ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230447

Dynamic Workload-Aware Bike Rebalancing for Bike-Sharing Systems

Yuan Luo[a;*]

[a]School of Science and Engineering, Shenzhen Institute of Artificial Intelligence and Robotics for Society, The Chinese University of Hong Kong, Shenzhen 518172, China and Guangdong Provincial Key Laboratory of Aerospace Communication and Networking Technology
ORCiD ID: Yuan Luo https://orcid.org/0000-0001-5129-0130

Abstract. Bike Sharing Systems (BSSs) offer a flexible and sustainable transport option that has gained popularity in urban areas globally. However, as users move bikes according to their own needs, imbalanced bike distribution becomes a significant challenge for BSS operators. To address this problem, we propose a Workload Awareness (WA) approach that considers the rebalancing workload of BSS sub-networks and congestion issues when repositioning bikes dynamically. Our algorithm, WA, identifies sub-networks in a BSS and ensures a similar rebalancing load for each sub-network. Our mixed integer nonlinear programming (MINLP) model then finds a repositioning policy for each sub-network, taking into account operator capacity, bike and dock information, and minimizing total losses due to bike shortages and dock congestion. Our experiments on the Ningbo City Bike system demonstrate that our approach outperforms state-of-the-art methods by reducing the loss of the system by up to 60% and significantly reducing computational time by up to 36%.

1 Introduction

Bike Sharing System (BSS) is a shared transport service where users can rent bikes for short journeys. Traditionally, a BSS has stations with docks set up across a city and stocked with a number of bikes. Users can pick up and return bikes at any station according to their real-time demand without booking in advance. As BSSs can provide a green and flexible travel option to individuals and mitigates the problem of carbon emission and traffic congestion at the social level, they have become increasingly popular worldwide [1]. Nowadays, there are 1914 BSSs with 8,967,122 bikes operating worldwide, with hundreds more in the planning stage [23].

However, a major problem with the flexibility for BSS users is imbalanced bike distribution [15]. The demand of users is self-oriented without a global vision and is asymmetric. In this case, a large number of users' individualistic movements can cause two possible problems: (i) starvation, where supply is less than demand; and (ii) congestion, where supply is more than demand. Such bike supply/demand imbalance can result in a number of consequences such as revenue loss for a BSS company and increased carbon emission. Hence, it is crucial for the owners of BSSs to redistribute the resources and to ensure the system is balanced and functioning effectively [12].

* Corresponding Author. Email: luoyuan@cuhk.edu.cn

To address these problems, researchers have adopted several approaches. A common approach is to deploy heavy vehicles (*e.g.*, trucks) to relocate bikes among stations, either at night or during the day. However, the use of fuels-burning vehicles goes against the concept of greenness and worsens the problem of heavy traffic [13].

As an alternative, researchers have been studying the use of smaller carriers to perform dynamic repositioning tasks in BSSs [9, 18]. Dynamic repositioning involves using trailers, a type of add-on to a bike that can carry a small number (usually 3-5) of bikes, to move bikes throughout the day to fix imbalances during rush hours. With a sufficient fleet size and proper scheduling, small vehicles distributed across a BSS can help relocate bikes to nearby stations and balance the system at a lower environmental cost. However, to simplify the problem, existing works only focus on the starvation problem and neglect the congestion problem. Thus, their proposed method fails to decrease the total loss of a BSS maximally. Furthermore, they do not consider the large scale of bike rebalancing problems. Without considering this commonly occurring issue [10], the designed rebalancing strategy will not be applicable in a real-world BSS which can consist of hundreds of stations.

There are several works devoted to decomposing a large BSS into smaller sub-networks for dynamic bike repositioning (see Section 2). However, these methods only consider the geographical information of stations and ignore the information of historical trip records. Furthermore, they do not consider rebalancing workload in each sub-network. As the number of operators who use smaller carriers to perform dynamic repositioning tasks for each sub-network is approximately the same, making sub-networks have an evenly rebalancing workload is crucial: sub-networks with heavy rebalancing workload may not have sufficient operators to finish the repositioning tasks within a given planning period, while sub-networks with light rebalancing workload make operators to be idle, causing a waste in the human resources.

Against this background, this paper proposes a mixed integer nonlinear programming (MINLP) with Workload Awareness (WA) approach. We first propose WA that groups stations in a large-scale BSS into different sub-networks based on the stations' geographical information and bike flows and ensures that each sub-network has a similar rebalancing workload. Then we allocate operators for conducting rebalancing in each sub-network. Considering the capacity and location of operators and tracing the docks and bikes information, we design a MINLP model to minimize the total loss.

This paper advances the state-of-art method for dynamic repositioning using small vehicles in the following ways. First, our WA algorithm is the only one that allows each sub-network to have an evenly rebalancing workload by incorporating both geographical information of stations and bike flows. Such a joint consideration can ensure that each sub-network has a similar rebalancing workload, leading to better loss-minimizing performance and higher computational efficiency. Second, we are the first to design the MINLP model that considers losses caused by lack of docks (congestion) and losses caused by lack of bikes (starvation). Finally, we design a simulator which is built on the real-world data set from Ningbo City Bike system to evaluate the effectiveness of rebalancing approaches. Our empirical results show that our approach can significantly decrease the loss of the system by up to 60% compared to the current state of the art. Moreover, compared to the state of the art algorithms, our proposed method can reduce the program's running time by up to 36%.

2 Related Work

Acknowledging the many benefits of BSSs, various researches are carried out to cover the problem of uneven resource distribution and optimally utilize its advantages. In this section, we summarize and give examples of the main strands of research that are closely related to our work.

Methods of bike rebalancing fall into one of two main categories: static repositioning and dynamic repositioning. In a static repositioning scheme, bike positioning is carried out during the "off" hours of a BSS, *e.g.*, at night, to achieve a strategically determined stock level for the next operation round. Costa Affonso *et al.*'s work [3] is one of the latest works in this strand. Usually, static repositioning employs a smaller number of heavy vehicles (*e.g.*, trucks) with larger capacity and greater travel range [21]. However, the effect of static repositioning is limited and can fail to match supply and demand quickly as the bikes start to move when the BSS operates during the day [14].

On the contrary, dynamic repositioning schedules for bike repositioning happen throughout the day when the BSS is in operation and considers the bike flows at different times of the day. Dynamic repositioning using trucks has been studied [20, 6, 14]. The use of a large number of small vehicles has also been studied a lot due to environmental concerns regarding heavy vehicles. The best example of this line of work is by Ghosh and Varakantham [9], who consider the problem of dynamic scheduling using small bike trailers. However, they only consider the loss caused by starvation but not the loss caused by congestion. Thus, their proposed method fails to decrease the total loss of a BSS significantly. Furthermore, they do not consider the large-scale bike rebalancing problem. Hence, the running time of this approach becomes excessive for a real-world BSS.

Concerning the time-effectiveness issue, clustering in bike rebalancing has attracted considerable attention. For example, Raviv *et al.* [22] cluster the stations based on the expected number of empty stations after a certain amount of time. Forma *et al.* [5] propose a clustering method based on station locations as well as their inventory capacity. Ghosh *et al.* [10] apply the k-means clustering based on station locations. Li *et al.* [16] use reinforcement learning to rebalance the system after clustering. The best example of this line of work is by Wang *et al.* [24], who exploit bike flows between stations in addition to the geographical information of stations. Their solution is called MFUA. However, they do not consider rebalancing workload in each sub-network and ignore the congestion problem in a rebalancing problem. Thus, they cannot help to reduce the total loss in a large-scale BSS most effectively.

3 System Model

In practical bike rebalancing operations, when the coverage of bike stations is broad, and the user demands are high, we need to decompose the large BSS into smaller sub-networks for real-time operations [10, 14, 24]. We consider a set $\mathcal{C} = \{1, 2, \ldots, C\}$ of C sub-networks. The objective of a BSS decomposition is to make each sub-network have three properties, *i.e.*, inner-balance in each sub-network, inter-independence between sub-networks and similar rebalancing workload [24]. The inner-balance property means that the total bike rent and return demands in a sub-network should be almost equal. The inter-independence property means no frequent transitions between any two sub-networks. The similar rebalancing workload means that sub-networks have an evenly rebalancing workload. Then we allocate operators for each sub-network to conduct bike rebalancing operations. The number of operators for each sub-network is approximately the same in a practical BSS system.

Then we extend the generic model of dynamic rebalancing and routing problem (DRRP) introduced by Ghosh *et al.* [10], which is common in the literature (e.g., [17, 7, 14]), for defining our problem. For each sub-network, the problem can be represented using the following tuple:

$$\langle \mathcal{S}_c, \mathcal{V}_c, \mathbf{C}^{\#}, \mathbf{C}^{*}, \mathbf{d}_c^{\#,0}, \{\sigma_{v,s}^0\}, \mathbf{H}_c, \mathbf{F}_c^t \rangle$$

- $\mathcal{S}_c$ represents the set of stations belonging to sub-network c. Each station $s \in \mathcal{S}_c$ has a capacity (number of docks) $C_s^{\#}$.
- $\mathcal{V}_c$ represents the set of operators allocated to sub-network c. Each operator $v \in \mathcal{V}_c$ has a capacity (maximum number of bikes carried) C_v^*.
- $d_s^{\#,0}$ specifies the initial number of bikes at station $s \in \mathcal{S}_c$ at the start of the day (planning period 0). Together, $\mathbf{d}_c^{\#,0}$ gives the initial distribution of bikes across stations belonging to sub-network c [1].
- $\sigma_{v,s}^0 = 1$ if operator $v \in \mathcal{V}_c$ is at station $s \in \mathcal{S}_c$ at the start of the day. Together, the set $\{\sigma_{v,s}^0\}$ gives the initial locations of operators.
- $\mathbf{H}_c$ is a $S_c \times S_c$ matrix where $H_{s,s'}$ denotes the distance between station $s \in \mathcal{S}_c$ and $s' \in \mathcal{S}_c$ that are belonged to sub-network c.
- $\mathbf{F}_c^t$ contains the demand flow (number of bikes demanded) in sub-network c across all K training scenarios, which are used for training purposes. $F_{s,s'}^{k,t}$ denotes the demand flow from station $s \in \mathcal{S}_c$ to $s' \in \mathcal{S}_c$ in training scenario k in the planning period t.

To dynamically schedule for bike relocation throughout the day, we divide the operating time of a BSS into a series of discrete planning periods. As the locations of bikes and operators change after each planning period, we update the variables regarding distributions and let $\mathbf{d}_c^{\#,t}$ and $\{\sigma_{v,s}^t\}, v \in \mathcal{V}_c, s \in \mathcal{S}_c$ denote the distribution of bikes and the locations of operators belonging to sub-network c at the start of planning period t.

As is common in the literature (e.g., [9, 7, 18]), we make the following assumptions for the ease of representation and evaluation[2]: (i) we assume that users and operators in planning period t always return their bikes at the beginning of planning period $t+1$; (ii) users are impatient and leave the system if they encounter an empty station.

[1] Throughout this paper, we use bold notations for vectors.

[2] These assumptions can easily be relaxed with minor modifications to our methods.

Algorithm 1 WA Algorithm

Input: $C, \mathcal{S}, \mathbf{C}^{\#}, \mathbf{d}^{\#,t}, \mathbf{H}, \mathbf{F}^t, K$;
Initialize: $\mathcal{B}_c \leftarrow \emptyset$ **for** $c \in \mathcal{C}$, $\mathcal{O} \leftarrow \emptyset$;
Compute $LB_s^k, LD_s^k, \forall s \in \mathcal{S}, k \in K$;
$AL_g =$ **ComputeAL**$(\mathcal{S})$;
Construct sub-networks based on community detection algorithm;
Select the centroides l_c **for sub-network** c;
for $c \in \mathcal{C}$ **do**
  $AL_c =$ **ComputeAL**$(\mathcal{B}_c)$;
  if $AL_c > AL_g$ **then**
    Sort $s \in \mathcal{B}_c$ **in descending order of** H_{s,l_c} **to get** $\mathcal{B}_c^{\downarrow}$;
    for $s \in \mathcal{B}_c^{\downarrow}$ **do**
      while $AL_c > AL_g$ **do**
        if ComputeAL $(\mathcal{B}_c \setminus \{s\}) \leq AL_g$ **and** $s \neq l_c$ **then**
          $\mathcal{O} \leftarrow \mathcal{O} \bigcup \{s\}$;
          $\mathcal{B}_c \leftarrow \mathcal{B}_c \setminus \{s\}$;
          $AL_c =$ **ComputeAL**$(\mathcal{B}_c)$;
        end if
      end while
    end for
  end if
end for
for $s \in \mathcal{O}$ **do**
  $\mathcal{C}^{\text{candidate}} \leftarrow \emptyset$;
  Sort $c \in \mathcal{C}$ **in ascending order of** AL_c **to get** $\mathcal{C}^{\uparrow}$;
  for $c \in \mathcal{C}^{\uparrow}$ **do**
    if ComputeAL$(\mathcal{B}_c \cup \{s\}) \leq AL_g$ **then**
      $\mathcal{C}^{\text{candidate}} \leftarrow \cup\{c\}$;
    end if
  end for
  Sort $c \in \mathcal{C}^{\text{candidate}}$ **in ascending order of** H_{s,l_c} **to get** $\mathcal{C}^{\uparrow,n}$ **and** c_1**, where** c_1 **is the first cluster index in** $\mathcal{C}^{\uparrow,n}$;
  $\mathcal{O} \leftarrow \mathcal{O} \setminus \{s\}$;
  $\mathcal{B}_{c_1} \leftarrow \mathcal{B}_{c_1} \cup \{s\}$;
  $AL_{c_1} =$ **ComputeAL**$(\mathcal{B}_{c_1})$;
end for
for $s \in \mathcal{O}$ **do**
  $\mathcal{O} \leftarrow \mathcal{O} \setminus \{s\}$;
  $c = \arg\min_{c' \in \mathcal{C}} AL_{c'}$;
  $\mathcal{B}_c \leftarrow \mathcal{B}_c \cup \{s\}$;
  $AL_c =$ **ComputeAL**$(\mathcal{B}_c)$;
end for
return $\mathcal{B}_c$ **for** $c \in \mathcal{C}$;

Users return their bikes to the nearest station if the destination station is full; and (iii) the events at each planning period follow a particular sequence. Operators first carry out the repositioning tasks, then arrival users pick up their bikes and lastly, users return their bikes at the destination station.

4 Methodology

In this section, we first describe the details of MINLP with WA approach. Then, we illustrate the system simulator that simulates the dynamic process of MINLP with WA approach for a BSS.

4.1 WA Algorithm

The WA algorithm aims at decomposing a large BSS into small sub-networks so that the complexity of solving the bike rebalancing problem can be reduced. In more detail, WA group stations in a large-scale BSS into different sub-networks, within which we can schedule and perform bike repositioning. Algorithm 1 presents WA algorithm. The main idea of this algorithm is to consider the geographical locations of stations, traffic flow and rebalancing workload while generating sub-networks. More specifically, WA generates sub-networks based on distance and traffic flow first, then adjust the sub-networks to eliminate any highly rebalance-demanding sub-networks.

We use the station's original total loss (*i.e.*, the total loss that would occur at that station if no rebalancing is performed) to measure the degree of rebalancing needed at a station quantitatively. The higher the original total loss at a station is, the more rebalancing should be performed around that station to reduce such loss.

To evaluate the amount of rebalancing needed by a set of stations, we use the notion of Average Loss (AL) over stations of non-zero loss. For a set of stations $\mathcal{P}$, we compute AL as

$$\text{ComputeAL}(\mathcal{P}) = \frac{1}{K}\sum_{k\in\mathcal{K}}\left(\frac{1}{\max\left(|\mathcal{P}^{+,k}|,1\right)}\sum_{s\in\mathcal{P}^{+,k}}\left(LB_s^k + LD_s^k\right)\right), \quad (1)$$

where $\mathcal{P}^{+,k}$ represents the set of stations in $\mathcal{P}$ with a non-zero total loss in scenario k, LB_s^k denotes loss caused by a lack of bikes, which is incurred when a user can't find an available bike to rent at station s in scenario k and LD_s^k is the loss caused by a lack of docks, which is incurred when a user can't find an available dock to return their bike at station s in scenario k.

WA generates sub-networks in planning period t in the following steps.

- **Initialize sub-network set and the outlier set**: We first initialize the set of stations ($\mathcal{B}_c$) for each sub-network c and the outlier set $\mathcal{O}$, which is a temporary storing place for outlier stations when we later adjust sub-networks.
- **Compute original total loss from demand flow**: In step 3, We compute the original total loss at each station in each training scenario from the demand flow $\mathbf{F}^t$.
- **Compute the global AL as a standard of AL**: We then calculate the AL over all stations in step 4 and denote it as AL_g. We will use AL_g as a threshold to adjust the clustering result.
- **Construct sub-networks based on distance and traffic flow**: We adopt a betweenness based community detection algorithm [19] to construct sub-networks. Each sub-network is almost inner-balanced and inter-independent with others sub-networks.
- **Select each sub-network's central station**: We select the centroides l_c for sub-network c based on K-means algorithm [4].
- **Compute and cap AL at AL_g by removing a single outlier station**: We then adjust sub-networks based on AL. For each cluster c, we first compute its AL, denoted as AL_c, in step 8. Then we compare AL_c with the threshold AL_g. If AL_c exceeds the threshold, we look at stations in c in descending order of distance to the sub-network's centre l_c. For each station, if removing it from the sub-network reduces this sub-network's AL_c to below AL_g, we do so and temporarily place the station in the outlier set.
- **Try assigning stations in the outlier set to sub-networks without breaking the AL_g bound**: We then need to assign outlier stations to different sub-networks. As the purpose is to average each sub-network's AL, for each outlier station s, we attempt all sub-networks in ascending order of AL_c. We first find a set of sub-networks $\mathcal{C}^{\text{candidate}}$ in which a sub-network can accept the outlier station s without having its AL_c exceed the threshold AL_g. Once we find this sub-network set, we assign station s to its nearest sub-network in the set $\mathcal{C}^{\text{candidate}}$.

- **Assign remaining stations in the outlier set to sub-networks of relatively low** AL_c: Note that there could be stations that will result in large AL within every cluster. As we need to assign every station to a sub-network, in steps 35-40, we assign each station to the sub-network with the lowest current AL_c, so the excess of resulted AL_c over AL_g is as small as possible.

4.2 MINLP Model

In this section, we formulate the objective and constraints of our MINLP model to generate a feasible repositioning policy for each sub-network. At the start of rebalancing time slot t, the initial bike distribution $\mathbf{d}_c^{\#,t}$ and the demand flow $\mathbf{F}_c^t$ for sub-network c in all K training scenarios are given as the input to the MINLP model. The distribution of operators working in sub-network c (*i.e.*, $\{\sigma_{v,s}^t\}$, $v \in \mathcal{V}_c, s \in \mathcal{S}_c$) is also the input to our MINLP model.

We define the decision variables in MINLP as follows:

- $y_{s,v}^{+,t}$ represents the number of bikes picked up from station $s \in \mathcal{S}_c$ by operator $v \in \mathcal{V}_c$.
- $y_{s,v}^{-,t}$ represents the number of bikes dropped off at station $s \in \mathcal{S}_c$ by operator $v \in \mathcal{V}_c$.
- $b_{s,v}^{+,t} = 1$ if operator $v \in \mathcal{V}_c$ picks up bikes from station $s \in \mathcal{S}_c$.
- $b_{s,v}^{-,t} = 1$ if operator $v \in \mathcal{V}_c$ drops off bikes at station $s \in \mathcal{S}_c$.

Then we present the formulation of the MINLP model for sub-network $c \in \mathcal{C}$ in Table 1. The physical meanings of the objective and constraints in the optimization problem are explained as follows:

- **Objective**: Our objective is to minimize the total number of expected loss bikes and loss docks over K training demand scenarios in sub-network $c \in \mathcal{C}$ in planning period t [3]. As each scenario has equal probability [9], we define the expected total loss over all K training scenarios as $\sum_{s\in\mathcal{S}_c, k\in\mathcal{K}} LB_s^{k,t} + LD_s^{k,t}$.
- **Compute loss bikes as the deficiency in the supply of bikes**: Constraint (2) ensures that the number of loss bikes at station $s \in \mathcal{S}_c$ in scenario k is lower bounded by the difference between demand and supply of bikes at station s. The number of bikes presents in a station s after accomplishing the repositioning task is estimated as $d_s^{\#,t} + \sum_{v\in\mathcal{V}_c}\left(y_{s,v}^{-,t} - y_{s,v}^{+,t}\right)$. As we minimize the sum of loss bikes over all the scenarios, these constraints are sufficient to compute the exact number of loss bikes.
- **Compute loss docks as the excess of bikes over station capacity:** Constraint (3) ensures that the number of loss docks at station $s \in \mathcal{S}_c$ in scenario k is lower bounded by the difference between the supply of bikes at station s and its capacity. As we minimize the sum of loss docks over all the scenarios, these constraints are sufficient to compute the exact number of loss docks.
- **Operator's capacity is not exceeded while picking up bikes:** Constraint (4) ensures that the number of bikes picked up by operator $v \in \mathcal{V}_c$ from station $s \in \mathcal{S}_c$ is bounded by the minimum value between the number of bikes available in this station and the operator's capacity.
- **The total number of bikes picked up from a station does not exceed the number of available bikes there:** Constraint (5) ensures that the total number of bikes picked up from a station by all operators is bounded by the number of bikes presents in this station.

[3] For analytical convenience, we do not consider the cost of repositioning. Adding the repositioning cost does not change our main insights, as repositioning is performed by employed operators who are on duty throughout the day and have a fixed pay.

$$\min_{y} \sum_{s\in\mathcal{S}_c, k\in\mathcal{K}} LB_s^{k,t} + LD_s^{k,t} : \tag{2}$$

$$LB_s^k \geq \sum_{s'\in\mathcal{S}_c} F_{s,s'}^{k,t} - \left(d_s^{\#,t} + \sum_{v\in\mathcal{V}_c}\left(y_{s,v}^{-,t} - y_{s,v}^{+,t}\right)\right) \tag{3}$$

$$\forall k, s \in \mathcal{S}_c \tag{4}$$

$$LD_s^k \geq \sum_{s'\in\mathcal{S}_c} F_{s',s}^{k,t} + \left(d_s^{\#,t} + \sum_{v}\left(y_{s,v}^{-,t} - y_{s,v}^{+,t}\right)\right) - C_s^{\#}, \tag{5}$$

$$\forall k, s \in \mathcal{S}_c$$

$$y_{s,v}^{+,t} \leq b_{s,v}^{+,t} \cdot \min\left(d_s^{\#,t}, C_v^*\right),\ \forall s \in \mathcal{S}_c, v \in \mathcal{V}_c \tag{6}$$

$$\sum_{v\in\mathcal{V}_c} y_{s,v}^{+,t} \leq d_s^{\#,t},\ \forall s \in \mathcal{S}_c \tag{7}$$

$$\sum_{v\in\mathcal{V}_c} y_{s,v}^{-,t} \leq C_s^{\#} - d_s^{\#,t},\ \forall s \in \mathcal{S}_c \tag{8}$$

$$y_{s,v}^{-,t} = b_{s,v}^{-,t} \cdot \sum_{s'\in\mathcal{S}_c} y_{s',v}^{+},\ \forall s \in \mathcal{S}_c, v \in \mathcal{V}_c \tag{9}$$

$$\left(b_{s,v}^{+,t} + b_{s',v}^{-} - 1\right) \cdot H_{s,s'} \leq H_{\max},\ \forall s, s' \in \mathcal{S}_c, v \in \mathcal{V}_c \tag{10}$$

$$\sum_{s\in\mathcal{S}_c} b_{s,v}^{+,t} = 1,\ \forall v \in \mathcal{V}_c \tag{11}$$

$$\sum_{s\notin G_v} b_{s,v}^{+,t} = 0,\ \forall v \in \mathcal{V}_c \tag{12}$$

$$\sum_{s\in\mathcal{S}_c} b_{s,v}^{-,t} = 1,\ \forall v \in \mathcal{V}_c \tag{13}$$

$$LB_s^{k,t}, LD_s^{k,t} \geq 0;$$
$$0 \leq y_{s,v}^{+,t}, y_{s,v}^{-,t} \leq C_v^*;$$
$$b_{s,v}^{+,t}, b_{s,v}^{-,t} \in \{0,1\},\ \forall k, s \in \mathcal{S}_c, v \in \mathcal{V}_c \tag{14}$$

Table 1: MINLP

- **Station capacity is not exceeded while dropping off bikes:** Constraint (6) ensures that the total number of dropped off bikes at station $s \in \mathcal{S}_c$ is bounded by the number of available docks at that station.
- **An operator should return the exact number of bikes she/he has picked up:** Constraint (7) enforces that, at the drop-off station (where $b_{s,v}^{-,t} = 1$), the number of bikes dropped off by an operator is exactly the number of bikes she/he has picked up.
- **An operator can only travel for a bounded distance:** We assume there is a maximum distance an operator can travel from the pick-up station to the drop-off station, which we represent as $H_{\max}$. Constraint (8) enforces that the distance travelled by an operator is limited by this threshold value $H_{\max}$.
- **An operator can only pick up bikes from one station:** We assume, in one planning period, an operator can only complete one relocating task. Constraint (9) enforces that an operator can only go to one pick-up station by allowing only one $b_{s,v}^{+,t}, s \in \mathcal{S}_c, v \in \mathcal{V}_c$ to be set to 1.
- **An operator can only pick up bikes from a nearby station:** We assume there is a maximum distance an operator can travel to get to the pick-up station. Therefore, based on the initial location of an operator, there is a set of stations that the operator can pick

up bikes from, which we denote as G_v. Constraint (10) allows an operator to pick up bikes from a nearby station only by forcing all decision variables representing picking up from a station that is not in the near station set to be 0.

- **An operator can only drop off bikes at one station:** Similar to Constraint (9), Constraint (11) enforces that an operator can only return bikes to one station by allowing only one $b^{-,t}_{s,v}, v \in \mathcal{V}_c, s \in \mathcal{S}_c$ to be set to 1.

Note that, Constraint (7) is not linear. However, as $b^{-,t}_{s,v}$ is a binary decision variable, we can use the big-M method [11] for the linearization. We can replace Constraint (7) as follows:

$$y^{-,t}_{s,v} \leq C^*_v \cdot b^{-,t}_{s,v} \qquad \forall s \in \mathcal{S}_c, v \in \mathcal{V}_c \qquad (15)$$

$$y^{-,t}_{s,v} \leq \sum_s y^{+,t}_{s,v} \qquad \forall s \in \mathcal{S}_c, v \in \mathcal{V}_c \qquad (16)$$

$$y^{-,t}_{s,v} \geq \sum_s y^{+,t}_{s,v} - \left(1 - b^{-,t}_{s,v}\right) \cdot C^*_v \qquad \forall s \in \mathcal{S}_c, v \in \mathcal{V}_c \qquad (17)$$

As the operator's capacity is small (*i.e.*, less than 5), such a linearization does not incur too much computational expense [9]. Then we solve the MINLP model using IBM ILOG CPLEX Optimization Studio V12.10 within python code on a 3.2 GHz Intel Core i7 machine.

4.3 System Simulator

We establish a simulator to evaluate the approach's performance on real-world data. We iterate the MINLP model with WA over planning period t to achieve dynamic bike rebalancing. At the beginning of planning period t, we respectively input the distribution of bike $\mathbf{d}^{\#,t}$ and demand flow $\mathbf{F}^t$ to WA and WA algorithm generates sub-networks $\mathcal{C}$ for planning period t. We then allocate operators $\mathcal{V}_c$ for each sub-network c to rebalance bikes among stations belonging to c and find the solution to the MINLP model. The solution of the MINLP model describes the repositioning policy, including the routing of operators and the number of bikes that are picked up and dropped off by operator v at station s in planning period t. Following the repositioning policy, we simulate the reposition, rent and return process in turn in each planning period, and repeat it until the end of planning period. For all sub-networks, the above processes are simulated simultaneously. Algorithm 2 shows the detailed procedure for planning period t.

Algorithm 2 MINLP with WA

```
Input: MINLP, cluster number C
Generate sub-networks with WA;
for cluster c ∈ C do
    Y_c^+, Y_c^-, B_c^+, B_c^- ← Solve MINLP within c;
end for
Y^+, Y^-, B^+, B^- ← Y_c^+, Y_c^-, B_c^+, B_c^- for c ∈ C;
return Y^+, Y^-, B^+, B^-;
```

5 Empirical Evaluation

We conduct an empirical study to evaluate the performance of our approach. This shows how considering both the rebalancing workload of sub-networks in a BSS and the congestion problem can minimize the system's total loss effectively. We compare our method to other state of the art mechanisms that do not include these two actions.

Real-world Dataset

We evaluate the performance of our approach with respect to the key performance metric of total loss on a real-world data set from Ningbo *City Bike* system[4]. The data set contains the following information: (i) customer trip records, from which we compute the demand scenarios; (ii) the number of stations, their capacity and initial distribution of bikes at each of the stations; (iii) geographical locations of stations (consisted of longitude and latitude), from which we calculate the relative distance between two stations; and (iv) the number of operators and their capacity. The dataset consists of 240 stations and 20 operators. We let each operator have a maximal capacity of 3 bikes, a maximum travel distance to pick-up station of 50 km and a maximum travel distance from pick-up station to drop-off station of 50 km[5]. The number of operators in different sub-networks are the same. And in each sub-network, operators are distributed across the stations uniformly at the start of the day[6]. As is common in the literature (*e.g.*, [7, 13]), we consider 6 hours of planning horizon in the morning peak (6 AM-12 PM) which is divided into 12 planning periods, each having a duration of 30 minutes. We generate 16 demand scenarios for the weekdays from one month of historical trip data. From 16 demand scenarios, 10 scenarios are used for training purposes and the other 6 scenarios are used for testing.

As the historical trip data ignores the unobserved lost demand, we employ a micro-simulation model from [8] with one minute of time discretization to determine the period when a station was empty and inject artificial demand based on the observed demand at that station in previous time step.

Note that the actual bike flows by users are restricted by the number of bikes available at a station, *i.e.*, not all user desired journey may happen. In simulations, we compute the actual bike flows based on both users' demand flows and bike availability. More specifically, we let $x_{s,s'}$ denote the actual bike flow from station s to s'. Then we calculate the value of $x_{s,s'}$ based on the following equation:

$$x_{s,s'} = F^t_{s,s'} \cdot \frac{\min\left(C^{\#}_s, \sum_{s' \in \mathcal{S}_c} F^t_{s,s'}\right)}{\max\left(\sum_{s' \in \mathcal{S}_c} F^t_{s,s'}, 1\right)}. \qquad (18)$$

Equation (18) characterizes two possible cases: (i) if the number of required bikes at a station is less than or equal to the number of bikes present there, then all the users bike requirements are satisfied and the actual flow is the same as the demand flow; and (ii) if the number of required bikes at a station is more than the number of bikes present there, only a proportion of demand flows can be satisfied. In each direction (from one specific station to another station), the ratio between actual flow and demand flow is $\frac{C^{\#}_s}{\sum_{s' \in \mathcal{S}_c} F^t_{s,s'}}$. Hence, the ratio are retained in the ratio between actual flows in different directions.

Once we compute the actual bike flows, we can update the distribution of bikes at station $s \in \mathcal{S}_c$ for the next planning period based

[4] We use this dataset for illustrative purposes. Using other real-world datasets (*e.g.*, Hubway dataset which is taken from Hubway bike sharing company of Boston [2] and used by Ghosh and Varakantham [9]) lead to similar results and thus omitted here.

[5] Distances are chosen based on the maximal speed of an electric vehicle and the length of a planning period. The results for other distances and other capacities of an operator are broadly similar.

[6] We use this setup for illustrative purposes. Other values lead to similar results and are thus omitted here.

on the following equation:

$$d_s^{\#,t+1} = d_s^{\#,t} + \sum_{v \in \mathcal{V}_c} \left(y_{s,v}^{-,t} - y_{s,v}^{+,t}\right) - \sum_{s' \in \mathcal{S}_c} x_{s,s'} + \sum_{s' \in \mathcal{S}_c} x_{s',s}, \tag{19}$$

which is the sum of unused bikes in planning period t, the net amount of dropped off bikes by the operators and the net amount of incoming bikes at station s.

However, equation (19) does not characterize the cases where a user cannot return their bike to a full station. Hence, we transfer excess bikes $(d_s^{\#,t+1} - C_s^{\#})$ to the nearest station with available docks, where $d_s^{\#,t+1}$ is calculated from (19). Note we also count the value of $(d_s^{\#,t+1} - C_s^{\#})$ as the number of loss docks. We update the value of $d_s^{\#,t+1}$ accordingly, which is then used to compute the repositioning policies for the next planning period.

Experiment Setup

We compare the MINLP with WA approach to other state of the art mechanisms[7]:

- **Static Repositioning**: We simulate the BSS without any system rebalancing. This approach serves as a baseline and shows the amount of loss that would occur if no dynamic repositioning is applied.
- **DRRPT**: As per Section 2, Ghosh and Varakantham [9] propose a DRRPT framework to generate the repositioning policy with the objective of minimizing the loss caused by starvation only. They do not consider the large-scale bike rebalancing problem, which makes this approach fail to be implemented in practice.
- **DRRPT with WA**: This approach combines the method proposed by Ghosh and Varakantham [9] with WA algorithm, which formulates the DRRPT framework for each sub-network to find repositioning policy separately.
- **MINLP with MFUA**: As per Section 2, Wang *et al.* [24] propose a clustering method called MFUA by considering both geographical information of stations and bike flows between stations. However, MFUA does not consider rebalancing workload across different sub-networks. As MFUA does not provide the repositioning policy, we use our MINLP framework in each sub-network that is generated by MFUA for finding the repositioning policy.

The performance of different approaches are compared in the following aspects:

- **Loss-minimizing performance**: We compare the total loss that occurred in the experimental scenarios after applying different approaches. A lower loss means a better loss-minimizing performance, which is the ultimate goal of performing bike rebalancing.
- **Runtime performance**: We compare the running time of different rebalancing approaches to solve the same experiment problems. We prefer an algorithm that is more efficient and can solve a problem in a shorter time.

Empirical Results

Loss-minimizing performance: Figure 1 shows the average total losses at all stations over different testing scenarios at different times

[7] All optimization problems involved are solved using IBM ILOG CPLEX Optimization Studio V12.10 within python code on a 3.2 GHz Intel Core i7 machine.

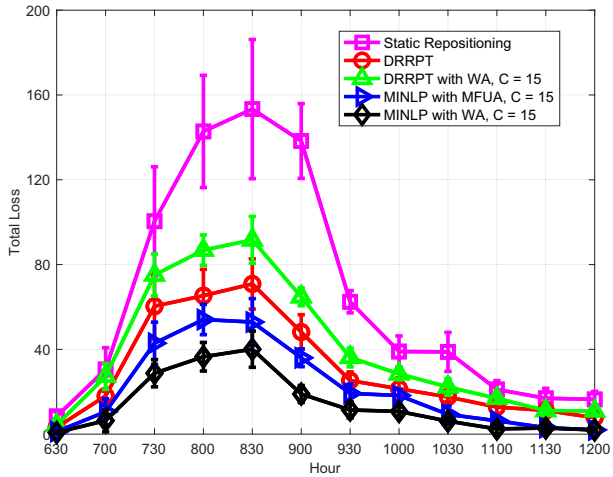

Figure 1: Total loss vs. different hours in a day

in a day. We let the number of sub-networks be 15 for illustrative purposes. The change in the cluster number does not impact the general loss-minimizing performance of our method.

As we can see, the total loss achieved by MINLP with WA is lower than that of other approaches at all times. Compared to Static Repositioning, MINLP with WA decreases the total loss by up to 90%. Compared to DRRPT and DRRPT with WA which only consider the starvation problem, the improvement regarding the total loss is at least 43%. This is because MINLP with WA addresses starvation and congestion problems simultaneously by tracing both docks and bike information. Compared to MINLP with MFUA which also addresses the congestion problem, MINLP with WA decreases the total loss by up to 60%. Because MINLP with WA considers the amount of rebalancing needed during the decomposition process, it can avoid the occurrence of sub-networks with unevenly high rebalancing demand and loss. As DRRPT performs global rebalancing, it outperforms DRRPT with WA which only conducts local rebalancing within sub-networks.

Runtime performance: Having shown MINLP with WA is performance efficiency, we now investigate the runtime performance of our approach on real-world demand scenarios. For a fair comparison, we only provide runtimes for DRRPT with WA, MINLP with MFUA and MINLP with WA as all of them generate rebalancing policy in each planning period in a round-robin fashion. We let the number of sub-networks be 15 for illustrative purposes (other values give broadly similar results).

Figure 2 illustrates the runtime with a different number of planning periods. The error bars are too small to be visible. As we can see, MINLP with WA outperforms the other approaches. Compared to MINLP with MFUA, MINLP with WA can reduce the runtime by up to 36%. This result demonstrates the advantage of WA over MFUA clustering in the aspect of runtime performance, which is a result of considering the needed amount of rebalancing during sub-network generation. The even distribution of loss among sub-networks can reduce the average difficulty of rebalancing within sub-networks so that we can find the rebalancing policy faster within the MINLP framework. As DRRPT only addresses the starvation problem, DRRPT with WA consumes much longer time than MINLP with WA to find the optimal rebalancing solution. Figure 2 shows that MINLP with WA can reduce the runtime by up to 30%.

Figure 2 also shows that DRRPT with WA takes a shorter runtime than MINLP with MFUA. This shows that decomposing a BSS into

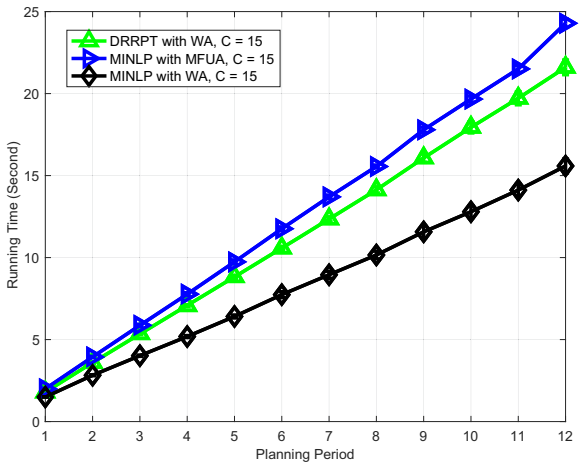

Figure 2: (Cumulative) Runtime time vs. different planning periods

small sub-networks plays a more important role than constructing a rebalancing model in terms of reducing the runtime.

6 Conclusion

We present a new scheme that can efficiently achieve resources (bikes) rebalance in a large-scale BSS. By jointly addressing the issues of starvation and congestion in a dynamic bike repositioning problem and considering the needed amount of rebalancing during the clustering process, MINLP with WA can significantly reduce the total loss in a BSS within a short running time. We believe that only by dealing with the interdependencies between bike demand, supply, operators' capacity and stations' geographic information can we make steps towards developing practical techniques for bike-sharing systems. MINLP with WA is the first such scheme.

Our next step is to consider a robust setting where the repositioning strategy is generated in each planning period by considering the demand uncertainties for multiple future time steps. Considering such robustness will lead to a different formulation and method that will further the scope of dynamic rebalancing in bike-sharing systems.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 62102343), Shenzhen Science and Technology Program (No. JCYJ20220818103006012), Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network (No. ZDSYS20220606100601002), Shenzhen Institute of Artificial Intelligence and Robotics for Society and Guangdong Provincial Key Laboratory of Aerospace Communication and Networking Technology.

References

[1] Jingxu Chen, Zhibin Li, Wei Wang, and Hang Jiang, 'Evaluating bicycle–vehicle conflicts and delays on urban streets with bike lane and on-street parking', *Transportation letters*, **10**(1), 1–11, (2018).

[2] Hubway Bike Sharing Company. Hubway dataset. https://hubwaydatachallenge.org/, 2021.

[3] Roberta Costa Affonso, Florent Couffin, and Patrice Leclaire, 'Modelling of user behaviour for static rebalancing of bike sharing system: Transfer of demand from bike-shortage stations to neighbouring stations', *Journal of Advanced Transportation*, **2021**, (2021).

[4] Edward W Forgy, 'Cluster analysis of multivariate data: efficiency versus interpretability of classifications', *biometrics*, **21**, 768–769, (1965).

[5] Iris A Forma, Tal Raviv, and Michal Tzur, 'A 3-step math heuristic for the static repositioning problem in bike-sharing systems', *Transportation research part B: methodological*, **71**, 230–247, (2015).

[6] Daniel Freund, Ashkan Norouzi-Fard, Alice Paul, Carter Wang, Shane G Henderson, and David B Shmoys, 'Data-driven rebalancing methods for bike-share systems', in *Analytics for the Sharing Economy: Mathematics, Engineering and Business Perspectives*, 255–278, Springer, (2020).

[7] Supriyo Ghosh, Jing Yu Koh, and Patrick Jaillet, 'Improving customer satisfaction in bike sharing systems through dynamic repositioning', in *Proceedings of the International Joint Conferences on Artificial Intelligence Organization*, (2019).

[8] Supriyo Ghosh and Pradeep Varakantham, 'Strategic planning for setting up base stations in emergency medical systems', in *Twenty-Sixth International Conference on Automated Planning and Scheduling*, (2016).

[9] Supriyo Ghosh and Pradeep Varakantham, 'Incentivizing the use of bike trailers for dynamic repositioning in bike sharing systems', in *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, (2017).

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

[11] Igor Griva, Stephen G Nash, and Ariela Sofer, *Linear and Nonlinear Optimization*, volume 108, Siam, 2009.

[12] Mingzhuang Hua, Jingxu Chen, Xuewu Chen, Zuoxian Gan, Pengfei Wang, and De Zhao, 'Forecasting usage and bike distribution of dockless bike-sharing using journey data', *IET Intelligent Transport Systems*, **14**(12), 1647–1656, (2020).

[13] Jinjia Huang, Mabel C Chou, and Chung-Piaw Teo, 'Bike-repositioning using volunteers: crowd sourcing with choice restriction', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11844–11852, (2021).

[14] Jingjing Li, Qiang Wang, Wenqi Zhang, Donghai Shi, and Zhiwei Qin, 'Dynamic rebalancing dockless bike-sharing system based on station community discovery', in *Proceedings of the International Joint Conferences on Artificial Intelligence Organization*, (2021).

[15] Xihan Li, Jia Zhang, Jiang Bian, Yunhai Tong, and Tie-Yan Liu, 'A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network', in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 980–988, (2019).

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

[17] Meghna Lowalekar, Pradeep Varakantham, Supriyo Ghosh, Sanjay Dominik Jena, and Patrick Jaillet, 'Online repositioning in bike sharing systems', in *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, (2017).

[18] Hongtao Lv, Chaoli Zhang, Zhenzhe Zheng, Tie Luo, Fan Wu, and Guihai Chen, 'Mechanism design with predicted task revenue for bike sharing systems', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 2144–2151, (2020).

[19] Mark EJ Newman and Michelle Girvan, 'Finding and evaluating community structure in networks', *Physical review E*, **69**(2), 026113, (2004).

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

[21] Tal Raviv and Ofer Kolka, 'Optimal inventory management of a bike-sharing station', *IIE Transactions*, **45**(10), 1077–1093, (2013).

[22] Tal Raviv, Michal Tzur, and Iris A Forma, 'Static repositioning in a bike-sharing system: models and solution approaches', *EURO Journal on Transportation and Logistics*, **2**(3), 187–229, (2013).

[23] PBSC UrbanSolution. The meddin bike-sharing world map report 2022 edition. https://bikesharingworldmap.com/, 2022.

[24] Yi-Jia Wang, Yong-Hong Kuo, George Q Huang, Weihua Gu, and Yaohua Hu, 'Dynamic demand-driven bike station clustering', *Transportation Research Part E: Logistics and Transportation Review*, **160**, 102656, (2022).