chance constraint to model the service requirements on bikes and docks in each station. Researchers also study the bikeshare station placement problem by exploiting bike trip demand from heterogeneous urban data [6].

This paper focuses on analyzing the service issue in current bike sharing system. We extract the exact rebalancing activity, i.e., the amount of load/inload bikes in each station in every one minute, from the user trip data and station status data. Based on this information, we find out the inefficiencies in current rebalancing service which motivate us to propose a novel bike rebalancing scheme to address these inefficiencies.

10.3 Rebalancing Schedule

There are two ways to rebalance the bikes among stations. The first way is to provide users incentives to rebalance the bikes. The second way is to dispatch operators to rebalance bikes using trucks. Currently, existing bike sharing systems use the second way, i.e., rebalancing bikes using trucks. Therefore, the rebalancing route problem has received lots of attention, and several models have been proposed to address this problem. Benchimol et al. [14] formulate bike rebalancing route problem as variants of the traveling salesman problem (TSP) and propose approximation algorithms to minimize the cost in the rebalancing route. Liu et al. [13] present mixed-integer programming (MIP) models to build rebalancing route to minimize the rebalancing cost.

Instead of proposing another rebalancing route formulation, this paper focuses on the rebalancing amount calculation problem which is motivated by the inefficiencies found from the existing bike sharing system through extensive data analysis. In detail, the proposed rebalancing amount formulation minimizes the total rebalancing cost while guaranteeing stations' safe rebalancing ranges from data analysis. An optimal solution using dynamic programming is proposed to this novel problem.

11 CONCLUSION

In this work, we propose a practical bike rebalancing app to reduce the total rebalancing cost for bike sharing systems. Our work provides novel contributions on (i) user demand analysis, (ii) rebalancing service analysis, and (iii) rebalancing amount design. Specifically, in user demand analysis, instead of predicting the exact demand amount in a station, we advertise to predict the safe rebalancing range. In rebalancing service analysis, we figure out the inefficiencies in current rebalancing operations as well as their root causes. To address these inefficiencies, we provide a novel rebalancing amount formulation based on the safe rebalancing range analysis as well as the practical requirements from the rebalancing operators. Finally, an optimal algorithm is proposed to this rebalancing amount problem.

REFERENCES

[1] 2014. Capital Bikeshare Member Survey Report Executive Summary.

[2] 2017. BRAVO Demo. https://youtu.be/fMro92f6cVk. Accessed: 2017-05-23.

[3] 2017. General Bikeshare Feed Specification. https://github.com/NABSA/gbfs. Accessed: 2017-04-10.

[4] 2017. Oobrien System. http://bikes.oobrien.com. Accessed: 2017-04-10.

[5] 2017. Released General Bikeshare Feed Specification (GBFS) Links. https://www.motivateco.com/use-our-data. Accessed: 2017-04-10.

[6] Longbiao Chen, Daqing Zhang, Gang Pan, Xiaojuan Ma, Dingqi Yang, Kostadin Kushlev, Wangsheng Zhang, and Shijian Li. 2015. Bike Sharing Station Placement Leveraging Heterogeneous Urban Open Data. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*. 571–575.

[7] Longbiao Chen, Daqing Zhang, Leye Wang, Dingqi Yang, Xiaojuan Ma, Shijian Li, Zhaohui Wu, Gang Pan, Thi-Mai-Trang Nguyen, and Jérémie Jakubowicz. 2016. Dynamic Cluster-based Over-demand Prediction in Bike Sharing Systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*. 841–852.

[8] P DeMaio. 2009. Bike-sharing: History, Impacts, Models of Provision, and Future. *Journal of Public Transportation* 12, 4 (2009).

[9] Jon Froehlich, Joachim Neumann, and Nuria Oliver. 2009. Sensing and Predicting the Pulse of the City Through Shared Bicycling. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. 1420–1426.

[10] Nicolas Gast, Guillaume Massonnet, Daniel Reijsbergen, and Mirco Tribastone. 2015. Probabilistic forecasts of bike-sharing systems for journey planning. In *ACM CIKM*.

[11] Andreas Kaltenbrunner, Rodrigo Meza, Jens Grivolla, Joan Codina, and Rafael Banchs. 2010. Urban Cycles and Mobility Patterns: Exploring and Predicting Trends in a Bicycle-based Public Transport System. *Pervasive Mob. Comput.* 6, 4 (2010), 455–466.

[12] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. 2015. Traffic Prediction in a Bike-sharing System. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 33:1–33:10.
[13] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. 2016. Rebalancing Bike Sharing Systems: A Multi-source Data Smart Optimization. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1005–1014.
[14] Benchimol M., Benchimol P., Chappert B., Taille A. D. L., Laroche F., Meunier F., and Robinet L. 2011. Balancing a dynamic public bike-sharing system. *RAIRO – Operations Research* 45, 1 (2011), 37–61.
[15] R. Naira, R. Hampshire E. Miller-Hooks, and A. Busic. 2013. Large-Scale Vehicle Sharing Systems: Analysis of Velib. *International Journal of Sustainable Transportation* 7, 1 (2013), 85–106.
[16] Tal Raviv, Tzur Michal, and I. Forma. 2013. Static Repositioning in a Bike-Sharing System: Models and Solution Approaches. *EURO Journal on Transportation and Logistics* 2, 3 (2013), 187–229.
[17] Susan A. Shaheen, Elliot W. Martin, Nelson D. Chan, Adam P. Cohen, and Mike Pogodzinski. 2014. Public Bikesharing In North America During A Period Of Rapid Expansion: Understanding Business Models, Industry Trends and User Impacts. *Mineta Transportation Institute* (2014).
[18] P. Vogel and D. Mattfeld. 2011. Strategic and operational planning of bike-sharing systems by data mining- a case study. In *Proceedings of the Second International Conference on Computational Logistics*. 127–141.
[19] Zidong Yang, Ji Hu, Yuanchao Shu, Peng Cheng, Jiming Chen, and Thomas Moscibroda. 2016. Mobility Modeling and Prediction in Bike-Sharing Systems. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. 165–178.
[20] Ji Won Yoon, Fabio Pinelli, and Francesco Calabrese. 2012. Cityride: A Predictive Bike Sharing Journey Advisor. In *Proceedings of the 13th International Conference on Mobile Data Management*. 306–311.

Received August 2017; revised November 2017; accepted January 2018