Spatial-Temporal Inventory Rebalancing for Bike Sharing Systems with Worker Recruitment

Yubin Duan, *Student Member, IEEE,* and Jie Wu, *Fellow, IEEE*

Abstract—Bike-sharing systems usually suffer from out-of-service events due to bike underflow or overflow. We propose to recruit workers to rebalance station loads. We partition the complex rebalancing problem in temporal and spatial domains. The temporal domain is divided into a sequence of slices with a fixed duration. In each slice, we allocate a pair of overflow/underflow stations to a worker such that the cost is minimized, which is NP-hard. A 3-approximation algorithm is proposed. We further investigate the worker shortage case and extend the matching algorithm to consider the number of unsatisfied users. Then, the configuration dynamic in the sequence of slices is captured by determining the rebalancing target for each rebalancing operation. We investigate heuristic approaches to minimize the total number of bike movements. Furthermore, we extend our scheme to dockless BSSs using clustering techniques. We simulate our algorithms on both real-world and synthetic datasets. Experiment results show that our approaches can reduce the average total detour per slice. In worker shortage, considering the number of unsatisfied users could improve the long-term performance of rebalancing. Besides, we find that our scheme could maintain worker satisfaction over multiple time slices, which indicates the sustainability of our rebalancing scheme.

Index Terms—Bike rebalancing scheme, minimum weighted matching, urban computing.

1 Introduction

With the boom of the sharing economy, bike-sharing systems (BSSs) have been widely deployed all around the world. [2], [3], [4]. The deployment of BSSs in cities brings benefits to both the environment and the economy. Specifically, it provides easily accessible bikes to the public with an affordable price, which greatly motivates users to ditch their cars for bikes. In addition, benefits brought by the development of BSSs also include financial savings for individuals, reduced congestion and fuel consumption, as well as transport flexibility. To fully extract these attractive benefits brought by the BSSs, the BSS operators need to maintain the sustainability of the system, while it is usually challenging.

The dynamics of user mobility often lead to an unbalanced bike distribution among stations. Fig. 1 illustrates the unbalanced bike distribution. Specifically, *overflow* stations are full of bikes and users cannot return bikes to these stations. In contrast, *underflow* stations are lack of bikes and no user could rent bikes. We refer both overflow and underflow stations as *Out-of-Service* (OoS) stations. Potential BSS users could be missed at these OoS stations. To maintain good user experience, it is necessary for BSS operators to rebalance the bike distribution by moving bikes from overflow stations to underflow stations.

The user-based rebalancing scheme has been deployed in some cities. For example, in the NYC Citibike system, a project named *Bike Angels*[1] has been launched. In this project, users would be rewarded points if they ride bikes from overflow stations to underflow stations indicated by the system operator. The points they gather could be redeemed as free rides or gift cards. An analysis on the Bike Angels project [5] has shown that the user-based rebalancing scheme is efficient and could be further optimized. In addition, compared with truck-based approaches, user-based approaches have several advantages. Truck-based rebalancing could only be implemented a few times per day, while user-based approaches are more flexible. Case studies on NYC and SF in Section 9 show the user-based approaches are usually more cost-efficient. Besides, user-based approaches are more environmental-friendly. However, existing researches about the user-based rebalancing mainly focus on designing incentive mechanism or pricing strategy. Little attention has been made on the assignment problem arises in the user-based rebalancing scheme.

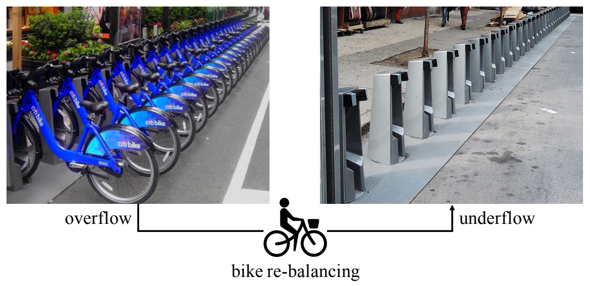


Fig. 1. A bike rebalancing scenario.

In this paper, a rebalancing scheme that recruiting workers to rebalance the BSS across spatial and temporal domains is proposed. Our scheme would determine the *rebalancing target*, which is the number of bikes to move in or out, for each station and recruit workers to rent or return bikes

Y. Duan and J. Wu are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, 19122.
E-mail: yubin.duan@temple.edu, jiewu@temple.edu
This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, and CNS 1564128. This paper is an extended version of the conference paper [1] published in IEEE ICDCS 2019.
Manuscript received September 17, 2019; revised February 14, 2020

1. https://www.citibikenyc.com/bikeangels/

from overflow stations to underflow stations. Besides, we propose to optimize worker assignments. We start from a simplified case where the number of available workers is sufficient to move all bikes indicated by rebalancing targets. Our objective is to minimize their overall detour distance. Then, we analyze the rebalancing problem with the worker shortage and jointly consider the worker detour and the number of unsatisfied users. Furthermore, we propose a worker satisfaction model to evaluate the number of available workers. Both worker detours and weather conditions are considered in the model.

Because of the complex user dynamics in spatial and temporal domains, the rebalancing is decoupled into a sequence of slices with a fixed duration in the temporal domain. For each time slice, the BSS operator would set rebalancing targets for overflow and underflow stations. In the spatial domain, the rebalancing scheme recruits workers and assigns a pair of overflow and underflow stations to each worker.Formally, we formulate the *Worker Assignment Problem* (WAP) for the spatial domain, and the *Configuration Design Problem* (CDP) for the temporal domain. Given the rebalancing targets for a fixed time slice and the sources and destinations of workers, the WAP aims to assign a pair of overflow and underflow stations to each worker to minimize the detour cost of all workers when the number of workers is sufficient. For the worker shortage case, the number of unsatisfied users is jointly considered with the detour distance. Besides, the CDP is formulated to find rebalancing targets for multiple slices to minimize the number of workers needed for rebalancing. Our scheme only moves bikes between stations without considering bike supplement or retirement, i.e., the total number of bikes moved in and out should be the same.

Decoupling the spatial and temporal domains in this way not only helps us handle the complex user dynamics, but also could maintain worker satisfaction. Specifically, minimizing the worker detour distance improves the worker satisfaction, which could further help attract more workers to participate in rebalancing. With more possible workers, the minimum overall detour cost could be reduced since the solution space is enlarged and there are more potential combinations to investigate. A smaller detour cost would further improve the worker satisfaction, and it forms a positive feedback loop on the worker satisfaction.

Although the rebalancing is decoupled in spatial and temporal domains, designing such a scheme is challenging. Specifically, the WAP in the *spatial* domain could be reduced from a 3D matching problem (i.e., matching among workers, rent stations, and return stations) which is NP-hard. In the *temporal* domain, it is hard to decide the number of time slices that the system should look ahead, even if the future user demands can be precisely predicted. It seems that choosing more time slices to look ahead leads to a better performance, while we find it is not true in some cases.

To solve the WAP, we propose a 3-approximation algorithm for the case where the number of workers is sufficient. Generally, our algorithm consists of two rounds of matching. It first finds the optimal match between rent and return stations, and then matches workers to paired stations. For the worker shortage case, we propose a utility function that jointly describes the worker detour distance and the expected number of unsatisfied users. A matching algorithm is proposed to iteratively improve the overall utility function value until it converges to a local optimal. As for the CDP, we investigate two heuristic approaches that consider different rebalance frequency granularities. One considers a fixed number of slices when deciding targets. The other one is inspired by [3] which greedily chooses the number of time slices to look ahead.

Our main contributions are summarized as follows:

- We propose a bike rebalancing scheme by recruiting workers. The temporal-spatial rebalancing problem is decoupled into WAP in the spatial domain and CDP in the temporal domain.
- A 3-approximation algorithm is proposed to solve the WAP, along with two heuristics for the CDP.
- We investigate the impact of worker shortages. It introduces worker-station dependencies to the matching. A two-sided-matching-based algorithm is proposed to deal with the externality.
- We further investigate the evolution of worker satisfaction during the rebalancing, and show that our scheme could maintain the worker satisfaction.
- We also extend our scheme for dockless BSSs by introducing virtual stations. The virtual stations are generated by applying clustering techniques.
- We use both real-world and synthetic datasets to simulate our scheme. Besides comparing the performances of our algorithms with previous approaches, we perform the cost-efficiency analysis of our scheme. Compared with truck-based rebalancing, recruiting workers is more cost-efficient.

2 Related Work

Problems in bike sharing include user demand prediction [4], [6], [7], [8], [9], [10] , rebalancing strategies [3], [11], [12], [13], [14], station location optimization [15], [16], bike lane planning [17], and suggestion for users' journeys [18], [19].

2.1 Demand prediction

The success of our scheme strongly relies on the precise prediction of user demands. Researches on demand prediction can be categorized as station level prediction and cluster level prediction. The earlier prediction approach focuses on predicting the bike usage at each station, such as [6], [7]. However, it may not always generate an accurate prediction [8]. Clustering similar stations is one way to address this problem. For example, Li et al. [8] proposed a hierarchical model that clusters stations based on location and transition patterns first, and then predicts the demand of each cluster. Chen et al. [9] further proposed to consider opportunistic contextual factors such as social and traffic events. Du et al. [10] used a density-peak based clustering algorithm to discover virtual stations and adapted a convolutional neural network to predict the demands. By utilizing demand prediction, our approach aims to optimize the worker assignment during rebalancing.

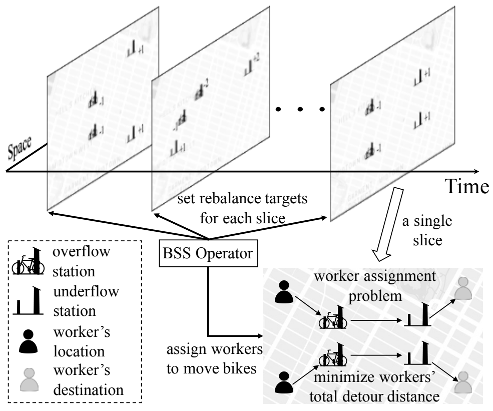

Space
Time
set rebalance targets for each slice
a single slice
BSS Operator
overflow station
underflow station
worker's location
worker's destination
worker assignment problem
assign workers to move bikes
minimize workers' total detour distance

Fig. 2. An overview of our bike rebalancing scheme.

2.2 Rebalancing strategy design

Two major approaches in the design of rebalancing strategies for BSSs are truck-based approaches [3], [20], [21], [22], [23] and user-based approaches [13], [24], [25], [26].

In the truck-based approach, a fleet of trucks is hired to transport bikes between overflow and underflow stations. In static models, the target inventory level for stations is constant. Liu et al. [3] proposed a method that first clusters stations according to station locations and status, and assigns a truck to each cluster. The truck routing is modeled as an integer programming problem. For dynamic bike reposition, Li et al. [20] proposed a reinforcement-learning model to learn an inner-cluster reposition policy. Truck-based rebalancing strategies cannot be directly applied to our problem since each worker can only carry one bike at a time while trucks can carry multiple bikes during rebalancing and trucks need to route among all OoS stations.

The existing truck-based approaches have several shortcomings. Firstly, in a truck-based approach, the rebalancing is usually implemented at certain times during the day, and the bike inventory among stations cannot be adjusted in real time. It costs a significantly long time (usually several hours) for a truck to route among stations. In contrast, workers could finish rebalancing within a short time, since they move bikes simultaneously. Secondly, hiring a fleet of trucks is associated with considerable costs and is environmentally unfriendly. During rebalancing, trucks are routing among overflow and underflow stations. Note that trucks have a capacity limitation, which introduces additional detour distances for truck routing. Optimizing the truck routing problem with these delivery constraints is NP-hard [23]. Not to mention, the cost of fuel, recruiting drivers, and maintaining or renting trucks is also considerable. The worker-based approach is more environmentally friendly and cost-efficient. The cost-effectiveness is analyzed in Section 9.

In the user-based approach, the BSS operator offers users monetary incentives and motivates them to rent/return bikes at certain stations [13], [24]. It is expected to achieve a self-balanced system to improve the overall service level by controlling the user's dynamics. Designing the pricing mechanism is challenging since the user cost is unknown. Waserhole [24] presented a dynamic pricing mechanism that incentivizes users to redistribute bikes by providing alternate rental prices. Singla et al. [13] proposed a crowdsourcing pricing mechanism to incentivize users based on the multi-armed bandit model. However, these approaches do not consider the spatial imbalance of BSS. Pan et al. [25] studied the pricing strategy for dockless bike sharing systems and proposed a deep reinforcement learning algorithm for deciding the incentive price. Although [25] divided the continuous map space into discrete regions, they did not consider the capacity limitation of each region. However, in a docked bike sharing system, each station has such a limitation. Hence, we cannot directly apply their rebalancing strategy to our problem. Besides, none of them considers the worker satisfaction, while our scheme could improve the worker satisfaction.

TABLE 1
Table of Notations

Notations	Description
W, S, D	the set of workers, their sources, and destinations
w, s_w, d_w	a worker and his/her source, and destination
B, N, P	the set of bike stations, rent, and return stations
b, n, p	a bike station, a rent station, and a return station
η_b	the capacity of station b
$\boldsymbol{\eta}_B$	the capacity vector for all stations in B
T, t	the set of time slices, and a time slice
$\phi_b(t)$,	the state of station b at the beginning of slice t
$\tau_b(t)$,$\rho_b(t)$	the demand and rebalancing target of b during t
$\boldsymbol{\tau}_B(t)$,$\boldsymbol{\rho}_B(t)$	the demand and rebalancing target vector of B

2.3 3D matching

A maximum 3D matching problem (unweighted) is a special case of a 3-set packing problem [27]. In the k-set packing problem, we are given a family of sets of size at most k, and the goal is to find a maximum size subfamily of pairwise disjoint sets. The best known polynomial time approximation ratio for k-Set Packing is $(k+1+\epsilon)/3$ [27]. As for the weighted version of k-set packing problem, Arkin et. al. [28] introduced a $(k-1+\epsilon)$-approximation algorithm based on local search. Chandra et. al. improved it to $(2k+2+\epsilon)/3$-approximation in [29]. Then, Berman further improved it to $(k+1+\epsilon)/2$-approximation in [30]. These algorithms are all based on local search.

3 System Model

In our work, we propose to rebalance the BSS by recruiting workers. The overview of our rebalancing scheme is shown in Fig. 2. Specifically, the BSS operator conducts several rounds of rebalancing among bike stations at predefined time intervals during the rebalancing period. In each round of rebalancing, the operator recruits a set of workers. Each worker can ride one bike from the overflow station to the underflow station assigned by the system. For multiple rounds of rebalancing, the operator needs to set rebalancing targets, i.e. the number of bikes to move in or out at stations, for each time slice. The rebalancing targets are decided based on states and demands of stations in the BSS.

In each round of rebalancing, the BSS operator needs to gather sources and destinations of workers, before it can generate an assignment for each worker. The worker

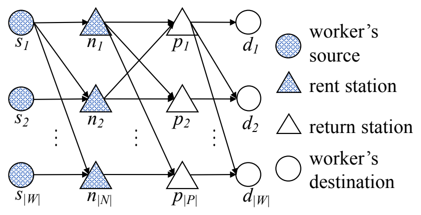


Fig. 3. An illustration of the assignment graph.

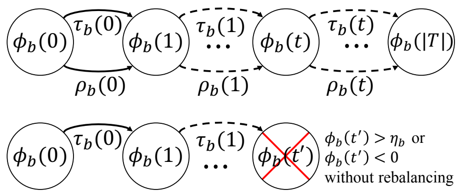


Fig. 4. Station state evolution in discretized time.

assignment is studied over the spatial domain. A worker is denoted by w, and the worker set is denoted by W. Each worker has his/her source s_w and destination d_w. The sets of s_w and d_w are denoted by $S = \{s_w | w \in W\}$ and $D = \{d_w | w \in W\}$, respectively. A bike station is denoted by b, and the set of bike stations is denoted by B. The number of bike stations is $|B|$, where $|\cdot|$ represents the cardinality of a set. Each station b has a capacity η_b and can store at most η_b bikes. Capacities of different stations can be different.

For multiple rounds of rebalancing, the states and user demand predictions for bike stations are necessary. The set of time slices is denoted by T, and each time slice is $t \in T$. The state of station b for slice t is defined as the number of on-dock bikes at the beginning of each time slice t, and it is a non-negative integer denoted by $\phi_b(t)$. The user demand of station b during the time slice t is an integer denoted by $\tau_b(t)$. It can be precisely predicted based on history bike rental information. The demand prediction problem has been well studied, such as the predictors proposed in [3], [9]. Note that the demand $\tau_b(t)$ can be either positive or negative. A positive demand means that more bikes are returned to the station and the number of on-dock bikes increases during time slice t. A negative demand has the opposite meaning.

Based on the state and demand information, the operator can determine a rebalancing target $\rho_b(t)$ for each station b during t. A positive $\rho_b(t)$ means that the station b needs $\rho_b(t)$ bikes, and a negative $\rho_b(t)$ means $|\rho_b(t)|$ bikes shall be removed from the station. $\sum_{b \in B} \rho_b(t)$ should be 0 since the rebalancing operation does not affect the number of bikes in the system. In each time slice t, bike stations with negative targets can form a *rent station set* N, and stations with positive targets can form a *return station set* P. Formally, $N = \{b \in B | \rho_b(t) < 0\}$ and $P = \{b \in B | \rho_b(t) > 0\}$. Elements in sets N and P are denoted as n and p, respectively.

In the paper, we assume the number of workers who can be recruited is large enough. If not, the BSS operator could use trucks for rebalancing. Truck-based rebalancing strategies are reviewed in section 2. After recruiting a sufficient amount of workers, the operator needs to determine a *rebalancing assignment* $(w, n, p) \in W \times N \times P$ for each worker, meaning that the worker w should rent a bike from station n and return it to station p. The corresponding detour distance of worker w is denoted by a real number δ_w. Formally, $\delta_w = dis(s_w, n) + dis(n, p) + dis(p, d_w) - dis(s_w, d_w)$, where $dis(\cdot, \cdot)$ is the distance function used to calculate the geographical distance between two locations.

The assignment for workers is modeled by the assignment graph $G = (V, E)$ shown in Fig. 3. The vertex set is constructed by workers' sources, destinations, and bike stations, i.e. $V = S \cup N \cup P \cup D$. The directed weighted edges have three types, including edges from S to N, edges from N to P, and edges from P to N. Formally, $E = (S \times N) \cup (N \times P) \cup (P \times D)$. We use the function $e : V^2 \mapsto \mathbb{R}$ to denote the edge weights. The weight is quantified by the geographical distance. For example, the weight of an edge from s_w to n is the the corresponding distance, i.e., $e(s_w, n) = dis(s_w, n)$. A flow from s_w to d_w is equivalent to an assignment for w. The sum weight of edges on the flow is w's moving distance, and is used to quantify w's detour since the distance between s_w and d_w is constant.

The time evolution of the state of a station is modeled by discretized time series. As shown in Fig. 4, the state $\phi_b(t+1)$ is determined by the state $\phi_b(t)$, the demand $\tau_b(t)$, and the rebalance operation $\rho_b(t)$ during time slice t. Formally, $\phi_b(t+1) = \phi_b(t) + \tau_b(t) + \rho_b(t)$. Without the rebalancing operation $\rho_b(t)$, an OoS event may occur- either an overflow event with $\phi_b(t') > \eta_b$, or an underflow event with $\phi_b(t') < 0$.

4 Problem Formulation and Analysis

4.1 Worker assignment problem

In a fixed time slice, we propose the Worker Assignment Problem (WAP). Given rebalancing targets of stations, sources and destinations of the workers, the WAP aims to find out an optimal assignment for workers that minimizes their overall detours when workers are moving bikes between stations. The WAP can be seen as finding a minimum cost flow from S to D. Based on the assignment graph, an assignment for worker w is equivalent to a flow from s_w to d_w. As shown in Fig. 3, a flow from s_w to d_w passing through a node n and a node p represents a worker w renting a bike from n and returning to p. Our WAP problem is to minimize the total cost of the k flows. We use $f(\cdot, \cdot)$ to denote the flow rate, which is an integer, between two vertices.

This problem can be formulated by an Integer Programming model, which can be described as:

$$\min \sum_{W,N,P} f(s_w,n)e(s_w,n) + f(n,p)e(n,p) + f(p,d_w)e(p,d_w)$$

$$s.t. \sum_{N} f(s_w,n) = 1, \ \sum_{P} f(p,d_w) = 1, \forall w \in W \tag{1}$$

$$\sum_{W} f(s_w,n) = |\rho_n|, \sum_{W} f(p,d_w) = |\rho_p|, \forall n \in N, p \in P \tag{2}$$

$$f(n,p) = \sum_{W} (f(s_w,n) \cdot f(p,d_w)), \forall n \in N, p \in P \tag{3}$$

$$f(s_w,n), f(p,d_w) \in \{0,1\}, \ f(n,p) \in \mathbb{N} \tag{4}$$

Our objective is to minimize the total weight of the flow. For the WAP in a Euclidean plane, it is the overall detour distances. In a more general version, which is denoted as the *general WAP*, the edge weight could be any kind of detour cost, such as traveling time. Eq. (1) is the assignment constraint, which means that each worker should be assigned to a rent station in N and a return station in P. Eq. (2) is the target constraint. It means that each station's positive or negative targets should be satisfied. Eq. (3) is the consistency constraint. If there is a flow from s_w to n and from p to d_w (i.e., the worker w rents a bike from n and returns to p), there should be a flow from n to p. Eq. (4) is the flow-rate constraint. The rates of flows from S to N and flows from P to D can be either 0 or 1. The rates of flows from N to P should be a non-negative integer.

4.2 Configuration design problem

In this subsection, we formulate the Configuration Design Problem (CDP) for multiple time slices. Given the demand prediction of each time slice, CDP aims to design a set of rebalancing targets which can minimize the number of bike movements over multiple time slices. We choose this objective as an effort to minimize workers' total detour over multiple slices. Formally, given the initial state vector $\boldsymbol{\phi}_B(0)$ and the demand vectors $\boldsymbol{\tau}_B(t)$ for all time slices $t \in T$, the objective is to find the rebalancing target vector $\boldsymbol{\rho}_B(t)$ for each time slice t, and minimize the total amount of moved bikes. Note that our rebalancing scheme could be implemented as a complementary of truck-based rebalancing. System operators can easily split the demand of each station into two parts: one for truck-based rebalancing schemes such as [23] and one for the worker-based approach.

$$\min \sum_{t\in T}\sum_{b\in B} |\rho_b(t)|$$

$$s.t.\quad \nu_b \le \phi_b(t) \le \eta_b - \nu_b,\ \forall\, b\in B,\ \forall\, t\in T \tag{5}$$

$$\sum_{b\in B} \rho_b(t) = 0,\ \forall\, t\in T \tag{6}$$

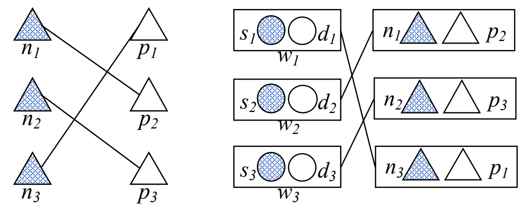
$$\rho_b(t) \in \mathbb{N},\ \forall\, b\in B,\ \forall\, t\in T \tag{7}$$

Note that $\sum_{t\in T}\sum_{b\in B}|\rho_b(t)|$ is actually twice the number of moved bikes, since moving a bike from a rent station to a return station can simultaneously fulfill a positive target and a negative target. Eq. (5) is the capacity constraint. Considering the predicted demand of each station may slightly differ from the actual demand, we set safety margins $\nu_b < \eta_b/2$ for stations $b \in B$. A station is treated as an underflow (or overflow station) if the number of bikes at the station is lower than ν_b (or higher than $\eta_b - \nu_b$). Eq. (6) is the matching constraint which requires the sum of rebalancing targets among all stations at each time slice to be 0. This is because the total number of bikes should remain unchanged during rebalancing. Eq. (7) indicates that rebalancing targets are non-negative integers.

4.3 Problem hardness

Theorem 1. The general WAP for a time slice is NP-hard.

Proof: The proof shows that our optimization problem can be reduced from a weighted 3D matching problem. The decision problem of 3D matching is known to be one of Karp's 21 NP-complete problems [31], and the weighted 3D matching problem belongs to the NP-hard problem class.



(a) Matching between stations (b) Matching between workers and station pairs

Fig. 5. Procedures of the Two-Round Matching algorithm.

The general WAP can be reduced from a maximum 3D matching problem if we let tripartite sets denote the set W, P, and N. A 3D matching M with $|M| = |W|$ is a legal assignment for bike rebalancing. That is, a triple (w, n, p) in M represents that worker w should move a bike from station n to station p. The weight of each triple is equal to a large positive constant minus the cost of the corresponding journey (i.e. the sum of the costs from s_w to n, then to p, and finally to d_w). A maximum matching maximizes the sum of negations of journey costs. Therefore, it minimizes the sum of journey costs. The assignment is optimal for workers because it minimizes workers' total detour costs. An instance of the maximum 3D matching is thus corresponding to an instance of the general WAP. ■

5 Fixed Time Slice Optimization

5.1 Two-Round Matching algorithm

In this section, we introduce a 3-approximation algorithm to solve the WAP in a Euclidean plane. Our algorithm is based on two stages of matchings. In the Two-Round Matching (TRM) algorithm, we first apply the weighted bipartite matching algorithm on the bipartite graph constructed by N and P. The procedure of the first round of matching is shown in Algorithm 1. Lines 1-4 in the algorithm are to construct the bipartite matching graph. In the weighted matching algorithm, a vertex can only be matched once. Therefore, we duplicate the stations based on their rebalancing targets to guarantee that the target of each vertex in set N (or P) is -1 (or 1). The vertex set is initialized in line 5 and the edge set is initialized in lines 6-7. We apply weighted matching on the bipartite graph in line 8. Notice that the weighted matching algorithms are designed for maximization. To find the non-zero minimum perfect matching, we modify the edge weight when using the maximum weighted matching algorithm. The modified weight is calculated by subtracting the original weight from a large number. It is $e' = LC - e$, where LC is a large constant number and e' is the modified edge weight.

The second round of matching generates the assignments for workers, which is also a bipartite matching problem. We construct another bipartite graph where one part of the graph represents workers W and the other part represents rent/return station pairs $N \times P$. The weighted edge between a worker w and a combination (n, p) equals the total moving distance of w if he/she rents a bike at station n and returns it at p. Notice that we use the workers' total moving distance instead of their overall detour as the edge

Algorithm 1 Two-Round Matching Algorithm - Stage 1

```
Input: positive station set $P$ as well as the corresponding demand set $\{\rho_P\}$, and negative station set $N$ as well as the corresponding demands $\{\rho_N\}$
Output: rent/return station pairs which are used for worker allocation in the next stage
1: for each $n \in N$ and $\rho_n < -1$ do
2:     Copy station $n$ $(|\rho_n| - 1)$ times in $N$.
3: for each $p \in P$ and $\rho_p > 1$ do
4:     Copy station $p$ $(\rho_p - 1)$ times in $P$.
5: $V \leftarrow N \cup p$, $E \leftarrow \emptyset$
6: for $n \in N, p \in P$ do
7:     $E \leftarrow E \cup (n,p)$, $e(n,p) \leftarrow dis(n,p)$
8: $X \leftarrow$ min-cost perfect matching of $G(V, E)$.
9: return $X$ as the rent/return station pair
```

weight. Actually, the detour reaches the minimum when the total moving distance is minimized, because the total length of workers' original journeys is constant. The procedure of the second round of matching is shown in Algorithm 2. Line 1 initializes the vertex set $V' = W \cup X$ and edge set $E' = \emptyset$. The weight edges are added to E in lines 2-3. Similarly, as in Algorithm 1, we modify the edge weights and apply the maximum weighted matching algorithm in line 5. Finally, an edge set M is calculated, and it constitutes the assignments for workers. The workers' overall moving distance is the sum weight of edges in M, and is denoted by $\|M\|$.

Theorem 2. The TRM is a 3-approximation algorithm for the WAP in the Euclidean space.

Proof: The calculation of the 3-approximation ratio is based on the triangle inequality and the optimality of each matching stage of TRM. For each negative station $n \in N$, there is a corresponding positive station $p \in P$ which is assigned in the first round of TRM. In addition, the station pair (n,p) is matched to a worker w with source s_w and destination d_w in the second round of TRM. We assume that in the optimal solution, the station n should be paired with station p^*, and the station pair (n,p^*) should be balanced by the worker w^* whose source and destination are s_w^* and d_w^* respectively. The relation among these nodes is shown in Fig. 6, which is a geometric graph in the Euclidean space. The total moving distance of workers generated by our algorithm is $\sum_{n\in N}(dis(s_w,n)+dis(n,p)+dis(p,d_w))$, and the optimal value is $\sum_{n\in N}(dis(s_w^*,n)+dis(n,p^*)+dis(p^*,d_w^*))$.

Based on the triangle inequality, we can conclude that $dis(p,p^*) \le dis(n,p) + dis(n,p^*)$ and $dis(p,d_w^*) \le dis(p,p^*) + dis(p^*,d_w^*)$ for each $n \in N$. According to the optimality of the first round of matching, we can conclude that $\sum_{n\in N} dis(n,p) \le \sum_{n\in N} dis(n,p^*)$. Besides, the optimality of the second round of matching guarantees that $\sum_{n\in N}(dis(s_w,n) + dis(p,d_w)) \le \sum_{n\in N}(dis(s_w^*,n) + dis(p,d_w^*))$. Combining these inequity relationships:

$$\begin{aligned}
&\sum_{n\in N}(dis(s_w,n) + dis(p,d_w)) \le \sum_{n\in N}(dis(s_w^*,n) + dis(p,d_w^*))\\
&\le \sum_{n\in N}(dis(s_w^*,n) + dis(p,p^*) + dis(p^*,d_w^*))\\
&\le \sum_{n\in N}(dis(s_w^*,n) + dis(n,p) + dis(n,p^*) + dis(p^*,d_w^*))\\
&\le \sum_{n\in N}((dis(s_w^*,n) + 2dis(n,p^*) + dis(p^*,d_w^*)).
\end{aligned}$$

Algorithm 2 Two Round Matching Algorithm - Stage 2

```
Input: worker set $W$, rent/return station pair set $X$
Output: rent and return allocation for each user, and workers' total travel distance $C$
1: $V' \leftarrow W \cup X$, $E' \leftarrow \emptyset$
2: for $w \in W$, $(n,p) \in X$ do
3:     $E' = E' \cup (w,(n,p))$,
       $e(w,(n,p)) \leftarrow dis(s_w,n) + dis(n,p) + dis(p,d_w)$
4: $M \leftarrow$ min-cost perfect matching of $G'(V', E')$.
5: return $M$ as the rent and return allocation for workers, and $||M||$ as workers' total travel distance.
```

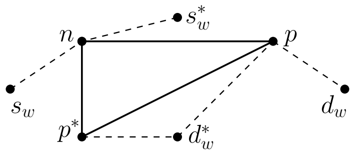


Fig. 6. The relation between the TRM assignment and the OPT.

Therefore,

$$\begin{aligned}
&\sum_{n\in N}(dis(s_w,n)+dis(n,p)+dis(p,d_w))\\
&= \sum_{n\in N}(dis(s_w,n) + dis(p,d_w)) + \sum_{n\in N} dis(n,p)\\
&\le \sum_{n\in N}((dis(s_w^*,n)+2dis(n,p^*) + dis(p^*,d_w^*)) + \sum_{n\in N} dis(n,p^*)\\
&\le 3\sum_{n\in N}(dis(s_w^*,n)+dis(n,p^*)+dis(p^*,d_w^*)) = 3OPT.
\end{aligned}$$

The 3-approximation holds. ■

6 Multiple-Slice Optimization

6.1 k-slice greedy algorithm

For CDP, we focus on the multiple-time-slice optimization. Firstly, we introduce a heuristic greedy algorithm: k-slice Greedy Algorithm (kGA). The idea is to look ahead k time slices, and find a proper rebalancing target for the following k time slices, where k is constant and is set by the BSS operator. The operator conducts a round of rebalancing for every k time slices. Recall that a proper target should minimize the number of bikes needed to move, and the sum of targets for all stations should be 0. In kGA, we firstly choose the target for each station greedily, and then check the sum of the chosen targets over all stations.

The procedures of kGA are illustrated in Algorithm 3. We first calculate the range of the feasible rebalancing target for each station. The largest value in the range for station b is denoted by α_b. A positive α_b means that at most, α_b bikes can be moved into b within the following k time slices. The smallest value in the range is denoted by β_b, and a negative β_b means at most $|\beta_b|$ bikes can be removed from b. If $\alpha_b < 0$, the station b will face overflow events in the following k slices and have to remove bikes beforehand. In contrast, $\beta_b > 0$ has the opposite meaning.

Algorithm 3 k-slice Greedy Algorithm.

```
Input: station capacity vector $\boldsymbol{\eta}_B$, current station state vector $\boldsymbol{\phi}_B(t_0)$, the demand prediction $\{\boldsymbol{\tau}_B(t)\}(t_0 \le t \le t_0 + k)$ of following time slices
Output: rebalancing target vector $\boldsymbol{\rho}_B(t_0)$
1: $\alpha_b \leftarrow \eta_b - (\phi_b(t_0) + \max[\sum_{l=t_0}^{t_0} \tau_b(l), \cdots, \sum_{l=t_0}^{t_0+k} \tau_b(l)])$, $\beta_b \leftarrow 0 - (\phi_b(t_0) + \min[\sum_{l=t_0}^{t_0} \tau_b(l), \cdots, \sum_{l=t_0}^{t_0+k} \tau_b(l)])$, $\forall b \in B$
2: positive target vector $\boldsymbol{\rho}_+ \leftarrow \text{find}(\boldsymbol{\beta}_B > 0)$, negative target vector $\boldsymbol{\rho}_- \leftarrow \text{find}(\boldsymbol{\alpha}_B < 0)$, $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho}_- + \boldsymbol{\rho}_+$
3: if $\text{sum}(\boldsymbol{\rho}_+, \boldsymbol{\rho}_-) > 0$ then
4:     Iteratively decrease $\rho_b, \arg\min_b(\boldsymbol{\beta}_B)$ (use final state to break tie) by 1, until $\text{sum}(\boldsymbol{\rho}_+, \boldsymbol{\rho}_-) = 0$
5: else if $\text{sum}(\boldsymbol{\rho}_+, \boldsymbol{\rho}_-) < 0$ then
6:     Iteratively increase $\rho_b, \arg\max_b(\boldsymbol{\alpha}_B)$ by 1, until $\text{sum}(\boldsymbol{\rho}_+, \boldsymbol{\rho}_-) = 0$
7: return $\boldsymbol{\rho}_B(t_0)$ as the rebalance target vector
```

6.2 Greedily look-ahead algorithm

Besides manually defining k for kGA, we can also design a greedy algorithm to automatically select the largest k. Following this approach, we propose another target setting algorithm: Greedily Look-ahead Algorithm (GLA).

The procedures of the GLA are illustrated in Algorithm 4. The survival time (i.e., the time duration before OoS events occur) of a station is related to the rebalancing target. In line 1, GLA tests the survival time of each station b under all possible rebalancing targets, and stores the longest one, which is denoted by T_b. In line 2, the GLA sets k as the minimum $T_b, \forall b \in B$. Unfeasible situations may occur if k is large. To deal with this case, the GLA repeatedly decreases k by 1 and then calls kGA until a feasible target is found.

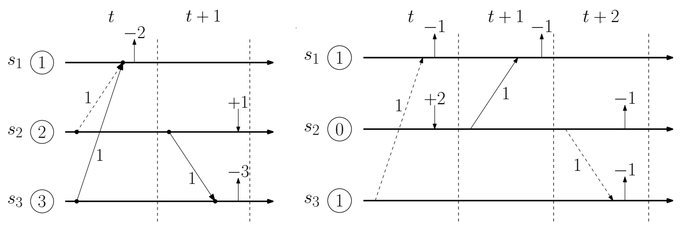
It seems looking up more time slices (i.e. larger k) can generate better results in terms of the total moved bikes in a day. However, this is not the case, since the greedy algorithm cannot generate an optimal substructure in our problem.

It is more clear to demonstrate the difference between GLA and kGA by using a *time-space view* from the classic distributed system [32]. Fig. 7(a) shows an example that choosing a larger k in kGA could achieve a better performance in the temporal domain optimization. In the figure, the slanted arrow line corresponds to a bike re-balancing event between stations. Each slice corresponds to a time period and two slices are separated by a vertical dotted line. Rent (return) event at a station is represented by a short outward (inward) vertical arrow. For simplicity, all bike movements are completed in a time slice. The initial state of three stations is $(1, 2, 3)$. The user demand at t is $(-2, 0, 0)$. To avoid OoS stations, either s_2 or s_3 should move one bike to s_1. In this setting, looking ahead two slices is better than just looking ahead one. For $k = 1$, i.e., choosing one slice to look ahead, the scheme would find no difference between removing one bike from s_3 or s_2. If s_3 is selected to remove one of its three bikes and the user demand at $t+1$ turns out to be (0, 1, -3), then s_2 has to move one of its bikes to s_3 at $t + 1$. On the other hand, when $k = 2$, the rebalancing scheme could notice that it is better to move one bike from s_2 to s_1, and it saves one bike movement.

From the time-space view, it is more clear to find that look ahead may not always generate a better result. Fig. 7(b)

Algorithm 4 Greedily Look-ahead Algorithm

```
Input: station capacity vector $\boldsymbol{\eta}_B$, current station state vector $\boldsymbol{\phi}_B(t_0)$, the demand prediction $\{\boldsymbol{\tau}_B(t)\}(t_0 \le t \le |T|)$ of following time slices
Output: rebalancing target vector $\boldsymbol{\rho}_B(t_0)$
1: Calculate longest survival time $T_b$ under different targets $\rho_b(t_0)$ for each station $b$, i.e., $\arg\max_{\rho_b(t_0)}(T_b | 0 \le \sum_{t=t_0}^{t_0+T_b} \tau_b(t) + \phi_b(t_0) + \rho_b(t_0) \le \eta_b)$, $\forall b \in B$
2: $k \leftarrow \min(T_b, \forall b)$
3: Apply $k$GA to calculate $\boldsymbol{\rho}_B(t_0)$. Repeatedly decrease $k$ if $k$GA cannot find a feasible set of targets.
4: return $\boldsymbol{\rho}_B(t_0)$ as the rebalance target
```



(a) An example of 2GA outperforming 1GA (b) An example of 1GA outperforming GLA

Fig. 7. Illustrations of various look ahead schemes.

shows the example in which using the GLA approach would lead to a worse performance compared with 1GA in terms of the number of bike movement. In Fig. 7(b), the station state is initially set as $(1, 0, 1)$. The user demands vectors are set as $(-1, 2, 0)$, $(-1, 0, 0)$, and $(0, -1, -1)$ for slices t, $t+1$, and $t+2$, respectively. If we use the GLA approach at the first time slice, the rebalancing targets would be moving one bike from s_3 to s_1 so that the resulting state $(2, 0, 0)$ could survive two slices, i.e. there is no OoS stations in the following two slices. However, at slice $t+2$ with state (0, 2, 0), s_2 has to move at least one bike to s_3 to meet the user demand. In total, there are two bike movements. In contrast, if we look at only one time slice, i.e., $k = 1$, no action is needed at slice t, one bike would be moved from s_2 to s_1 at slice $t+1$, and no action is needed at slice $t+2$. There is only one bike movement, which is better than using GLA.

7 BSS Balancing in Worker Shortage

When the number of workers is sufficient, the rebalancing targets set by configuration design problem (CDP) could be entirely satisfied. However, in practice, it is hard to recruit so many workers for each time slot. Therefore, we also investigate the worker assignment problem (WAP) when the system is short of workers.

The shortage of rebalancing workers would increase the number of unsatisfied users. The worker shortage refers to the case in which the number of workers recruited at each time slot is less than the number of bike movements required by the rebalancing targets. An unsatisfied user is a user who is going to rent/return a bike but cannot find an available bike/dock at its source/destination station. If users cannot find available bikes, they would leave the system. If they cannot find available bikes at stations near their destinations, they have to take a detour and may

Algorithm 5 Worker-Station Matching with Externality (WSME)

```
Input: worker set $W$, rent/return station pair set $X$ generated by Algorithm 1
Output: assignments for all available workers $\mathcal{A}$
1: $\mathcal{A} \leftarrow$ randomly match $(w,(n,p)), \forall w \in W, \forall (n,p) \in X$.
2: converge ← False
3: while not converge do
4:     Sort workers in $W$ based on their contribution to the utility function in descending order. The contribution of $(w,(n,p))$ can be calculated as $u(t,(w,(n,p))) = \frac{f(s_w,n)dis(s_w,n)+f(n,p)dis(n,p)+f(p,d_w)e(p,d_w)}{\max_{W,N} dis(s_w,n)+\max_{N,P} dis(n,p)+\max_{P,W} dis(p,d_w)} + C_u \cdot \frac{\upsilon_n(\phi_n(t))+\upsilon_p(\phi_p(t))}{\max_{b\in B}\upsilon_b(\phi_b(t))}$
5:     for $w \in W$ do
6:         if $\exists (n',p') \in X$ such that $u(t,\mathcal{A}') < u(t,\mathcal{A})$, where $\mathcal{A}' = \mathcal{A} \backslash (w,(n,p)) \cup (w,(n',p'))$ then
7:             Update $\mathcal{A} \leftarrow \mathcal{A}'$
8:         else
9:             converge ← True
10: return $\mathcal{A}$ as the worker assignment.
```

lose interest in using the system again. Both events affect the potential profit of the system operator. Therefore, the number of unsatisfied users should be considered in the objective of our WAP when there is a worker shortage.

We use a Markov Chain to model the number of unsatisfied users. Specifically, we have introduced the time series model to describe the fluctuation of the number of bikes at each station. In the Markov Chain model, we use $\upsilon_b(\phi_b(t))$ to denote the expected number of unsatisfied users in station $b \in B$ during the remaining rebalancing cycle after t, where $\phi_b(t)$ describes the number of bikes in station b at time t. When the further user demands are known, the value of $\upsilon_b(\phi_b(t))$ can be calculated by running a simulator. When considering a more practical case where the future demands are estimated, the expected value of $\upsilon_b(\phi_b(t))$ can be numerically calculated by the model in [37].

Briefly, the state of the station is modeled by a bounded birth and death process with birth rate μ_t and death rate λ_t. Let $\pi(i,j,t)$ denote the probability of the station having j bikes at time t provided that it has i bikes at time 0. Then, the value of $\upsilon_b(\phi_b(t))$ can be expressed as follows:

$$\upsilon_b(\phi_b(t)) = \int_t^T (\pi(\phi_b(t),0,t) + \pi(\phi_b(t),\eta_b,t))\mathrm{d}t.$$

The first term represents the accumulated number of underflow events and the second term shows the accumulated number of overflow events. T represents the end of a rebalancing cycle, e.g., a day. The summation of two terms is the accumulated number of unsatisfied users.

In our discretized time domain, the value of $\upsilon_b(\phi_b(t))$ can be numerically calculated [33]. Let δ denote the length of each time slot. According to [33], the number of expected dissatisfaction value can be estimated as $\upsilon_b(\phi_b(t)) \approx \delta \sum_{k=0}^{T/\delta-1}(\pi(\phi_b(t),0,t+0.5\delta)\mu_{k\delta} + \pi(\phi_b(t),\eta_b,t+0.5\delta)\lambda_{k\delta})$.

We define a utility function to jointly describe the worker detour distances and the expected number of unsatisfied users. Let $u(t,\mathcal{A})$ denote the utility function at time t with the worker assignment $\mathcal{A}$. The worker assignment $\mathcal{A}$ consists of the flows from worker sources S to their destinations D, passing through overflow stations N. The flow functions $f(\cdot,\cdot)$ describe the number of bikes moved between two parties, and it is introduced in subsection 3.1.Then, the utility function is defined as $u(t,\mathcal{A}) = \frac{\sum_{W,N,P} f(s_w,n)dis(s_w,n)+f(n,p)dis(n,p)+f(p,d_w)e(p,d_w)}{\max_{W,N} dis(s_w,n)+\max_{N,P} dis(n,p)+\max_{P,W} dis(p,d_w)} + C_u \cdot \frac{\sum_{b\in B}\upsilon_b(\phi_b(t)+f(n,b)-f(b,p))}{\max_{b\in B}\upsilon_b(\phi_b(t))}$. The first term of the utility function describes the overall worker detours. It is scaled by the upper bound of a worker's total moving distance. The second term represents the expected number of unsatisfied users. It is scaled by the upper bound of the dissatisfaction at a bike station. The hyper-parameter C_u is used to adjust the importance between two terms. With the utility function, the objective of our WAP becomes $\min_{\mathcal{A}} u(t,\mathcal{A})$ instead of just minimizing the overall worker detour distance. Besides updating the objective function, the constraint shown in Eq. (2) is removed since the number of workers is not large enough to satisfy all rebalancing targets. Other constraints remain the same.

Our Two Round Matching (TRM) algorithm is no longer suitable for the extended problem. The main reason is that the expected number of unsatisfied users of a station is correlated with the number of workers assigned to the station. This dependency relation in the matching is usually referred to as externality [34]. The second round of the minimum cost bipartite matching in the TRM can no longer be used because of the dynamic preferences between workers and station pairs.

Inspired by the idea of two-sided exchange-stable matchings [35], we propose to iteratively improve our worker assignments rather than directly match workers and station pairs in the single round. In two-sided exchange-stable matchings, if one side cannot remain unmatched, allowing swapping their assignments to the other side can lead to the exchange stability. In our problem, no worker can remain unmatched. Instead of swapping assignments between workers, we propose to iteratively improve each worker's assignment until the utility function converges. Specifically, the first round of the bipartite matching in the TRM is kept since there is no externality when matching overflow and underflow stations to minimize the overall detour distance. The second stage then starts to match workers with overflow-underflow station pairs. It starts from a random assignment between workers and station pairs. In each following iteration, a worker is chosen if updating its assignment could reduce the utility function value. The algorithm stops when such a worker does not exist. The detailed procedures of the second phase are shown in Algorithm 5. Lines 1 and 2 initialize the assignment. Lines 3-12 iteratively improve the assignment. In line 4, the workers are sorted based on their contribution to the utility function. The contribution of a worker w is quantified by the utility value of $u(t,(w,(n,p)))$ where (n,p) is the station pair assigned to the worker w. For each worker, if assigning it to another station pair could reduce the utility value, we update the assignment. Otherwise, the algorithm converges to a local minimum.

Our proposed algorithm converges in a finite number of iterations. It is because our utility function is finite. The utility function is limited since both the worker detour

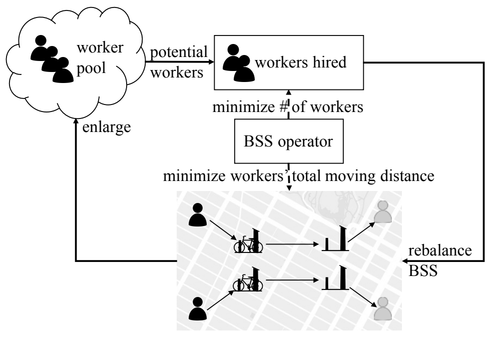


Fig. 8. The positive feedback loop on the worker satisfaction.

distance and the expected number of unsatisfied users are limited. Besides, in each iteration, the number of workers and the number of station pairs are finite as well. Our algorithm would reduce the utility function value after each iteration. Therefore, our algorithm will converge in a finite number of iterations.

8 Worker Satisfaction Analysis

8.1 Reinforcement on the Worker Satisfaction

There exists a possible reinforcement on worker satisfaction within multiple time slices. The reinforcement of worker satisfaction makes it easier to hire enough workers to participate in the rebalancing. Generally, the TRM algorithm aims to minimize the total detour cost of workers in each time slice. It has the potential to improve the worker satisfaction and attract more workers to join. With more penitential workers that could be chosen in the spatial domain assignment, the performance of TRM can also be improved, which further improve the worker satisfaction.

The reinforcement on the worker satisfactions is shown in Fig. 8. If the system can reduce the workers' detour as much as possible, then their cost is reduced and they could earn more via the rebalancing operation. We assume more workers are willing to join the system (enlarge the worker pool) if their satisfaction is improved. If more workers are willing to get involved in the system, the performance of matching could be further improved, which again improves the worker satisfaction. Specifically, the performance of the second stage (as shown in Algorithm 2) of the TRM would increase if the worker pool is enlarged. It is because a larger worker set W provides more choices for the min-cost perfect matching that is used in the second stage. Assume that the worker set W is enlarged to $W \cup \Delta W$, where $|\Delta W| > 0$. The assignment found based on the enlarged worker set $W \cup \Delta W$ is at least as good as the assignment found based on W. This positive feedback loop helps improve the robustness of the rebalancing system and support our assumption that the operator could find sufficient workers to conduct rebalancing. Therefore, minimizing workers' detour not only improves the profits of workers but also improves the stability of the rebalancing system.

8.2 Quantification of the Worker Satisfaction

We then quantify the worker satisfaction and its positive feedback property. The satisfaction could be related to both incentive price and detour distance. The design of the incentive price has been invested by [36]. We follow the OPT-FIX mechanism proposed in [36], where the incentive price is constant. Since we consider that the incentive for workers is constant, the worker satisfaction can be determined by the overall worker detour in a time slice. Specifically, we assume the satisfaction is determined by the relative detour distance, which is defined as the ratio between the detour distance (or extra distance) and the original distance. Recall that the detour distance of worker w is defined as $\delta_w = dis(s_w, n) + dis(n, p) + dis(p, d_w) - dis(s_w, d_w)$. Let $\Delta l = \sum_w \delta_w$ denote the overall worker detour distance. Considering that the overall detour distance is limited in real-world applications, we define $l_c = C$ to denote the threshold. The threshold C represents the maximum detour that can be accepted by workers. Above all, the worker satisfaction z can be defined by the following equations:

$$z = \begin{cases} 1 - \frac{\Delta l}{l_c} = 1 - \frac{l' - l}{C}, & \text{if } l' - l < l_c \\ 0, & \text{Otherwise} \end{cases}$$

Specifically, the worker satisfaction is negatively correlated with the detour distance. For calculation's convenience, we normalize the detour distance Δl based on the threshold l_c. Then the worker satisfaction is kept between 0 and 1 by setting $z = 1 - \Delta l / l_c$, i.e, a zero detour is mapped to $z = 1$ and the worker satisfaction is 0 if the overall detour equals or exceeds l_c. This definition directly calculates the worker satisfaction based on the total detour of all workers.

Then we quantify the worker density and the worker satisfaction. Intuitively, the worker density is positively correlated with the worker satisfaction. In addition, the worker density should have both upper and lower limits which are denoted as L and U respectively. The density of workers is varied between $[L, U]$. We set L as the maximum number of total rebalance demands and $U = kL$. The meaning is that we assume the system can at least find L workers to perform rebalancing. If there are not sufficient workers, the system can increase the incentive strength. The upper bound U is set as kL. It means even if the worker satisfaction is 1, there are at most kLworkers who are willing to help. When the worker satisfaction lies within $[0, 1]$, the worker density is modeled by the following equation:

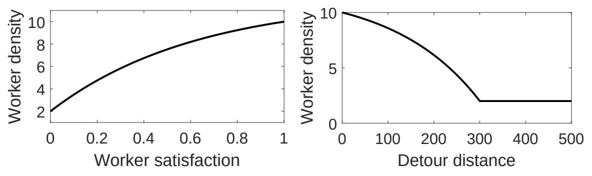
$$(L + U) - (\frac{L}{U})^z \cdot U$$

The equation is illustrated in Fig. 9.

Based on these steps of quantification, we evaluate the variation of worker satisfaction and the results are shown in the experiment. Even starting with a relatively low worker satisfaction, the existence of the possible feedback loop could improve the worker satisfaction and maintain the worker satisfaction to a high level. It shows that our scheme is stable when multiple time slices are taken into account.

8.3 Worker Densities and Weather Conditions

The worker density also correlated with weather conditions. In extreme weather conditions, the worker density would be significantly lower than that in normal weather conditions. Few prior works have been done on evaluating the impact



(a) The worker density verses satisfaction (b) The worker density verses distance

Fig. 9. An illustration of the satisfaction and density function

of weather conditions on the worker density since the lack of data. In this paper, we assume the number of available workers is proportional to the number of users to the system. Weather conditions would impact the number of users to the system. It affects the number of available workers to the same degree. We introduce a factor C_w to describe the effect. Formally, $|W| = C_w|W|_{\text{ave}}$, where $|W|_{\text{ave}}$ is the average number of available workers of a day. We use the regression model to estimate the value of C_w based on the weather impact on the number of system users. In the regression model, we use x_i to denote the weather condition vector of the day i. It includes temperature, heating/cooling degree days, precipitation, and snow depth. Let y_i denote the ratio between the number of users of the day i and the average number of users of a day in history. Running the regression model on the dataset consists of (x_i, y_i) quantifies the weather impact on the number of workers. When calculating the size of the worker pool, the C_w is calculated based on the regression model and is multiplied to the worker density $(L + U) - (\frac{L}{U})^z \cdot U$. The fluctuation of the worker pool size further impacts worker satisfaction in the following time slots. In this way, we introduce the weather impact into the feedback loop on worker satisfaction.

9 Rebalancing for Dockless BSSs

Our rebalancing scheme is designed for docked BSSs, and it could be extended to dockless BSSs. Dong et. al. [37] have built the bridge between docked and dockless BSSs by using clustering techniques. Specifically, aggregation areas of dockless bikes are detected and a flow model is proposed to describe the bike mobility. In this section, we investigate the problem of rebalancing dockless BSSs by converting it to docked BSS scenario.

Inspired by their cluster approach, we define virtual stations in the dockless BSS. A virtual station represents an aggregation area of bikes. As illustrated in Fig. 10, a virtual station consists of a cluster centroid location and a cluster radius. Formally, let V denote the set of centroid locations of virtual stations, $v_p \in V$ denote the location of an overflow virtual station, and $v_n \in V$ denote the location of an underflow virtual station. The radius of a virtual station v is denoted as r_v. For example, the radius of an overflow station v_p is r_p. In addition, each virtual station has its state $\phi_v(t)$ and demand $\tau_v(t)$. The state $\phi_v(t)$ represents the stock level of the virtual station v at the beginning of time slice t. The stock level is the number of bikes whose distance to the cluster centroid is less than the cluster radius. The demand $\tau_v(t)$ is the number of bikes rented or returned at the range of the virtual station v during time slice t. The bike demand

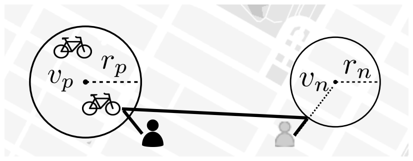


Fig. 10. An illustration of virtual bike stations.

of each virtual station could be predicted by the flow model proposed in [37]. Although there is no actual bike dock in a virtual station, each virtual station v has a capacity η_v. In our scheme, we assume the capacity is proportional to the size of the virtual station. Formally, we assume that $\eta_v = \gamma \cdot r_v^2$, where γ is a constant parameter. Through virtual stations, the problem of rebalancing dockless BSSs is converted to rebalancing docked BSSs.

Our dockless rebalancing scheme is built upon the virtual stations. Based on the state, demand, and capacity of each virtual station, our kGA and GLA could be applied to generate the rebalancing configuration $\boldsymbol{\rho}_V(t)$ for virtual stations in each time slice t, i.e. the number of bikes that should be moved to/from the station. Then, the system operator could recruit workers at the beginning of each time slice. The assignment of workers could be generated by TRM. When applying TRM, we use the cluster centroids as the virtual stations' locations in matching. Note that when workers perform rebalancing, they do not need to pick up or drop off bikes at the cluster centroid. In contrast, a worker could rent the nearest bike that is located in the range of the virtual station assigned to him/her. Similarly, a worker could return the bike to the location that is nearest to his/her destination in the range of the assigned return station. The nearest return location could be found by projecting the worker's destination to the boundary of the virtual underflow station. Fig. 10 illustrates the actual pick up and drop off locations. The difference between the pick-up (drop-off) distance used in matching and the actual pick-up (drop-off) distance is bounded by the radius of the virtual overflow (underflow) station.

10 Experiment

10.1 Real-world dataset

We use the public data of NYC Citi Bike[2] to construct our the NYC dataset. We use a set of history trip data from 8/1/2017 to 9/30/2017. The data contains the records of each trip including trip duration (in seconds), trip start/stop time, start/end station ID, and latitude/longitude, etc. Our NYC dataset contains more than 1.5 million trip records and 328 bike stations.

Besides the NYC Citibike dataset, we also use the SF Bay Area dataset[3], or the simply SF dataset, to test our proposed algorithm, and use the Mobike Shanghai dataset to test our extended scheme for dockless BSSs. The SF dataset contains the bike trip information and station status information of a BSS in the San Francisco Bay Area. It contains 83 bike stations and more than 800 bikes in the system. The Mobike

2. https://www.citibikenyc.com/system-data
3. https://www.kaggle.com/benhamner/sf-bay-area-bike-share

Shanghai dataset contains 102,362 bike trace records in Shanghai. Fig. 11 illustrates some bike locations and virtual stations. The trace data includes the user start/end time and location, which is sufficient for our experiment.

The state information of bike stations in the NYC and SF datasets could be calculated based on the trace data. The trace history records the IDs, locations, and the start and end times. The number of bikes rented/returned at a station b during a given time duration $[t_1, t_2)$ could be calculated by counting the number of data entries whose start/end station ID is b and the start/end time is within range $[t_1, t_2)$. We only need to note that the count of return events is positive and the count of rent events should become negative.

In the NYC and SF datasets, the bike demands of stations during any time slice could be generated by using the prediction algorithm [3]. The demands of virtual stations in the Mobike Shanghai dataset could be determined by counting the average number of rent/return events in the corresponding cluster during a time slice. With rebalancing demands in each time slice, the rebalancing targets could be determined by kGA or GLA.

We use the weather data of NYC[4] to evaluate the weather impact on the number of available workers. The attributes of the NYC weather dataset includes temperature, heating/cooling degree days, precipitation, and snow depth. Some extreme weather conditions, e.g., the temperature is too high, would reduce the number of available workers. The shortage of available workers further brings a negative impact on performances of rebalancing algorithms, which is evaluated in Subsection 10.5 along with our extended algorithms. We find that the temperature significantly affects the number of trips. The effect of snow is not statistically significant for the number of trips. There is the most number of trips when the temperatures are in the range of $21.5°\text{C} - 31.5°\text{C}$. The number of trips decreases when temperatures decrease lower than $21.5°\text{C}$. It also decreases when temperatures increase higher than $31.5°\text{C}$.

10.2 Synthetic Dataset

The synthetic dataset is used as a supplement for the real-world dataset. The location of each bike station in the NYC dataset is static, and so is the density of bike stations. We want to test how the density of stations affects the performance of TRM, since the station density of BSSs in different cities may vary significantly. Therefore, we build a synthetic dataset which contains several station sets with different densities.

In our synthetic dataset, we randomly generate the locations of bike stations following a uniform distribution. The source and destination location distributions of workers are also uniform. The density of stations is measured by the expected number of stations in a $5 \times 5\ \text{km}^2$ square. The capacity of each station is set as 20 and the initial inventory of each station is set as 3/4 of the capacity, which is 15. The number of rent and return events of stations in each time slice is generated by the Poisson process with parameter $\lambda = 7$ which is the average number of daily rent events in the NYC dataset.

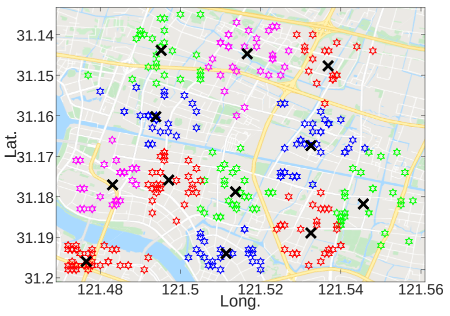


Fig. 11. Virtual stations in the Mobike dataset.

10.3 Comparison algorithms

The *Branch-and-Bound* (BB) algorithm is a global optimization method which is used to find the optimal solution to the WAP problem. Although it cannot be used in realistic scenarios, it can provide an optimal result when the input size is small. The optimal result can be used to evaluate the performance of our two-round matching algorithm. The complexity of calculating this lower-bound heuristic is factorial. Therefore, the comparison is conducted with a small input size.

The *Local Search* (LS) algorithm is a local optimization approach. Finding the weighted 3D matching can be seen as finding the minimum weighted subfamily of pairwise disjoint sets and the size of each set is equal to 3. To the best of our knowledge, the approximation algorithm with the tightest bound for the problem is proposed by Berman in [30]. The approximation ratio of the local search algorithm is $(k+1+\epsilon)/2$ and the time complexity is $m^{O(k)}$, where m is the number of nodes in the intersection graph. In our problem $k = 3$, and $m = |W|^3$. Therefore, the approximation ratio of the LS is $2 + \epsilon$ with time complexity $O(|W|^9)$.

In addition, a *Greedy* algorithm is used as a baseline approach. In the greedy algorithm, each worker chooses the nearest station in N to rent a bike, and returns it to the station in P that is nearest to his/her destination. Workers sequentially make their decisions in an arbitrary order, i.e. the order is randomly shuffled. After a worker made his/her decision, the worker and the chosen stations are deleted.

10.4 Experiment Settings

10.4.1 Performance Comparison

The code used in our experiment is available online[5]. In the synthetic dataset, we separately examine the performance of our algorithms for spatial and temporal optimization in the synthetic dataset. In spatial domain, we compare TRM with BB, LS and Greedy in terms of overall worker moving distance in a time slice. In temporal domain, we compare kGA with GLA in terms of number of workers needed. Besides, we test the performance of TRM on different station densities. Totally, we choose three densities, 10, 20 and 40, to represent sparsely, regularly, and densely distributed stations, respectively. Notice that the BB algorithm is extremely time-consuming. Therefore, when comparing with BB, we extract a subset of stations and choose a short slice length

4. https://w2.weather.gov/climate/xmacis.php?wfo=okx

5. https://github.com/YubinDuan/BikeRebalancing.git

(20 min) to decrease the number of bikes that need to be moved. The stations are randomly extracted from the NYC dataset in a $2 \times 2\ \mathrm{km}^2$ area. For generalization, we repeat the experiment 100 times and record the average results. These results are shown in Fig. 12 and Fig. 13.

In the real-world dataset, we combine optimization algorithms in spatial and temporal domains and show the overall performance of our scheme. The overall performance is measured by the overall moving distance of workers recruited in one day (24 hours). In the experiment, we vary the number of stations and the stations are randomly extracted from the real-world dataset. Fig. 14 shows the corresponding results.

Our extension for dockless BSSs is tested over the Mobike Shanghai dataset. The k-means algorithm is used to generate virtual locations. We test our scheme with different numbers of clusters in terms of overall moving distance. Besides, we test the impact of the capacities of virtual stations by using different γ. Fig. 15 illustrates the results.

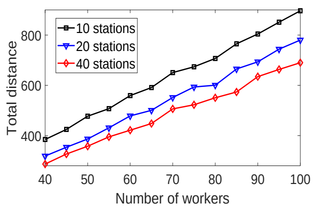
10.4.2 *Worker Shortage and Cost-Efficiency*

We also evaluate our extended algorithm for the case where the number of workers is not sufficient on the NYC dataset. The performance is measured by the overall utility function value. We evaluate both short-term and long-term performances of our extended algorithm (WSME as shown in Algorithm 5) with $C_u = 1$. In the short-term experiment, we consider the bike rebalancing of one day. The rebalancing target is generated by 1GA. For each time slot, the number of available workers is ζ% of the number of bikes to be moved. The extended algorithm is compared with TRM and greedy algorithms. In comparison, we vary the value of ζ. In the long-term experiment, we consider the rebalancing of multiple days. The number of workers in each time slot is generated by our worker satisfaction model, and the impact of weather conditions are considered. We use weather data from 8/7/17-8/11/17. The initial number of workers is set to 30% of the number of required workers. We compare the performance of our extended algorithm with TRM. The experiment results are shown in Fig. 16.

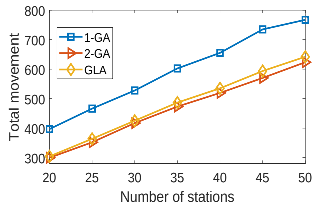
In addition, we examine the cost-efficiency of our scheme over both NYC and SF datasets. The incentive price is set based on the 'bike angel' project, a real-world user-based rebalancing project launched in the NYC Citi bike system. The project rewards users with points if they rent bikes from or return them to specific stations. The points could be redeemed for free trips or gift cards. According to their reward policy[6], each trip could earn at most 3 points. Each point is approximately worth \$0.1. The cost of a truck-based rebalancing scheme including driver's payment and the fuel cost. In the state-of-the-art truck-based rebalancing scheme which aims to minimize detour distances [3], a city is divided into multiple clusters and each cluster has a truck. For fare comparison, we also cluster the stations and perform the comparison in a cluster with 48 stations. We use the same setting in the SF dataset. The only difference is that the cluster size is 35 stations. We show the results on Fig. 17.

We conduct a Monte Carlo simulation to examine our assumption on worker satisfaction in the synthetic dataset.

6. https://www.citibikenyc.com/bikeangels/rewards

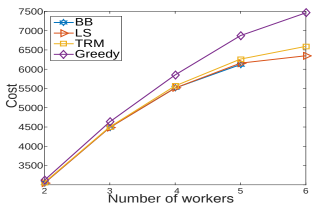


(a) Different station densities

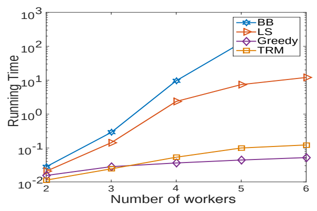


(b) Comparing GLA with kGA

Fig. 12. Performance comparison in the synthetic dataset.



(a) Comparison on distance



(b) Comparison on running time

Fig. 13. Comparison between TRM and existing algorithms.

The worker sources and destinations are uniformly distributed. We further generate rebalancing demands for 15 time slots. The number of workers in each time-slot is calculated based on the worker satisfaction of the previous time-slot. The initial worker satisfaction is set to 0. The initial number of workers is set as the rebalancing demands of the first time-slot, which is $L = 50$ in the experiment. The number of workers of the following time slices are calculated based on the predefined equations. When worker satisfaction is high, there will be more workers who are willing to join in the rebalancing. The increase rate of the number of new workers with the satisfaction is determined by the maximum number of workers U, which represents the upper bound of worker amounts in the map. We test the performance with different value of U. The evolution of the worker satisfaction is shown in Fig. 18.

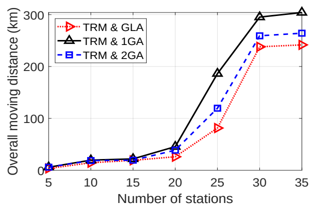
We have argued that our scheme could have better performance if there are more available workers. We test our argument over both NYC and SF datasets. We also test the greedy algorithm and investigate whether more available workers could improve its performance. We vary the ratio between number of available workers and the number of bikes to be moved in each time slice, and record the overall worker moving distances. The results are shown in Fig. 19.

10.5 Evaluation Results and Analyses

10.5.1 *Performance Analyses*

Firstly, we present the evaluation results of TRM with different densities in Fig.12(a). Each line represents a unique station density. From the simulation result, and we can find that if the number of workers is fixed, a larger station density costs a smaller total move distance. This is easy to explain because higher station density may lead to smaller spatial intervals between stations, and thus leads to a smaller total moving distance.

Fig. 12(b) illustrates the comparison between GLA and kGA algorithms in the synthetic dataset, we can find out that the GLA outperforms 1GA and 2GA. Although we find


(a) Overall moving distance

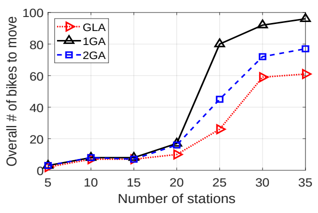
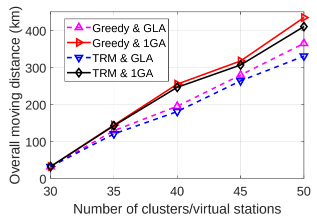

(b) Comparing GLA with kGA

Fig. 14. Performance comparison on the SF dataset.


(a) γ=0.02

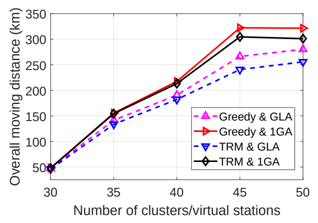

(b) γ=0.04

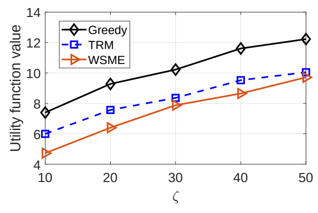
Fig. 15. Performance comparison on the Mobike dataset.

an example where 1GA outperforms GLA, GLA has better performance in general cases. It is not surprising since the GLA can adjust the time slice to look ahead automatically and it is more flexible than kGA. In the experiment, we also track the k chosen by GLA in each iteration and find that the k value barely exceeds 4, which gives a clue for deciding the rebalancing frequency.

The comparison between TRM and other algorithms in the synthetic dataset is shown in Fig. 13. We can conclude that the performances of TRM and LS are similar. They both outperform the greedy algorithm and are not far from the optimal solution. However, considering the running time in Fig. 13(b), we can conclude that the running time of LS is larger than that of TRM. If the problem size is larger, applying the LS to solve the WAP is no longer appropriate since it costs more than one day to provide the solution. Although TRM slightly underperforms compared to LS, TRM is more efficient in terms of its running time.

The result of the experiment on our scheme over the SF dataset is shown in Fig. 14. Fig. 14(a) illustrates the overall moving distances of workers in a day. The rebalancing target is set by either GLA or kGA. The rebalancing assignment is determined by our TRM algorithm. From the result we can find that using TRM with GLA always outperforms other algorithms. It could be explained by the result shown by Fig. 14(b). The number of bikes moved by GLA is less than that generated by 1GA and 2GA. The comparison on the SF dataset illustrates the efficiency of our rebalancing scheme.

Fig. 15 shows the experiment results of our extended scheme on the dockless BSS. From Fig. 15(a), we find that the overall moving distance becomes larger if we set more virtual stations, since the chance for self-balancing decreases when the number of clusters becomes larger and the size of each cluster becomes smaller. Comparing Fig. 15(a) with Fig.15(b) we find that a larger virtual station capacity leads to smaller overall worker movement. The reason is that stations are not likely to overflow if the capacity is larger and the number of bikes to be moved is smaller.


(a) Short-term evaluation

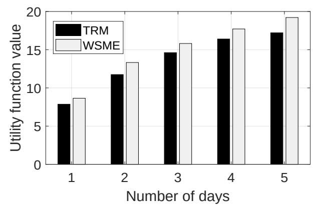

(b) Long-term evaluation

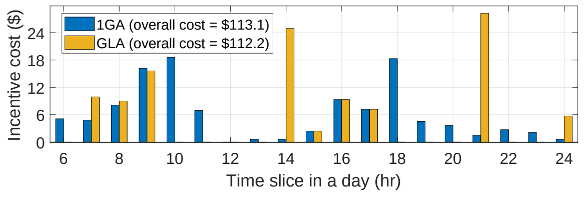
Fig. 16. Evaluation on the NYC dataset with worker shortage.

10.5.2 Impact of Worker Shortage and Cost-Efficiency

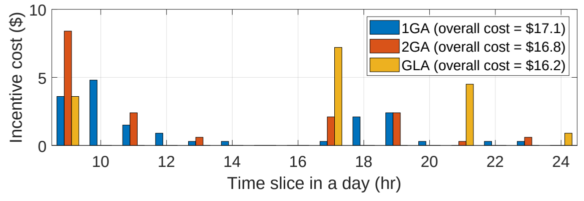
The evaluation results of the worker shortage case are shown in Fig. 16. Fig. 16(a) shows the results of the short-term evaluation, i.e., counting the utility function value for a one-day rebalancing circle. The results show that WSME outperforms both TRM and Greedy. The main reason is that TRM and Greedy only consider the worker detour distance but ignore the expected number of unsatisfied users generated by improper rebalancing operation. The performance gap between WSME and TRM reduces when the percentage of available workers increases. It is because the number of unsatisfied users decreases. When the available workers are sufficient, the utility function value mainly depends on the detour distance. Fig. 16(b) shows the evaluation result of 5 continuous days. The utility function value increases every day since the system is not balanced at the end of the previous day. We can find the utility of TRM increases faster than that of WSME. It shows the long-term effect of the worker shortage. The negative effect of unbalanced stations accumulates with time. During rebalancing, it is necessary to choose stations that could reduce the expected number of unsatisfied users.

We illustrate the case study result of cost efficiency in Fig. 17. The figure shows the cost of rebalancing in each time slot during a day. In the NYC dataset, there is no need to balance before 6 AM. The overall incentive cost is at most \$113.1 when 1GA is used. If the BSS operator chooses to hire trucks to rebalance the system, the cost would be higher. Even only one truck is used, the operator needs to hire at least one driver. The minimum wage in NYC is \$15 per hour. Hiring a full-time driver would cost \$120, assuming the driver works 8 hours per day. This already exceeds the incentive cost, not to mention there are other fees such as fuel, maintaining trucks, etc. In the SF dataset where the demand of each station is low, using our scheme is much more cost-efficient. We find that the overall incentive cost is under \$20 no matter which target configuration algorithm is used. Our case study in NYC and SF shows that our incentive-based scheme is more cost efficient compared with the truck-based rebalancing approaches.

Fig. 18 shows the simulation result on worker satisfaction over the synthetic dataset. From Fig. 18(a), we can find out that the satisfaction increases from 0.12 to around 0.5, and stays at the high level, when U is set as $15L$. It means the maximum number of available workers is $15L$, where L is the rebalancing demand of the first time slot. It shows the effect of the positive feedback loop, which can increase and maintain the worker satisfaction to a higher level. The effect of worker density also can be revealed by

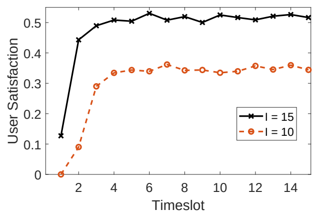


(a) In NYC dataset

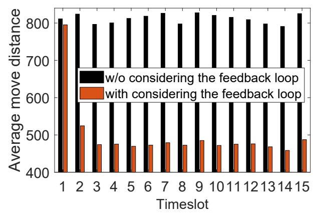


(b) In SF dataset

Fig. 17. The incentive cost of rebalancing.



(a) Worker satisfaction



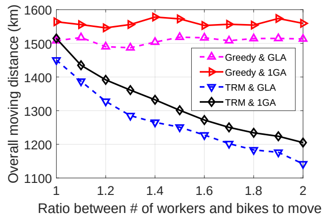
(b) Average move distance

Fig. 18. The evolution of the worker satisfaction and move distance.

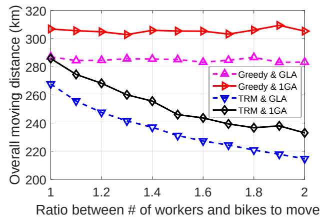
comparing the black line with the red line in Fig. 18(a). The worker satisfaction is higher if the worker density is larger. Fig. 18(b) shows the average move distance of a worker in each time-slot. The red bars show that the average move distance decreases and is kept in a relatively low value even the total demands of all time-slots are the same. It is more clear with comparing with the black bars which show the average move distance without considering the feedback on worker density, i.e. we set a fixed worker density for each time slot. This result directly shows that our scheme could efficiently reduce the move distance of workers and it is not only because of the optimization by the TRM in the spatial domain but also due to the existence of the feedback loop on worker satisfaction over multiple time slices.

From the experiment results, we find out that worker satisfaction increases if the feedback effect is considered. Specifically, without considering the feedback, the worker satisfaction would remain at the initial level, i.e. the satisfaction will be kept at 0.12 for $k = 15$. In addition, we can find out that the satisfaction of workers would be stable eventually. It is reasonable since the number of available workers is limited. No matter how high the incentive price is, there are no more available workers to participate in the rebalancing. As a result, the performance of matching could no longer be improved. Therefore, the satisfaction of workers is kept at a certain value eventually.

Our scheme could achieve better performance with more available workers, it can be examined in real-world datasets as shown in Fig. 19. From results on both datasets, we find that our TRM could achieve better performance if the number of available workers increases. In contrast, the performance of the greedy algorithm does not have an obvious improvement with more workers. The reason might be that the greedy workers have no cooperation at all, and it brings no global profit that more workers are competing with each other. Our scheme is based on global matching and could achieve better coordination among workers.



(a) In NYC dataset



(b) In SF dataset

Fig. 19. Performance evaluation with different numbers of workers.

As a brief summary, the experiment results shows that our scheme is scalable, have great cost efficiency, and could form positive feedback loop worker satisfaction. Although LS slightly outperforms TRM, TRM is much faster than LS and can be applied to real-world BSSs. Compare with truck-based approach, our scheme has lower monetary cost and more flexible. It adaptively reduces cost when the bike usage demand is low. Furthermore, reducing moving distance for workers could improve worker satisfaction, which could further help to reduce the overall moving distance. The existence of the positive feedback loop would attract more workers to participate in the BSS rebalancing.

11 Conclusion

Toward the imbalance bike distribution in BSSs, we propose a rebalancing scheme by recruiting workers. We decouple the problem by slicing the rebalancing in the temporal domain and generate a sequence of slices with a fixed time duration. In each time slice, we formulate the WAP which is NP-hard. When the number of workers is sufficient, a 3-approximation algorithm is proposed to solve WAP in the Euclidean plane. We further extend it to deal with the externality in matching for the worker shortage case. Over multiple time slices, we formulate the CDP and investigate two greedy approaches, namely kGA and GLA. Furthermore, we extend our scheme to dockless BSSs by clustering. We evaluate our rebalancing scheme on both real-world and synthetic datasets. Experiment results show the scalability and reliability of our rebalancing scheme. Although TRM slightly underperforms a local-search approach algorithm in the spatial domain, it runs much faster and can be applied to large-scale real-world BSSs. In the temporal domain, GLA could efficiently reduce the number of workers recruited during rebalancing in the real-world dataset, although we show it may generate unnecessary bike movements in some special cases. Besides, our case studies on NYC and SF show the cost efficiency of our scheme. Furthermore, investigation on the worker satisfaction indicates that our scheme could maintain the satisfaction to a reasonable level which helps to further reduce the overall worker detours.

References

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

[2] E. Fishman, “Bikeshare: A review of recent literature,” *Transport Reviews*, vol. 36, no. 1, pp. 92–113, 2016.

[3] J. Liu, L. Sun, W. Chen, and H. Xiong, “Rebalancing bike sharing systems: A multi-source data smart optimization,” in *Proc. of ACM KDD*, 2016, pp. 1005–1014.

[4] J. Liu, L. Sun, Q. Li, J. Ming, Y. Liu, and H. Xiong, “Functional zone based hierarchical demand prediction for bike system expansion,” in *Proc. of ACM KDD*, 2017, pp. 957–966.

[5] H. Chung, D. Freund, and D. B. Shmoys, “Bike angels: An analysis of citi bike’s incentive program,” in *Proc. of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, 2018, pp. 1–9.

[6] J. Froehlich, J. Neumann, N. Oliver *et al.*, “Sensing and predicting the pulse of the city through shared bicycling.” in *Proc. of IJCAI*, vol. 9, 2009, pp. 1420–1426.

[7] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, and R. Banchs, “Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system,” *Pervasive and Mobile Computing*, vol. 6, no. 4, 2010.

[8] Y. Li, Y. Zheng, H. Zhang, and L. Chen, “Traffic prediction in a bike-sharing system,” in *Proc. of ACM SIGSPATIAL*, 2015.

[9] L. Chen, D. Zhang, L. Wang, D. Yang, X. Ma, S. Li, Z. Wu, G. Pan, T.-M.-T. Nguyen, and J. Jakubowicz, “Dynamic cluster-based over-demand prediction in bike sharing systems,” in *Proc. of ACM UbiComp*, 2016, pp. 841–852.

[10] B. Du, X. Hu, L. Sun, J. Liu, Y. Qiao, and W. Lv, “Traffic demand prediction based on dynamic transition convolutional neural network,” *IEEE Transactions on Intelligent Transportation Systems*, 2020.

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

[12] E. O’Mahony and D. B. Shmoys, “Data analysis and optimization for (citi) bike sharing.” in *Proc. of AAAI*, 2015, pp. 687–694.

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

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

[15] J. Liu, Q. Li, M. Qu, W. Chen, J. Yang, H. Xiong, H. Zhong, and Y. Fu, “Station site optimization in bike sharing systems,” in *Proc. of IEEE ICDM*, 2015, pp. 883–888.

[16] L. Chen, D. Zhang, G. Pan, X. Ma, D. Yang, K. Kushlev, W. Zhang, and S. Li, “Bike sharing station placement leveraging heterogeneous urban open data,” in *ACM UbiComp*, 2015.

[17] J. Bao, T. He, S. Ruan, Y. Li, and Y. Zheng, “Planning bike lanes based on sharing-bikes’ trajectories,” in *Proc. of ACM KDD*, 2017, pp. 1377–1386.

[18] J. W. Yoon, F. Pinelli, and F. Calabrese, “Cityride: a predictive bike sharing journey advisor,” in *Proc. of IEEE MDM*, 2012.

[19] J. Zhang, P. Lu, Z. Li, and J. Gan, “Distributed trip selection game for public bike system with crowdsourcing,” in *Proc. of IEEE INFOCOM*, 2018.

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

[21] D. Chemla, F. Meunier, and R. W. Calvo, “Bike sharing systems: Solving the static rebalancing problem,” *Discrete Optimization*, vol. 10, no. 2, pp. 120–146, 2013.

[22] S. Ghosh, M. Trick, and P. Varakantham, “Robust repositioning to counter unpredictable demand in bike sharing systems,” in *Proc. of IJCAI*, 2016, pp. 3096–3102.

[23] Y. Duan, J. Wu, and H. Zheng, “A greedy approach for vehicle routing when rebalancing bike sharing systems,” in *Proc. of IEEE GLOBECOM*, 2018.

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

[25] L. Pan, Q. Cai, Z. Fang, P. Tang, and L. Huang, “A deep reinforcement learning framework for rebalancing dockless bike sharing systems,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 1393–1400.

[26] Y. Duan and J. Wu, “Optimizing rebalance scheme for dock-less bike sharing systems with adaptive user incentive,” in *Proc. of IEEE MDM*, 2019.

[27] M. Cygan, “Improved approximation for 3-dimensional matching via bounded pathwidth local search,” in *Proc. of IEEE FOCS*. IEEE, 2013, pp. 509–518.

[28] E. M. Arkin and R. Hassin, “On local search for weighted k-set packing,” *Mathematics of Operations Research*, vol. 23, no. 3, pp. 640–648, 1998. [Online]. Available: http://www.jstor.org/stable/3690563

[29] B. Chandra and M. M. Halldórsson, “Greedy local improvement and weighted set packing approximation,” *Journal of Algorithms*, vol. 39, no. 2, pp. 223–240, 2001.

[30] P. Berman, “A d/2 approximation for maximum weight independent set in d-claw free graphs,” in *Scandinavian Workshop on Algorithm Theory*. Springer, 2000, pp. 214–219.

[31] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of computer computations*. Springer, 1972, p. 85.

[32] J. Wu, *Distributed system design*. CRC press, 1998.

[33] T. Raviv, M. Tzur, and I. A. Forma, “Static repositioning in a bike-sharing system: models and solution approaches,” *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 187–229, 2013.

[34] M. Pycia and M. B. Yenmez, “Matching with externalities,” 2019.

[35] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman, “Peer effects and stability in matching markets,” in *International Symposium on Algorithmic Game Theory*. Springer, 2011, pp. 117–129.

[36] Y. Singer, “Budget feasible mechanisms,” in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010, pp. 765–774.

[37] J. Dong, B. Chen, C. Ai, and F. Zhang, “A spatio-temporal flow model of dockless shared bikes,” in *2019 IEEE DSC*, 2019, pp. 312–317.

Yubin Duan received his B.S. degree in Mathematics and Physics from University of Electronic Science and Technology of China, Chengdu, China, in 2017. He is currently a Ph.D. student in the Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania, USA. His current research focuses on urban computing.

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