Transportation Research Part E 201 (2025) 104155

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Repositioning in bike sharing systems with broken bikes considering on-site repairs

Runqiu Hu [a], W.Y. Szeto [a,b,c,*], Sin C. Ho [d]

[a] Department of Civil Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong
[b] The University of Hong Kong Shenzhen Institute of Research and Innovation, Shenzhen, China
[c] Guangdong–Hong Kong–Macau Joint Laboratory for Smart Cities, China
[d] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, China

ARTICLE INFO

Keywords:
Bike repositioning problem
Broken bike repositioning
On-site repair routing
Time-indexed formulation
Station budget constrained heuristic

ABSTRACT

On-site repairs by repairers can handle broken bikes in a bike-sharing system and simultaneously satisfy usable bike demand without requiring vehicle repositioning for off-site repairs. However, this option has not been jointly considered with vehicle-based repositioning for both usable and broken bikes in the literature to improve repositioning performance. This paper presents a mixed-integer linear programming model for a static bike repositioning problem that combines vehicle-based bike delivery/collection with labor-based on-site repairs, aiming to minimize the total cost of user dissatisfaction and carbon emissions within a time budget. A hybrid algorithm, consisting of hybrid genetic search with adaptive diversity control and a station budget constrained heuristic, is proposed to solve the problem. This method introduces a station budget concept to limit the time spent at each station by considering the benefit-cost ratio function of each station along the route. The proposed algorithm is evaluated on networks of various sizes and configurations of the number of trucks and repairers. The results demonstrate that the algorithm can obtain optimal solutions on small instances and can outperform a commercial solver on larger networks in achieving superior solutions within a fraction of the computational time. Moreover, we investigated the cost-effectiveness of introducing repairers, revealing that long repair times and a low percentage of broken bikes diminish the effectiveness of introducing them.

1. Introduction

As an indispensable part of urban public transportation, Bike Sharing Systems (BSSs) provide a sustainable mode of transportation for citizens that can effectively reduce traffic congestion and air pollution. However, the imbalance between demand and supply seriously impedes the smooth operation of a BSS. For instance, after morning peak hours in a station-based BSS, some stations have a shortage of shared bicycles or even no shared bicycles available, while some have a surplus inventory of shared bicycles that leaves nearly no vacant docks at those stations. Consequently, users are often left with the problem of no shared bikes available for renting or no vacant docks available for returning the bikes. To ensure that each station has enough shared bikes or unoccupied docks to meet the customers' rental or return demand, the bikes must be redistributed regularly. The problem of designing effective schemes for shared bicycle redistribution is known as the Bike Repositioning Problem (BRP).

* Corresponding author at: Department of Civil Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong.
E-mail address: ceszeto@hku.hk (W.Y. Szeto).

https://doi.org/10.1016/j.tre.2025.104155
Received 22 April 2024; Received in revised form 17 April 2025; Accepted 22 April 2025
Available online 5 June 2025
1366-5545/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Currently, there are two forms of bicycle repositioning: static and dynamic. For the static repositioning, the operation is carried out throughout the night, assuming that the demand for shared bicycles during that period is negligible. The purpose of this operation is to rearrange the bicycles to prepare for the next day. In contrast, the dynamic repositioning runs throughout the day and adjusts to the time-varying demand, and the planned scheme is altered frequently (Forma et al., 2015). These two forms of repositioning have gained the attention of researchers, and the corresponding static and dynamic BRPs have been well-studied. However, in addition to redistributing shared bikes between stations to meet user demand, the problem arising from broken bikes in BSSs is also non-negligible. For example, in Barcelona, the operator reported that about 12 % of shared bikes were severely damaged, and about 55 % were lightly damaged (Castro Fernández, 2011). In New York, 2 % of the total number of bikes in the BSS need major repairs every day (Kaspi et al., 2016). From users' perspective, if a broken bike is found when renting a bike, time for seeking another usable bike is required. In a worse case, when all the bikes located at the station are broken, the users are required to go to another station. Meanwhile, a user may also encounter all docks being occupied when returning a bike, whereas some parked bikes might be damaged, taking up the space for returning usable bikes. As Kaspi et al. (2016) pointed out, even if the percentage of damaged bikes in a BSS is small, it can still seriously affect user satisfaction with the system. The decrease in user satisfaction and confidence in the system may lead to the abandonment of usage. Therefore, it is necessary to consider handling broken bicycles when solving BRPs.

There have been a few studies on BRPs with broken bikes, some of which focused on broken bikes only and formulated models for the optimal routes for broken bike collection (Zhang et al., 2018; Lu et al., 2019, 2022). The remainder considered both broken and usable bikes and designed the routes of repositioning vehicles and loading instructions for both usable and broken bikes at each station (Alvarez-Valdes et al., 2016; Chang et al., 2018; Wang and Szeto, 2018, 2021; Du et al., 2020; Zhang et al., 2020). All these studies on BRPs with broken bikes assumed that broken bikes were only repaired at depots (representing bike shops or repair centers) and required to be collected by repositioning trucks first. Nevertheless, in practical operations, broken bicycle repairs are not restricted to shops or repair centers. They are also conducted in the field by dispatching repairers to the respective locations. According to Citi Bike's monthly operating report for June 2024 (Citi Bike, 2024), 21,637 unique bikes were checked or repaired by repairers in depots or the field during January. On-site repairs by repairers can handle the broken bikes and satisfy the usable bike demand simultaneously without vehicle repositioning. Therefore, it is essential to determine how to send repairers to the field (i.e., the routes of repairers for on-site repairs) and how many broken bikes should be handled by each repairer at each visited station in addition to the optimal strategy combining usable bike repositioning and broken bike collection. To the best of our knowledge, no studies have made such integration at the time of this writing. Moreover, as discussed above, user dissatisfaction is greatly influenced by the presence of broken bikes. However, most existing studies focus on minimizing the absolute deviations of the inventories from a desired level or maximizing the satisfied demand for bikes (Xu et al., 2019; Chang et al., 2021; Wang and Szeto, 2021; Cai et al., 2022), which does not necessarily lead to the minimized user dissatisfaction. Therefore, it is vital to take user dissatisfaction into account when designing the optimization objectives.

As stated above, regular usable bike rebalancing in a BSS is required. However, frequent usable bike rebalancing can significantly increase the fuel consumption of rebalancing vehicles (Luo et al., 2020), which contradicts the original aim of running a BSS that offers an eco-friendly mode of transportation. For example, in London, due to the intensive repositioning work, the fossil-fueled vehicle usage for rebalancing has outweighed the environmental benefit of launching the BSS as the replacement for car trips (Fishman et al., 2014). Shui and Szeto (2018) pointed out that repositioning plans with long routes or heavy vehicle loads may produce more air pollutants. The problem is more severe if the collection of broken bikes is included, given that the collected broken bikes have to be carried all the way and cannot be unloaded until the vehicles arrive at depots. Hence, it is meaningful to take the environmental factors into account when designing the optimal rebalancing scheme.

In this paper, we propose a static BRP with broken bikes and on-site repairers. The problem is finding an effective scheme for repositioning trucks and repairers. This scheme encompasses the routes and the loading/unloading instructions of broken and usable bikes for the trucks, the routes for the repairers, and the number of broken bikes to be repaired by each repairer at each station. The objective is to minimize the sum of the penalty costs of the total user dissatisfaction at all stations and the cost of the total CO_2 emissions of the trucks. We consider the cost of total CO_2 emissions as the environmental objective, as CO_2 is one of the most severe threats to the environment through the greenhouse effect. Using a time-index formulation approach, a Mixed Integer Linear Programming (MILP) model is proposed for the problem. To solve the problem efficiently, we propose a hybrid algorithm, abbreviated as HGSADC-SBC, encapsulating Hybrid Genetic Search with Adaptive Dynamic Control (HGSADC) and a newly proposed Station Budget Constrained (SBC) heuristic. The algorithmic performance is demonstrated under the scenarios with different network sizes and numbers of repairers and trucks. Moreover, the cost-effectiveness of introducing a repairer under different repairing times and broken bike inventory levels was studied on a 15-station instance.

The contributions of this paper lie in the following:

1. to introduce a new and practical BRP with broken bikes and on-site repairers;
2. to propose a time-indexed MILP model for the problem and solve it;
3. to present new findings and operational insights:

- We find that long repair times and a low proportion of broken bikes lead to the diminishment of the cost-effectiveness of introducing on-site repairers.
- We demonstrate that the cost-effectiveness of introducing on-site repairers changes dynamically with the proportion of broken bikes, indicating the need for adaptable repairer allocation.

The remainder of the paper is organized as follows. Section 2 reviews the literature on BRPs and points out the research gaps.

Section 3 presents a mathematical model for the studied problem. Section 4 introduces the proposed solution algorithm. Section 5 describes the numerical studies and discusses the results. Finally, Section 6 concludes this paper.

2. Literature review

In this section, we first briefly introduce existing research on BRPs. Then, we discuss how current research deals with broken bikes in BRPs. Finally, we summarize the above works and point out the research gaps.

2.1. Categories, objectives, and algorithms in existing studies on BRPs

Existing approaches to dealing with bike demand–supply imbalance can be divided into two categories: user- and operator-based repositioning. User-based methods intend to motivate users to participate in the bike repositioning process by encouraging them to pick up or return bikes at designated stations in exchange for monetary incentives (Singla et al., 2015; Pan et al., 2019; Cheng et al., 2021; Wang and Wang, 2021). While this mechanism can help align supply and demand during system usage, leading to efficient resource allocation, its ability to increase social and economic surplus may be limited. For instance, if the pricing scheme is set too high, the system might achieve perfect balance but fail to generate societal benefits due to reduced participation. Moreover, the effectiveness of this method depends on the willingness of users to participate, which can become highly uncontrollable. As a result, the operator-based rebalancing approaches are adopted more extensively in current BSSs (Singla et al., 2015). In operator-based bike repositioning, vehicles depart from the depot with a load of bikes, pick up and drop off a certain number of bikes at surplus and deficit stations, respectively, and then return to the depot where they started (Lu et al., 2020). Specifically, as mentioned in the introduction, operator-based bike repositioning can be further divided into static repositioning (Raviv et al., 2013; Forma et al., 2015; Liu et al., 2018; Huang et al., 2020; Wang and Szeto, 2021) and dynamic repositioning (Caggiani and Ottomanelli, 2013; O'Mahony and Shmoys, 2015; Caggiani et al., 2018; Brinkmann et al., 2019; Chiariotti et al., 2020; Hu et al., 2021; Chen et al., 2024; Guo et al., 2024). These two forms of operator-based repositioning have been extensively modeled in the BRP literature.

The objectives of existing BRP research were mainly developed according to the critical concerns of the bike-sharing operators, which could be generally categorized into operator- and user-oriented ones. From the operator-oriented perspective, the objectives were set to minimize the operational cost of repositioning, such as travel costs (Dell'Amico et al., 2016) and time costs (Lee et al., 2020). From the user-oriented perspective, the objectives related to user dissatisfaction were optimized, including minimizing the total unmet demand (Shui and Szeto, 2018; Huang et al., 2020; H. Jia et al., 2020), the sum of the absolute deviation of the usable bike inventory after repositioning from the target level at each station (Rainer-Harbach et al., 2015), and the sum of the penalty cost incurred at each station (Ho and Szeto, 2014). Apart from the above prevalent perspectives, some researchers also took environmental influence into account, e.g., total fuel and CO_2 emission costs (Wang and Szeto, 2018), to minimize the impact of fossil-fueled vehicle repositioning. It is worth mentioning that a certain number of studies included more than one of the objectives from above in the problem formulations, which were often handled by the weighted sum approach (Caggiani and Ottomanelli, 2012; Regue and Recker, 2014; Dell'Amico et al., 2018; Shui and Szeto, 2018; You, 2019; Chen and Szeto, 2023; Chen et al., 2024; Guo et al., 2024).

To solve the aforementioned models, different algorithms were designed. Generally, algorithms solving BRPs can be classified into exact and inexact methods. For exact algorithms, Contardo et al. (2012) solved a dynamic repositioning problem for BSSs using column generation combined with Benders decomposition. Caggiani and Ottomanelli (2012) solved the problem of minimizing the sum of the relocation costs and lost user costs with the branch-and-bound algorithm. However, as the scale of the problem grows, exact methods are intractable to provide optimal solutions within an acceptable time. Hence, heuristics, as one class of inexact methods, have become prevalent in solving such problems. For instance, You et al. (2019) adopted a two-phase heuristic, in which the first phase decided on the number of bikes to be picked up and dropped off for all stations, and the second phase decided on the bike redistribution using the repositioning trucks. Apart from heuristics, *meta*-heuristics have also been widely applied for BRPs, such as iterated tabu search (Ho and Szeto, 2014), chemical reaction optimization (Szeto et al., 2016), hybrid large neighborhood search (Ho and Szeto, 2017), artificial bee colony algorithm (Shui and Szeto, 2018; H. Jia et al., 2020; Wang and Szeto, 2021), and genetic algorithm (Lee et al., 2020). Meanwhile, hybrid approaches combining exact and inexact methods have also been put forward to make a tradeoff between accuracy and efficiency, such as hybrid ant-colony optimization with Constraint Programming (CP) (Di Gaspero et al., 2013a), as well as hybrid Large Neighborhood Search (LNS) with CP (Di Gaspero et al., 2013b). In more recent years, the reinforcement learning framework has also been utilized to solve BRPs, particularly in dynamic cases, which can better handle the system's dynamicity and facilitate prompt decision-making (Chen et al., 2024; Guo et al., 2024; Shi et al., 2024).

2.2. Current research on the BRP with broken bikes

As reviewed above, BRPs have been well-studied with different model formulations and solution algorithms. Nevertheless, studies on BRPs considering broken bikes were relatively scarce. The first study was conducted by Kaspi et al. (2016), who proposed a probabilistic model for unusable bicycle detection. Later, a series of similar studies on BRPs with broken bikes came out with different problem characteristics.

In terms of the research scope, some studies focused on the collection of broken bikes only (Zhang et al., 2018; Lu et al., 2019, 2022), and some considered both usable and broken bikes (Alvarez-Valdes et al., 2016; Chang et al., 2018, 2023; Wang and Szeto, 2018, 2021; Du et al., 2020; Zhang et al., 2020; Cai et al., 2022; Jin et al., 2022). Among the above studies, most of the literature dealt with docked BSSs (Alvarez-Valdes et al., 2016; Wang and Szeto, 2018; Zhang et al., 2018, 2020; Du et al., 2020; Cai et al., 2022; Jin et al., 2022), with a few studies considering the dockless BSSs (Chang et al., 2018, 2023; Lu et al., 2019; Usama et al., 2019, 2020). The major difficulty in coping with broken bikes in dockless BSSs is the scatteration of the broken bikes due to the flexibility of parking in dockless BSSs (Usama et al., 2020).

Regarding the objectives of the existing models, a certain number of existing works set the minimization of the travel cost for usable bike repositioning and broken bike collection as the objective (Alvarez-Valdes et al., 2016; Chang et al., 2018; Zhang et al., 2018, 2020; Lu et al., 2019). One thing in common among the aforementioned studies is that they required a perfect balance for usable bikes and a complete collection of broken bikes at each station. However, given limited vehicle capacity, the perfect balance may require that trucks visit the depot and some stations multiple times to meet only a tiny fraction of demand, leading to possible waste of resources or even solution infeasibility, especially under the limited time budget. Therefore, some works allowed incomplete satisfaction by including minimizing the absolute deviations of the usable bikes from the target levels into the objective. Specifically, Usama et al. (2019) formulated user dissatisfaction as the sum of the absolute deviations of inventory levels after rebalancing from the corresponding target inventory levels at all the stations and minimized it. Wang and Szeto (2021) introduced the sum of absolute deviations from the target inventory level of usable bikes at each station and the penalty for the unhandled broken bikes at each station in their objective. Furthermore, some works also took the environmental issue into account and incorporated the emissions of CO_2 into their objectives (Wang and Szeto, 2018, 2021; Chang et al., 2023).

2.3. Research gaps

The existing research on BRPs with broken bikes is summarized in Table 2.1. It can be observed that the aforementioned studies mainly concentrated on obtaining the minimum absolute deviation of the inventory level from the target level at the minimum operational cost. Broken bikes were only regarded as another type of bike in need of collection. However, as mentioned before, some repairing work can be conducted on-site by repairers, who can handle the broken bikes and satisfy the usable bike demand simultaneously without vehicle repositioning. To the best of our knowledge, no study has so far considered the combination of repairers and repositioning vehicles in an optimization model. We noticed one study on repairer scheduling in a BSS (Freund, 2018), which focused on broken dock repairs only.

Regarding the model objective, most researchers focus on minimizing unmet demand for usable bikes and the number of unhandled

Table 2.1
A summary of current studies on BRPs with broken bikes.

Reference	Scenario	Objectives					Assumptions			Repositioning quantities	
		[1]AD/ [2]SD	[3]EUDF	[4]TC/TL/ TT	[5]MC	[6]CE	[7]MD	[8]MV	[9]OR	[10]UB	[11]BB
Kaspi (2015)	Docked, static		✓	✓						[12]DV	DV
Alvarez-Valdes et al. (2016)	Docked, static			✓			✓	✓		[13]PB	PB
Chang et al. (2018)	Dockless, static			✓	✓			✓		[14]NC	PB
Wang and Szeto (2018)	Docked, static					✓	✓	✓		DV	DV
Zhang et al. (2018)	Docked, static			✓				✓		NC	PB
Lu et al. (2019)	Dockless, static			✓				✓		NC	PB
Xu et al. (2019)	Docked, static	✓		✓						DV	DV
Du et al. (2020)	Docked, static			✓			✓	✓		PB	PB
Usama et al. (2020)	Dockless, static			✓	✓					PB	DV
Zhang et al. (2020)	Docked, static			✓				✓		PB	PB
Chang et al. (2021)	Dockless, dynamic	✓		✓	✓		✓			DV	DV
Wang and Szeto (2021)	Docked, static	✓				✓				DV	DV
Cai et al. (2022)	Docked, dynamic	✓						✓		DV	DV
Lu et al. (2022)	Dockless, static			✓						NC	PB
Jin et al. (2022)	Docked, static/dynamic			✓	✓			✓		DV	PB
Chang et al. (2023)	Dockless, dynamic	✓			✓	✓	✓	✓		PB	PB
This study	Docked, static		✓			✓		✓	✓	DV	DV

[1] AD: Absolute Deviation of the number of usable bikes from the target inventory level.
[2] SD: Satisfied Demand.
[3] EUDF: Extended User Dissatisfaction Function
[4] TC/TL/TT: Travel Cost/Travel Length/Travel Time.
[5] MC: Manual-handling Cost.
[6] CE: Carbon Emissions.
[7] MD: Multiple Depots.
[8] MV: Multiple Vehicles.
[9] OR: On-site Repairers.
[10] UB: Usable Bikes.
[11] BB: Broken Bikes.
[12] DV: Decision Variable.
[13] PB: Perfect Balance.
[14] NC: Not Considered.

broken bikes when considering user-oriented objectives. However, this approach assumes that all unmet demand and broken bikes at every station and at any time have an equal impact on user dissatisfaction, which is not always the case in real-world scenarios. For some more frequently used stations during peak hours, user dissatisfaction should be more sensitive to unmet demand and unhandled broken bikes than others, and thus, higher priority should be given to these stations in repositioning. Raviv and Kolka (2013) introduced a User Dissatisfaction Function (UDF) to measure user dissatisfaction at each station as the expected weighted number of users who are unable to rent or return a bicycle during a specific period, given an initial inventory at each station. This measurement is station-based, and it captures both the shortage of bike rent/return demand and the timing of these events. Several subsequent BRP studies also incorporated the function into their optimization models (Raviv et al., 2013; Ho and Szeto, 2014, 2017; Forma et al., 2015; Zhang et al., 2017; Chen and Szeto, 2023). To address the limitation of assuming all the bikes are usable, Kaspi (2016) extended the UDF by considering the presence of broken bikes, creating an Extended User Dissatisfaction Function (EUDF). However, to the best of our knowledge, apart from the study by Kaspi (2015), which proposed a model to minimize the weighted sum of user dissatisfaction and routing costs, no other studies on BRPs with broken bikes have utilized the EUDF. Moreover, no study has simultaneously considered both user dissatisfaction and the cost of total carbon emissions in their models for BRPs with broken bikes, not to mention the corresponding solution algorithm.

Concerning the repositioning quantities, a few studies set the number of broken bikes to be handled at each station as decision variables and minimizing the unhandled broken bikes in the objective (Wang and Szeto, 2018, 2021; Usama et al., 2019; Chang et al., 2023). Nevertheless, none of these studies considered the effect of broken bikes on occupying the docks, which may lead to a potential failure in returning shared bicycles.

This study pioneers in filling these gaps by involving repairer routing and the number of broken bikes assigned for repairs at every visited station in the optimization model, aiming to minimize the sum of the penalty costs of the total user dissatisfaction at all stations in the network, and the cost of the total carbon emissions during the repositioning process. To illustrate the practicality and feasibility of the proposed model, we developed an effective hybrid algorithm called HGSADC-SBC, where HGSADC developed by Vidal et al. (2012, 2013, 2014) serves as the foundation of our method and has been successfully applied to solve a similar bike repositioning problem involving multiple bike types by Li et al. (2016). The SBC heuristic, newly proposed in this study, is based on a novel station budget concept that assigns time at each station by considering the Benefit-Cost Ratio Function (BCRF) for each station in the route. This allows for a more efficient loading, unloading, and repair quantity assignment across stations in the routes. Note that the model proposed by Li et al. (2016) focused on a single vehicle and did not consider repairers, so our method also extends it to handle multiple trucks and repairers.

3. Model formulation

In this paper, we consider a BSS with one depot and a finite number of stations. Every station has a fixed capacity, an initial inventory of usable bikes, and an initial inventory of broken bikes. The numbers of usable bikes and empty docks at each station are assumed to be known as the current technology of BSSs provides the system operator with this information in real-time. Moreover, the number of broken bikes at each station is assumed to be known in this study, as we assume that *users report the existence of broken bikes once they know in the daytime,* and we only consider a nighttime bike repositioning problem.

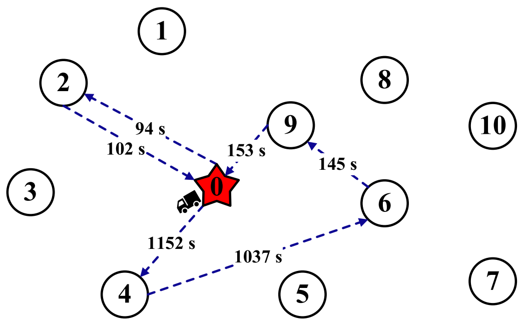
A fleet of homogeneous repositioning trucks with the same capacity depart from the depot, then travel among selected stations in the system to deliver usable bikes to visited stations or pick up the usable bikes from visited stations, collect broken bikes at visited stations, and finally return to the depot to drop off usable and broken bikes within a given time budget. Throughout this process, the trucks can return to the depot to drop off usable and broken bikes when needed, but the number of bikes carried on each truck cannot exceed the truck capacity. Meanwhile, a set of repairers also depart from the depot, head to stations with broken bikes to conduct repairs directly on-site, and return to the depot within the same time budget. Loading, unloading, and repairing each bike take time, with a constant duration for each operation type. For each repositioning truck, every station and the depot can be visited multiple times. For each repairer, each station can be visited at most once. A station can be visited by at most one repairer. We assume the depot holds an infinite inventory of usable bikes and can store an infinite number of usable and broken bikes. Moreover, we assume that the repairers only repair broken bikes at stations but do not repair broken bikes at the depot. Broken bikes at the depot are repaired by the repairing staff there but these bikes have no urgency to be repaired within the repositioning period as we assume we have a sufficient number of usable bikes in the system. Furthermore, we assume that the repairing staff at the depot and the onsite repairers are hired at fixed daily salary rates. Therefore, the total repair staffing cost is constant daily, which is independent of the number of broken bikes repaired onsite or sent back to the depot within the repositioning period. In addition, we assume that the repositioning process involves no rental or returning behaviors from the users, and the final inventories after the repositioning set the starting inventories for the following day. Additionally, we assume the repairers travel on electric tricycles that generate no emissions during the repositioning process, and the travel speed of the electric tricycle is proportional to that of a truck. The objective of the BRP is to minimize the cost consisting of the following components:

(i) The sum of user dissatisfaction at all stations on the following day and;
(ii) The CO_2 emissions of the repositioning trucks during the rebalancing process.

Note that the daily repair staffing cost is not included in our optimization objective as it does not affect the optimal repositioning decisions.

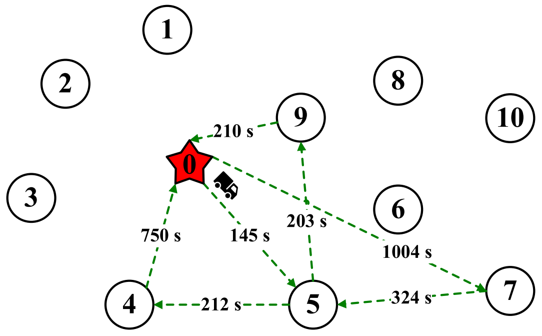
Fig. 3.1 illustrates an example of the problem. The BSS consists of one depot (represented by a red star and labeled with 0) and ten stations (labeled from 1 to 10). The tables in Fig. 3.1 **(a)**-**(d)** show the number of usable and broken bikes loaded onto a truck and unloaded from a truck at each node, and the number of bikes repaired by repairers at each node. Two trucks (Truck 1 in Fig. 3.1 **(a)** and Truck 2 in Fig. 3.1 **(b)**) with the identical capacity of 25 bikes and two repairers (Repairer 1 in Fig. 3.1 **(c)** and Repairer 2 in Fig. 3.1 **(d)**) depart from the depot to visit these stations. Specifically, Truck 1 follows the route 0–4–6–9–0–2–0, with one intermediate return to the depot to drop off 18 broken bikes and pick up three usable bikes. Truck 2 follows the route 0–5–4–0–7–5–9–0, with one intermediate return to the depot to drop off eight usable bikes and 13 broken bikes. Repairer 1 follows the route 0–6–8–0, performing on-site repairs before returning to the depot, while Repairer 2 follows the route 0–1–3–0. Node 5 is visited by Truck 2 twice because the total number of broken and surplus usable bikes to be loaded at the station (26) is more than the truck capacity (25). To reduce vehicle emissions, it is wise to drop off all broken bikes when Truck 2 visits the depot during the repositioning process. The travel times between nodes are labeled on the arcs in the figures, and the travel times between nodes are not assumed to be symmetric (e.g., the back-and-forth travel times for Truck 1 between nodes 0 to 2 are different). Note that Station 10 is not visited by both trucks and repairers.

Fig. 3.2 shows the repositioning and repair process of Truck 1 and Repairer 1 from the example in Fig. 3.1 from a temporal perspective. The repositioning time budget is 7200 s as marked on the time axis. The figure shows the routes taken, inventory variations at the stations, and temporal aspects of the operations (e.g., loading and repairing times, arrival times). Loading or unloading one bike takes 60 s while repairing one broken bike takes 300 s. Note that $(p, b) \rightarrow (p', b')$ at each station represents the variations of usable bike inventory from p to p' and broken bike inventory from b to b' after the operations of trucks or repairers. $< A, B >$ on the axis



Node	0	4	6	9	0	2	0
The number of usable bikes unloaded	0	0	0	7	0	3	0
The number of usable bikes loaded	0	2	5	0	3	0	0
The number of broken bikes unloaded	0	0	0	0	18	0	9
The number of broken bikes loaded	0	1	17	0	0	9	0

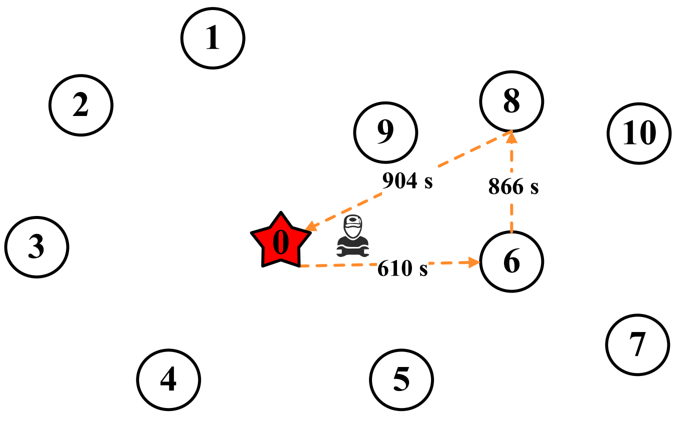
(a) The repositioning route and (un)loading quantities of usable and broken bikes for Truck 1



Node	0	5	4	0	7	5	9	0
The number of usable bikes unloaded	0	0	0	8	0	0	8	0
The number of usable bikes loaded	0	12	0	0	3	1	0	0
The number of broken bikes unloaded	0	0	0	13	0	0	0	5
The number of broken bikes loaded	0	10	3	0	2	3	0	0

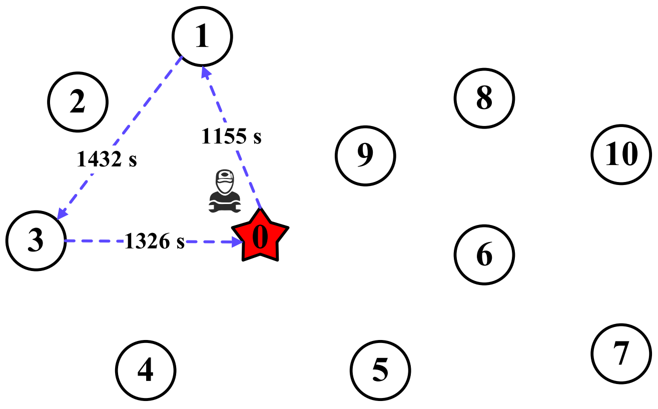
(b) The repositioning route and (un)loading quantities of usable and broken bikes for Truck 2

Fig. 3.1. Problem illustration



Node	0	6	8	0
The number of broken bikes repaired	0	4	12	0

(c) The repairing route and repairing quantities of broken bikes for Repairer 1



Node	0	1	3	0
The number of broken bikes repaired	0	3	2	0

(d) The repairing route and repairing quantities of broken bikes for Repairer 2

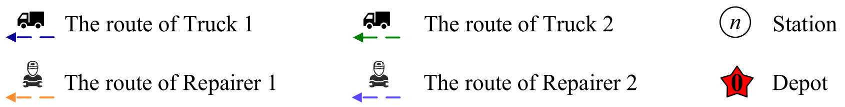


Fig. 3.1. (*continued*).

represents the truck inventory of A usable bikes and B broken bikes when it travels between the nodes. The round-cornered shapes show the variation of the user dissatisfaction under the inventories before and after the operations. Specifically, Truck 1 follows the route 0–4–6–9–0–2–0 as indicated above the time axis. It first visits Station 4 to collect two usable bikes and one broken bike, then travels to Station 6 to collect five usable and 17 broken bikes, reaching its full capacity of 25 bikes. Next, the truck travels to Station 9 to unload the seven usable bikes. Following that, it returns to the depot to unload all broken bikes, loads three usable bikes, and proceeds to Station 2 to deliver the three usable bikes and collect nine broken bikes. Finally, it returns to the depot, unloading all broken bikes and completing its operations in 7123 s, calculated by adding up the arrival time at the depot (6583 s) and the operational time for unloading the nine broken bikes (540 s). Meanwhile, Repairer 1 performs on-site repairs along the route 0–6–8–0, repairing four broken bikes at Station 6 and 12 at Station 8. It concludes its operations in 7180 s. Note that Station 6 is visited by both Truck 1 and Repairer 1, and all 21 broken bikes at the station are handled, with four repaired by the repairer before the truck arrives at the station

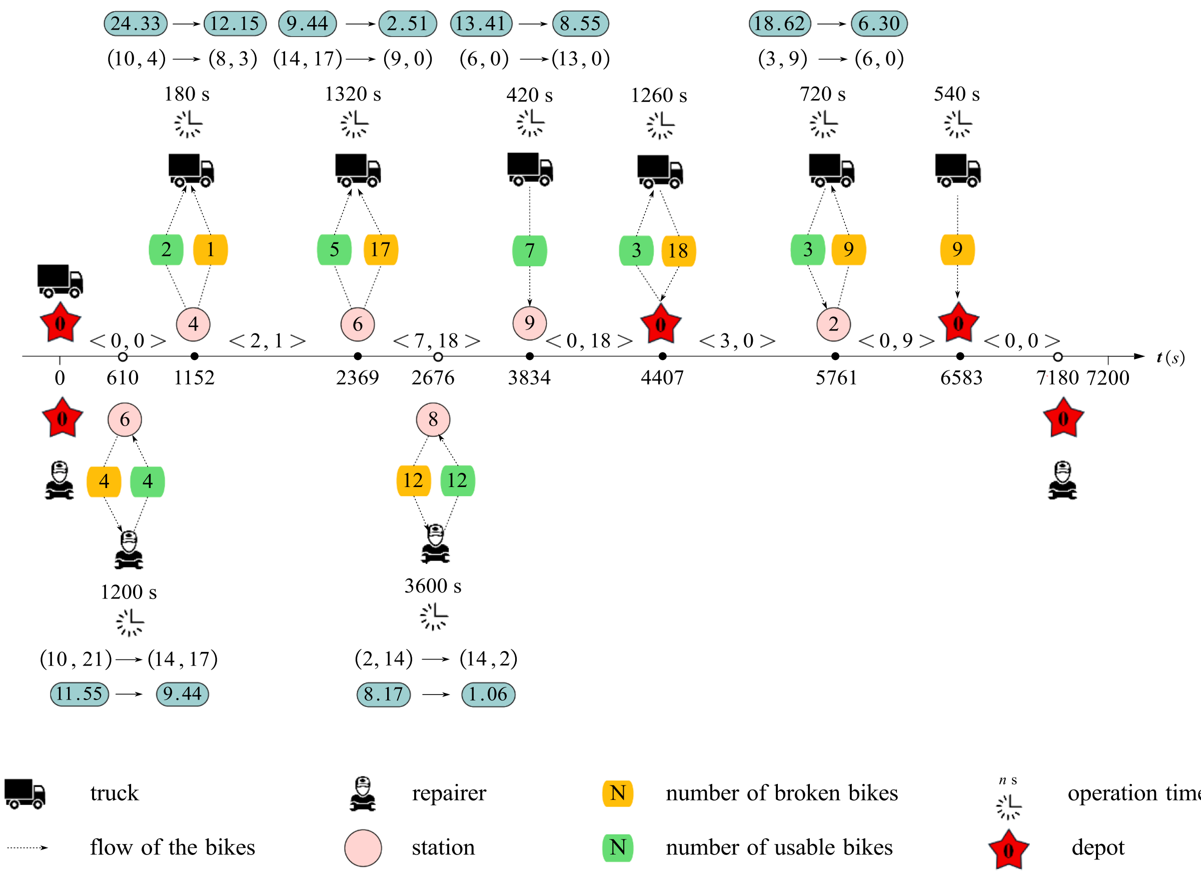


Fig. 3.2. The repositioning and repair process of Truck 1 and Repairer 1.

and 17 loaded by the truck. The four repaired bikes, together with one usable bike originally at Station 6 and two usable bikes at Station 4 are used to partially address the deficit of bikes at Station 9. The remaining deficit of eight usable bikes at Station 9 is satisfied by the usable bikes collected by Truck 2 from other stations and delivered to the station.

The first component of the objective function of the problem involves the calculation of user dissatisfaction throughout the day following the rebalancing process based on the starting inventories resulting from that process. The method proposed by Kaspi et al. (2017) is used for this calculation, which is briefly introduced here. Given station i with C_i docks and a finite period for the following day, at the beginning of that day, there are p_i usable bikes and b_i broken bikes present at the station. Throughout this timeframe, users intending to rent or return a bike arrive at the station based on a stochastic process. When a user's demand for a bike or an available dock is met, the bike is rented or returned, subsequently updating the station's inventory level. Conversely, if the demand cannot be fulfilled, it is assumed that the user promptly leaves the station, corresponding to a failure in renting or returning a bike. The user dissatisfaction at a station is calculated as the weighted expected number of occurrences where users are unable to rent or return bikes during the specified period, provided that the usable and broken bike inventories at the beginning of the period are p_i and b_i, respectively. It is assumed that changes in inventory levels are solely caused by users' rental or return actions, with no external interventions such as truck repositioning or repairs. An Extended User Dissatisfaction Function (EUDF), denoted as $F_i(p_i, b_i)$, is defined to represent the dissatisfaction at station i given the inventories p_i and b_i. The domain of this bivariate function is $\Theta_i = \{(p_i, b_i)|p_i \geq 0, b_i \geq 0, p_i + b_i \leq C_i\}$. Kaspi et al. (2017) proposed a method for determining a convex polyhedral function that has nearly identical values to the EUDF values in its range, enabling the integration of the EUDF into the subsequent linear optimization model. Specifically, for each $(p_i, b_i) \in \Theta_i$, a plane $\alpha^i_{p_i,b_i} \cdot p_i + \beta^i_{p_i,b_i} \cdot b_i + \gamma^i_{p_i,b_i}$ that passes as close as possible to $F_i(p_i, b_i)$ can be fitted, where $\alpha^i_{p_i,b_i}, \beta^i_{p_i,b_i}$, and $\gamma^i_{p_i,b_i}$ are the coefficients of the plane that can be computed for every $(p_i, b_i) \in \theta_i$. The computation details of the coefficients of the fitted plane are given in Appendix A. Due to limited space, we omit the lengthy details on generating the station EUDF values. Interested readers can consult Kaspi et al. (2017) for generating them.

The second component requires the calculation of the CO_2 emissions of a repositioning truck as it moves. Wang and Szeto (2021) used the following emission model to calculate the amount of CO_2 emissions e produced by a truck when it carries bikes over a distance:

$$e = CER \cdot \left(\rho_0 + \frac{\rho^* - \rho_0}{Q}\psi\right) \cdot d \tag{1}$$

where *CER* is the CO_2 emission rate, ρ^* and ρ_0 are the fuel consumption rates when the vehicle is empty and fully loaded, respectively, Q is the vehicle capacity, ψ is the number of bikes carried by the truck, and d is the distance traveled by the truck. Given the similarities between their and our research context, we chose to adopt this emission model for the calculation in the second component.

As aforementioned, each station can be visited by each truck an arbitrary number of times. Hence, the classic arc-indexed formulation is not suitable for our problem. As proposed by Raviv et al. (2013), the time-indexed formulation requires the whole time budget T to be discretized into multiple periods with equal length, which allows us to keep track of each visit to a station and the inventory of bikes on each truck and at each station during every period, thereby enabling the formulation (1) for multiple visits to a station and the depot by vehicles and repairers and (2) to decide the pickup quantities of usable bikes by trucks and the repair quantities of broken bikes by repairs based on the actual inventory status. Therefore, we formulate our problem based on the time-indexed formulation.

3.1. Notations

Sets	
S	The set of stations, indexed by $i = 1, \ldots, \vert S\vert$
S_0	The set of nodes, including the stations and the depot (denoted by 0)
K	The set of rebalancing trucks
R	The set of repairers
Parameters	
p_i^0	The initial inventory of usable bikes at node $i \in S_0$
b_i^0	The initial quantity of broken bikes at node $i \in S_0$
ρ_0	The empty-load fuel consumption rate of each truck (liters/km)
ρ^*	The full-load fuel consumption rate of each truck (liters/km)
CER	The CO_2 emission rate (kg/liter)
Q_k	The capacity of truck k
C_i	The capacity of node $i \in S_0$
τ	The length of a discretized period (second)
t^{load}	Time for loading/unloading a bike ($t^{load} < \tau$)
t^{repair}	Time for repairing a bike on-site ($t^{repair} < \tau$)
T	Rebalancing time budget (second)
t_{ij}^{v}	Travel time for a truck from node i to node j
t_{ij}^{r}	Travel time for a repairer from node i to node j
t_{ij}'	The number of periods required by a truck to travel from node i to node j, which is calculated by $t_{ij}' = \begin{cases} \lceil t_{ij}^{v}/\tau \rceil, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}$
t_{ij}''	The number of periods required by a repairer to travel from node i to node j, which is calculated by $t_{ij}'' = \begin{cases} \lceil t_{ij}^{r}/\tau \rceil, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}$
d_{ij}	The travel distance from node i to node j (km)
α_{p_i,b_i}^{i}	The first coefficient of the fitted plane to approximate the EUDF at station i, given the usable bike inventory level of p_i and the broken bike inventory level of b_i
β_{p_i,b_i}^{i}	The second coefficient of the fitted plane to approximate the EUDF at station i, given the usable bike inventory level of p_i and the broken bike inventory level of b_i
γ_{p_i,b_i}^{i}	The third coefficient of the fitted plane to approximate the EUDF at station i, given the usable bike inventory level of p_i and the broken bike inventory level of b_i
T'	The number of periods in the rebalancing time budget such that $T' = \lfloor T/\tau \rfloor$
c^{p}	The penalty cost for a user unable to rent/return a bike (CNY)
c^{e}	The unit carbon trading cost (CNY/kg)
Decision variables	
$x_{i,j,t,k}^{v}$	If truck k travels directly from node i to node j during period t, $x_{i,j,t,k}^{v} = 1$, 0 otherwise
$x_{i,j,t,m}^{r}$	If repairer m travels directly from node i to node j during period t, $x_{i,j,t,m}^{r} = 1$, 0 otherwise
$p_{i,t}$	The number of usable bikes at node i at the end of period t
$b_{i,t}$	The number of broken bikes at node i at the end of period t
$y_{i,t,k}^{u^+}$	The number of usable bikes delivered to node i by truck k during period t
$y_{i,t,k}^{u^-}$	The number of usable bikes collected from node i by truck k during period t
$y_{i,t,k}^{b^+}$	The number of broken bikes delivered to node i by truck k during period t
$y_{i,t,k}^{b^-}$	The number of broken bikes collected from node i by truck k during period t
$g_{i,t,m}$	The number of broken bikes repaired by repairer m at node i during period t
$\psi_{i,j,t,k}^{u}$	The number of usable bikes on truck k when it departs from node i to node j during period t
$\psi_{i,j,t,k}^{b}$	The number of broken bikes on truck k when it departs from node i to node j during period t
$e_{i,j,t,k}$	The CO_2 emissions generated when truck k travels from node i to node j during period t
∂_k^{v}	The period in which truck k returns to the depot and remains stationary until the rebalancing time budget is used up
∂_m^{r}	The period in which repairer m returns to the depot and remains stationary until the rebalancing time budget is used up

(*continued on next page*)

(*continued*)

$\varsigma^{\mathrm{v}}_{t,k}$	If truck k remains stationary at the depot from period t to the end of the repositioning process, $\varsigma^{\mathrm{v}}_{t,k} = 1$, 0 otherwise
$\varsigma^{\mathrm{r}}_{t,m}$	If repairer m remains stationary at the depot from period t to the end of the repositioning process, $\varsigma^{\mathrm{r}}_{t,m} = 1$, 0 otherwise
$\widehat{F}_i$	The auxiliary variable for the linearization of the EUDF of station i
Function	
$F_i(p_i, b_i)$	The EUDF defined over the usable bike inventory level p_i and broken bike inventory level b_i at station i

3.2. *Time-indexed formulation*

The time-indexed formulation of the BRP separating the routes of trucks and repairers is as follows.

$$\text{minimize } c^{\mathrm{p}}\sum_{i\in S}F_i\left(p_{i,T'}, b_{i,T'}\right) + c^{\mathrm{e}}\sum_{i\in S_0}\sum_{j\in S_0}\sum_{t=t'_{i,j}+1}^{T'}\sum_{k\in K}e_{i,j,t-t'_{i,j},k} + \theta\cdot\left(\begin{array}{c}\sum_{i\in S_0}\sum_{j\in S_0}\sum\limits_{t=t'_{i,j}+1}^{T'}\sum_{k\in K}\left(t^{\mathrm{v}}_{i,j}\cdot x^{\mathrm{v}}_{i,j,t-t'_{i,j},k}\right) + \sum_{i\in S_0}\sum_{t=t'_{i,j}+1}^{T'}\sum_{k\in K}t^{\mathrm{load}}\cdot\left(y^{\mathrm{u}^+}_{i,t,k} + y^{\mathrm{u}^-}_{i,t,k} + y^{\mathrm{b}^+}_{i,t,k} + y^{\mathrm{b}^-}_{i,t,k}\right) + \\ \sum_{i\in S_0}\sum_{j\in S_0}\sum\limits_{t=t''_{i,j}+1}^{T'}\sum_{m\in R}\left(t^{\mathrm{r}}_{i,j}\cdot x^{\mathrm{r}}_{i,j,t-t''_{i,j},m}\right) + \sum_{i\in S_0}\sum_{t=t''_{i,j}+1}^{T'}\sum_{m\in R}t^{\mathrm{repair}}\cdot g_{i,t,m}\end{array}\right) \tag{2}$$

Subject to

$$p_{i,1} = p^0_i - \sum_{k\in K}y^{\mathrm{u}^-}_{i,1,k} + \sum_{k\in K}y^{\mathrm{u}^+}_{i,1,k} + \sum_{m\in R}g_{i,1,m}, \forall i\in S_0, \tag{3}$$

$$p_{i,t} = p_{i,t-1} - \sum_{k\in K}y^{\mathrm{u}^-}_{i,t,k} + \sum_{k\in K}y^{\mathrm{u}^+}_{i,t,k} + \sum_{m\in R}g_{i,t,m}, \forall i\in S_0, t = 2,\dots,T', \tag{4}$$

$$b_{i,1} = b^0_i - \sum_{k\in K}y^{\mathrm{b}^-}_{i,1,k} + \sum_{k\in K}y^{\mathrm{b}^+}_{i,1,k} - \sum_{m\in R}g_{i,1,m}, \forall i\in S_0, \tag{5}$$

$$b_{i,t} = b_{i,t-1} - \sum_{k\in K}y^{\mathrm{b}^-}_{i,t,k} + \sum_{k\in K}y^{\mathrm{b}^+}_{i,t,k} - \sum_{m\in R}g_{i,t,m}, \forall i\in S_0, t = 2,\dots,T', \tag{6}$$

$$p_{i,t} + b_{i,t} \le C_i, \forall i\in S, t = 1,\dots.T', \tag{7}$$

$$\sum_{j\in S_0}\psi^{\mathrm{u}}_{0,j,1,k} = y^{\mathrm{u}^-}_{0,1,k}, \forall k\in K, \tag{8}$$

$$\sum_{j\in S_0}\psi^{\mathrm{u}}_{i,j,t,k} = \sum_{j\in S_0, t-t'_{j,i}>0}\psi^{\mathrm{u}}_{j,i,t-t'_{j,i},k} - y^{\mathrm{u}^+}_{i,t,k} + y^{\mathrm{u}^-}_{i,t,k}, \forall i\in S_0, t = 2,\dots,T', k\in K, \tag{9}$$

$$\psi^{\mathrm{u}}_{i,j,T',k} = 0, \forall i\in S_0, j\in S_0, k\in K, \tag{10}$$

$$\psi^{\mathrm{u}}_{i,j,t,k} \le Q_k x^{\mathrm{v}}_{i,j,t,k}, \forall i\in S_0, j\in S_0, t = 1,\dots,T', k\in K, \tag{11}$$

$$\psi^{\mathrm{b}}_{i,j,1,k} = 0, \forall i\in S_0, j\in S_0, k\in K, \tag{12}$$

$$\sum_{j\in S_0}\psi^{\mathrm{b}}_{i,j,t,k} = \sum_{j\in S_0, t-t'_{j,i}>0}\psi^{\mathrm{b}}_{j,i,t-t'_{j,i},k} - y^{\mathrm{b}^+}_{i,t,k} + y^{\mathrm{b}^-}_{i,t,k}, \forall i\in S_0, t = 2,\dots,T', k\in K, \tag{13}$$

$$\psi^{\mathrm{b}}_{i,j,T',k} = 0, \forall i\in S_0, j\in S_0, k\in K, \tag{14}$$

$$\psi^{\mathrm{b}}_{i,j,t,k} \le Q_k x^{\mathrm{v}}_{i,j,t,k}, \forall i\in S_0, j\in S_0, t = 1,\dots,T', k\in K, \tag{15}$$

$$\psi^{\mathrm{u}}_{i,j,t,k} + \psi^{\mathrm{b}}_{i,j,t,k} \le Q_k, \forall i\in S_0, j\in S_0, t = 1,\dots,T', k\in K, \tag{16}$$

$$y^{\mathrm{u}^+}_{i,t,k} \le \min\{C_i, Q_k\}\cdot\sum_{j\in S_0}x^{\mathrm{v}}_{i,j,t,k}, \forall i\in S_0, t = 1,\dots,T', k\in K, \tag{17}$$

$$y^{\mathrm{u}^-}_{i,t,k} \le \min\{C_i, Q_k\} \cdot \sum_{j\in S_0} x^{\mathrm{v}}_{i,j,t,k}, \forall i \in S_0, t = 1, ..., T\prime, k \in K, \tag{18}$$

$$y^{\mathrm{b}^+}_{0,t,k} \le Q_k \cdot \sum_{j\in S_0} x^{\mathrm{v}}_{0,j,t,k}, \forall t = 1, ..., T\prime, k \in K, \tag{19}$$

$$y^{\mathrm{b}^+}_{i,t,k} = 0, \forall i \in S, t = 1, ..., T\prime, k \in K, \tag{20}$$

$$y^{\mathrm{b}^-}_{0,t,k} = 0, \forall t = 1, ..., T\prime, k \in K, \tag{21}$$

$$y^{\mathrm{b}^-}_{i,t,k} \le \min\{b^0_i, Q_k\} \cdot \sum_{j\in S_0} x^{\mathrm{v}}_{i,j,t,k}, \forall i \in S, t = 1, ..., T\prime, k \in K, \tag{22}$$

$$g_{0,t,m} = 0, \forall t = 1, ..., T\prime, m \in R, \tag{23}$$

$$g_{i,t,m} \le b^0_i \cdot \sum_{j\in S_0} x^{\mathrm{r}}_{i,j,t,m}, \forall i \in S, t = 1, ..., T\prime, m \in R, \tag{24}$$

$$\sum_{t=1}^{T\prime} \sum_{k\in K} y^{\mathrm{b}^-}_{i,t,k} + \sum_{t=1}^{T\prime} \sum_{m\in R} g_{i,t,m} \le b^0_i, \forall i \in S, \tag{25}$$

$$\sum_{j\in S_0} x^{\mathrm{v}}_{0,j,1,k} = 1, \forall k \in K, \tag{26}$$

$$\sum_{j\in S} x^{\mathrm{r}}_{0,j,1,m} = 1, \forall m \in R, \tag{27}$$

$$\sum_{i\in S_0} x^{\mathrm{v}}_{i,0,T' - t'_{i,0},k} = 1, \forall k \in K, \tag{28}$$

$$\sum_{i\in S_0} x^{\mathrm{r}}_{i,0,T' - t''_{i,0},m} = 1, \forall m \in R, \tag{29}$$

$$\sum_{j\in S_0, t-t'_{ji}>0} x^{\mathrm{v}}_{j,i,t-t'_{ji},k} = \sum_{u\in S_0} x^{\mathrm{v}}_{i,u,t,k}, \forall i \in S_0, t = 2, ..., T\prime, k \in K, \tag{30}$$

$$\sum_{j\in S_0, t-t''_{ji}>0} x^{\mathrm{r}}_{j,i,t-t''_{ji},m} = \sum_{u\in S_0} x^{\mathrm{r}}_{i,u,t,m}, \forall i \in S_0, t = 2, ..., T\prime, m \in R, \tag{31}$$

$$\varsigma^{\mathrm{v}}_{t,k} \ge \left(t - \partial^{\mathrm{v}}_k + 1\right) \Big/ T\prime, \forall t = 1, \ldots, T\prime, k \in K, \tag{32}$$

$$\varsigma^{\mathrm{v}}_{t,k} \le \left(t - \partial^{\mathrm{v}}_k + 0.5\right) \Big/ T\prime + 1, \forall t = 1, \ldots, T\prime, k \in K, \tag{33}$$

$$\varsigma^{\mathrm{r}}_{t,m} \ge \left(t - \partial^{\mathrm{r}}_m + 1\right) \Big/ T\prime, \forall t = 1, \ldots, T\prime, m \in R, \tag{34}$$

$$\varsigma^{\mathrm{r}}_{t,m} \le \left(t - \partial^{\mathrm{r}}_m + 0.5\right) \Big/ T\prime + 1, \forall t = 1, \ldots, T\prime, m \in R, \tag{35}$$

$$\sum_{i,j\in S_0} \sum_{t\le w, t>t'_{ij}} t^{\mathrm{v}}_{i,j} \cdot x^{\mathrm{v}}_{i,j,t-t'_{ij},k} + \sum_{i\in S_0} \sum_{t\le w} t^{\mathrm{load}} \left(y^{\mathrm{u}^+}_{i,t,k} + y^{\mathrm{u}^-}_{i,t,k} + y^{\mathrm{b}^+}_{i,t,k} + y^{\mathrm{b}^-}_{i,t,k} \right) \le w \cdot \tau, \forall w = 1, ..., T', k \in K, \tag{36}$$

$$\sum_{i,j\in S_0} \sum_{t\le w, t>0} t^{\mathrm{v}}_{i,j} \cdot x^{\mathrm{v}}_{i,j,t,k} + \sum_{i\in S_0} \sum_{t\le w} t^{\mathrm{load}} \left(y^{\mathrm{u}^+}_{i,t,k} + y^{\mathrm{u}^-}_{i,t,k} + y^{\mathrm{b}^+}_{i,t,k} + y^{\mathrm{b}^-}_{i,t,k} \right) \ge (w-2) \cdot \tau - \varsigma^{\mathrm{v}}_{w,k} T, \forall w = 1, ..., T', k \in K, \tag{37}$$

$$\sum_{i,j\in S_0} \sum_{t\le w, t>t''_{ij}} t^{\mathrm{r}}_{i,j} \cdot x^{\mathrm{r}}_{i,j,t-t''_{ij},m} + \sum_{i\in S} \sum_{t\le w} t^{\mathrm{repair}} g_{i,t,m} \le w \cdot \tau, \forall w = 1, ..., T\prime, m \in R, \tag{38}$$

$$\sum_{i,j\in S_0} \sum_{t\le w, t>0} t^{\mathrm{r}}_{i,j} \cdot x^{\mathrm{r}}_{i,j,t,m} + \sum_{i\in S} \sum_{t\le w} t^{\mathrm{repair}} g_{i,t,m} \ge (w-2) \cdot \tau - \varsigma^{\mathrm{r}}_{w,m} T, \forall w = 1, ..., T\prime, m \in R, \tag{39}$$

$$x^{\mathrm{v}}_{0,0,t,k} \leq 1 - \left(\varsigma^{\mathrm{v}}_{t+1,k} - \varsigma^{\mathrm{v}}_{t,k}\right), \forall t = 1, \ldots, T\prime - 1, k \in K, \tag{40}$$

$$x^{\mathrm{r}}_{0,0,t,m} \leq 1 - \left(\varsigma^{\mathrm{r}}_{t+1,m} - \varsigma^{\mathrm{r}}_{t,m}\right), \forall t = 1, \ldots, T\prime - 1, m \in R, \tag{41}$$

$$x^{\mathrm{v}}_{i,i,t,k} \leq \left(y^{\mathrm{u}^+}_{i,t,k} + y^{\mathrm{u}^-}_{i,t,k} + y^{\mathrm{b}^+}_{i,t,k} + y^{\mathrm{b}^-}_{i,t,k}\right) + \lfloor \tau/_{t^{\mathrm{load}}} \rfloor \varsigma^{\mathrm{v}}_{t,k}, \forall i \in S_0, t = 1, \ldots, T^{'}, k \in K, \tag{42}$$

$$x^{\mathrm{r}}_{i,i,t,m} \leq g_{i,t,m} + \lfloor \tau/_{t^{\mathrm{repair}}} \rfloor \varsigma^{\mathrm{r}}_{t,m}, \forall i \in S_0, t = 1, \ldots, T^{'}, m \in R, \tag{43}$$

$$\sum_{i \in S_0} \sum_{j \in S_0} x^{\mathrm{v}}_{i,j,t,k} \leq 1, \forall t = 1, \ldots, T\prime, k \in K, \tag{44}$$

$$\sum_{i \in S_0} \sum_{j \in S_0} x^{\mathrm{r}}_{i,j,t,m} \leq 1, \forall t = 1, \ldots, T\prime, m \in R, \tag{45}$$

$$\sum_{j \in S_0, j \neq i} \sum_{t = t''_{j,i}+1}^{T\prime} \sum_{m \in R} x^{\mathrm{r}}_{j,i,t-t''_{j,i},m} \leq 1, \forall i \in S_0, \tag{46}$$

$$e_{i,j,t,k} \geq CER \cdot \left[\rho_0 \cdot x^{\mathrm{v}}_{i,j,t,k} + \frac{\rho^* - \rho_0}{Q_k} \cdot \left(\psi^{\mathrm{u}}_{i,j,t,k} + \psi^{\mathrm{b}}_{i,j,t,k}\right)\right] \cdot d_{ij}, \forall i,j \in S_0, t = 1, \ldots, T\prime, k \in K, \tag{47}$$

$$x^{\mathrm{v}}_{i,j,t,k} \in \{0, 1\}, \forall i,j \in S_0, t = 1, \ldots, T\prime, k \in K, \tag{48}$$

$$x^{\mathrm{r}}_{i,j,t,m} \in \{0, 1\}, \forall i,j \in S_0, t = 1, \ldots, T\prime, m \in R, \tag{49}$$

$$\varsigma^{\mathrm{v}}_{t,k} \in \{0, 1\}, \forall t = 1, \ldots, T\prime, k \in K, \tag{50}$$

$$\varsigma^{\mathrm{r}}_{t,m} \in \{0, 1\}, \forall t = 1, \ldots, T\prime, m \in R, \tag{51}$$

$$p_{i,t} \geq 0, \forall i \in S_0, t = 1, \ldots, T\prime, \tag{52}$$

$$b_{i,t} \geq 0, \forall i \in S_0, t = 1, \ldots, T\prime, \tag{53}$$

$$y^{\mathrm{u}^+}_{i,t,k} \geq 0, \text{integer}, \forall i \in S_0, t = 1, \ldots, T\prime, k \in K, \tag{54}$$

$$y^{\mathrm{u}^-}_{i,t,k} \geq 0, \text{integer}, \forall i \in S_0, t = 1, \ldots, T\prime, k \in K, \tag{55}$$

$$y^{\mathrm{b}^+}_{i,t,k} \geq 0, \text{integer}, \forall i \in S_0, t = 1, \ldots, T\prime, k \in K, \tag{56}$$

$$y^{\mathrm{b}^-}_{i,t,k} \geq 0, \text{integer}, \forall i \in S_0, t = 1, \ldots, T\prime, k \in K, \tag{57}$$

$$g_{i,t,m} \geq 0, \text{integer}, \forall i \in S_0, t = 1, \ldots, T\prime, m \in R, \tag{58}$$

$$\partial^{\mathrm{v}}_k \geq 0, \text{integer}, \forall k \in K, \tag{59}$$

$$\partial^{\mathrm{r}}_m \geq 0, \text{integer}, \forall m \in R, \tag{60}$$

$$\partial^{\mathrm{v}}_k \leq T\prime, \text{integer}, \forall k \in K, \tag{61}$$

$$\partial^{\mathrm{r}}_m \leq T\prime, \text{integer}, \forall m \in R, \tag{62}$$

$$\psi^{\mathrm{u}}_{i,j,t,k} \geq 0, \forall i,j \in S_0, t = 1, \ldots, T\prime, k \in K, \tag{63}$$

$$\psi^{\mathrm{b}}_{i,j,t,k} \geq 0, \forall i,j \in S_0, t = 1, \ldots, T\prime, k \in K, \quad \text{and} \tag{64}$$

$$e_{i,j,t,k} \geq 0, \forall i,j \in S_0, t = 1, \ldots, T\prime, k \in K. \tag{65}$$

The objective (2) is to minimize the sum of the penalty cost of the total user dissatisfaction at all stations and the cost of the total CO_2 emissions of all the trucks. Note that we add a third term to penalize the sum of the total travel time of trucks and repairers, bike loading/unloading times for trucks, and repairing times for the repairers. The inclusion of the third term helps discourage non-intuitive or redundant operations that, while feasible, may not be necessary given the available resources and time. For example, as shown in Fig. 3.3 **(a)**, the first case illustrates a truck loading eight usable bikes at a station in period t_0 and then unloading two bikes at the same station in the next period $t_0 + 1$. This solution is feasible and could be optimal when the third term in the objective was not included, but such an operation is redundant as it can be simplified by loading only six bikes at once. The second case in Fig. 3.3 **(b)** shows a repairer repairing one broken bike at one station, traveling to another station to repair one broken bike, and then returning to the original station to repair one additional bike. Again, while feasible, this solution is non-intuitive because the unnecessary return visit to the same station can be avoided. Overall, the third term in the objective penalizes such operations, ensuring a more straightforward and efficient rebalancing process. To avoid introducing unnecessary bias when the experiments involve the scenario without repairers, we set the coefficient θ for this penalty term to be 1e-8. This ensures that the third term does not affect solution optimality. The largest possible value for this term is $(|K| + |R|)T$, and under our experimental setting, the maximum value of the term is 43,200 ($|K| = 2$, $|R| = 2$, and $T = 10800$), which is small enough to avoid any noticeable impact on results that are rounded to two decimal places.

Constraints (3)-(6) keep track of the numbers of usable and broken bikes at each node at the end of each period. Constraints (7) stipulate that the total number of usable and broken bikes at each station is no more than its capacity. Constraints (8) specify the initial load of usable bikes on each truck during the first period. Constraints (9) update the load of usable bikes on each truck at each node during each period. Constraints (10) require that each truck holds no usable bikes at the end of the last period. Constraints (11) ensure that the quantity of usable bikes carried by each truck in any period does not exceed its capacity. Similar constraints for broken bikes are provided in constraints (12)-(15). Constraints (16) ensure that the total number of usable and broken bikes on each truck during each period is no more than its capacity.

Constraints (17) and (18) make sure that the total quantity of usable bikes delivered to or collected from a visited station in a period is less than or equal to the minimum of the station capacity and the truck capacity. Constraints (19) limit that the number of broken bikes that can be deposited at the depot in a period is less than or equal to the truck capacity. Constraints (20) ensure no broken bikes are delivered to any station. Constraints (21) enforce that no broken bikes are collected from the depot. Constraints (22) make sure that the total quantity of broken bikes collected from a visited station in a period is less than or equal to the minimum of the initial broken bike inventory at the station and the truck capacity. Constraints (23) guarantee that no broken bike is repaired by the on-site repairers at the depot. Constraints (24) limit that the number of broken bikes repaired at any station is less than or equal to the initial broken bike inventory at the station. Constraints (25) ensure that the total number of broken bikes collected by trucks and repaired by repairers at each station does not exceed the initial broken bike inventory.

Constraints (26)-(29) require that both trucks and repairers depart from the depot initially and return to the depot eventually. Constraints (30) and (31) are the flow conservation conditions for trucks and repairers, respectively. Constraints (32)-(35) relate $\varsigma^{v}_{t,k}$(the variable used to define whether a truck or a repairer stays still at the depot from period t to the end of the rebalancing process) to ∂^{v}_{k} (the period of the truck's or repairer's final return to the depot).

Constraints (36)-(39) restrict the upper and lower bounds of the operational time, and are modified from the study of Liu et al. (2018). Specifically, constraints (36) imply that up to each period, the sum of route travel time and the service time at each node cannot exceed the time from the start of repositioning to the end of that period, and constraints (37) imply that up to each period, the sum of route travel time and the service time at each node cannot be lower than the time from the start of repositioning to the end of two periods before that period. Similar constraints are also imposed on the repairers in constraints (38) and (39). Note that constraints (37) and (39) are only activated before the period that the truck and the repairer return to the depot for the last time, respectively.

Constraints (40) and (41) ensure the truck or the repairer stays at the depot after the final return. Constraints (42) and (43) limit that before the final return to the depot, if the truck or repairer stays at the same node for another period, it must conduct at least one loading, unloading, or bike repair here. Constraints (44) enforce that a truck can travel between at most one pair of nodes within a period. A similar set of constraints for repairers is given in constraints (45). Constraints (46) ensure that each station can be visited at

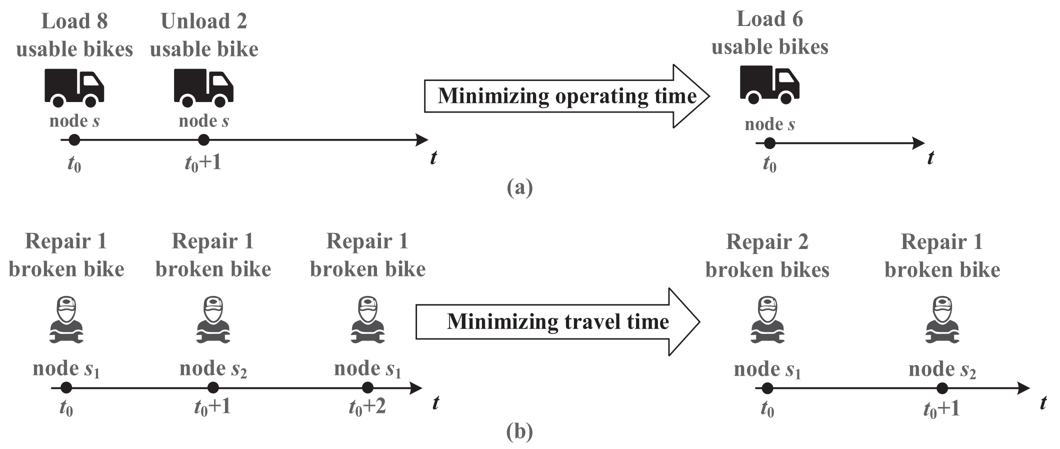


Fig. 3.3. Examples of using the third component in the objective to avoid redundant operations.

most once by at most one repairer in the whole repositioning horizon. Constraints (47) define the carbon emissions of each truck produced in each period when traveling between any two nodes. Constraints (48)-(65) are the domain constraints.

Following Kaspi (2015), the EUDF is linearized by introducing a new variable $\widehat{F}_i$ for each station $i \in S$, and the following linear constraints are added to the formulation.

$$\widehat{F}_i \geq \alpha^i_{p_i,b_i} \cdot p_{i,T^{'}} + \beta^i_{p_i,b_i} \cdot b_{i,T^{'}} + \gamma^i_{p_i,b_i}, \forall (p_i, b_i) \in \left\{ (p_i, b_i) \in \mathbb{Z}^2 \middle| p_i \geq 0, b_i \geq 0, p_i + b_i \leq C_i \right\}, \forall i \in S. \tag{66}$$

The resultant objective is

$$\begin{aligned} &\text{minimize } c^{\mathrm{p}} \sum_{i\in S} \widehat{F}_i + c^{\mathrm{e}} \sum_{i\in S_0} \sum_{j\in S_0} \sum_{t=t^{'}_{i,j}+1}^{T^{'}} \sum_{k\in K} e_{i,j,t-t^{'}_{i,j},k} + \\ &\theta \cdot \left(\begin{array}{c} \sum_{i\in S_0} \sum_{j\in S_0} \sum_{t=t^{'}_{i,j}+1}^{T^{'}} \sum_{k\in K} \left(t^{\mathrm{v}}_{i,j} \cdot x^{\mathrm{v}}_{i,j,t-t^{'}_{i,j},k} \right) + \sum_{i\in S_0} \sum_{t=t^{'}_{i,j}+1}^{T^{'}} \sum_{k\in K} t^{\mathrm{load}} \cdot \left(y^{\mathrm{u}^+}_{i,t,k} + y^{\mathrm{u}^-}_{i,t,k} + y^{\mathrm{b}^+}_{i,t,k} + y^{\mathrm{b}^-}_{i,t,k} \right) \\ \sum_{i\in S_0} \sum_{j\in S_0} \sum_{t=t^{''}_{i,j}+1}^{T^{'}} \sum_{m\in R} \left(t^{\mathrm{r}}_{i,j} \cdot x^{\mathrm{r}}_{i,j,t-t^{''}_{i,j},m} \right) + \sum_{i\in S_0} \sum_{t=t^{''}_{i,j}+1}^{T^{'}} \sum_{m\in R} t^{\mathrm{repair}} \cdot g_{i,t,m} \end{array} \right). \end{aligned} \tag{67}$$

4. The solution algorithm

This study developed a hybrid algorithm (HGSADC-SBC) consisting of Hybrid Genetic Search with Adaptive Diversity Control (HGSADC) and a Station Budget Constrained (SBC) heuristic to solve the proposed model. The HGSADC is the backbone of the algorithm, and the SBC heuristic is encapsulated to determine the quantities of loading, unloading, and repairing of bikes for a given route.

4.1. Solution representation

Each solution is represented by $\phi = (S_{\mathrm{R}}, S_{\mathrm{K}}, \Omega_{\mathrm{R}}, \Omega_{\mathrm{K}})$, where $S_{\mathrm{R}} = \left(r_1, r_2, ..., r_{|R|}\right)$ denotes the list of repairer routes $r_i\,(i \in R)$, and $S_{\mathrm{K}} = \left(r^{'}_1, r^{'}_2, ..., r^{'}_{|K|}\right)$ denotes the list of truck routes $r^{'}_k\,(k \in K)$. The repairing quantities are represented by $\Omega_{\mathrm{R}} = \left(G_1, G_2, ..., G_{|R|}\right)$, with $G_i\,(i \in R)$ specifying the repairing quantities associated with the repairer route $r_i\,(i \in R)$. Similarly, the (un)loading quantities are represented by $\Omega_{\mathrm{K}} = \left(Y_1, Y_2, ..., Y_{|K|}\right)$, where $Y_k\ (k \in K)$ specifies the (un)loading quantities associated with the truck route $r^{'}_k(k \in K)$.

The route of each repairer is represented as $r_i = \left(s_{i_0}, s_{i_1}, s_{i_2}, ..., s_{i_{|r_i|-2}}, s_{i_{|r_i|-1}}\right), i \in R$, and the route of each truck is represented as $r^{'}_k = \left(s^{'}_{k_0}, s^{'}_{k_1}, s^{'}_{k_2}, ..., s^{'}_{k_{\left|r^{'}_k\right|-2}}, s^{'}_{k_{\left|r^{'}_k\right|-1}}\right), k \in K$, where $s_{i_j} \in S_0$ and $s^{'}_{k_l} \in S_0 \ \left(0 \leq j \leq |r_i| - 1, 0 \leq l \leq \left|r^{'}_k\right| - 1\right)$, and the subscripts j and l are used to define the order of nodes being visited. Every route comprises a permutation of visited nodes, with the first and last being the depot, labeled with 0, i.e., $s_{i_0} = s_{i_{|r_i|-1}} = 0\,(i \in R)$ and $s^{'}_{k_0} = s^{'}_{k_{\left|r^{'}_k\right|-1}} = 0\,(k \in K)$. For any of the two routes in S_{R}, no common stations are allowed between the initial and ending depot.

For repairer $i\ (i \in R)$, the repairing quantities associated with route r_i are represented as $G_i = \left(g_{s_{i_0},i}, g_{s_{i_1},i}, ..., g_{s_{i_{|r_i|-1}},i}\right)$, where $g_{s_{i_j},i}$ denotes the number of broken bikes repaired at node s_{i_j} in $r_i\ (0 \leq j \leq |r_i| - 1\,)$. Similarly, for truck $k\ (k \in K)$, the (un)loading quantities associated with route $r^{'}_k$ are denoted by $Y_k = \left(\left(y^{\mathrm{u}^+}_{0,k}, y^{\mathrm{u}^-}_{0,k}, y^{\mathrm{b}^+}_{0,k}, y^{\mathrm{b}^-}_{0,k}\right), \left(y^{\mathrm{u}^+}_{1,k}, y^{\mathrm{u}^-}_{1,k}, y^{\mathrm{b}^+}_{1,k}, y^{\mathrm{b}^-}_{1,k}\right), ..., \left(y^{\mathrm{u}^+}_{\left|r^{'}_k\right|-1,k}, y^{\mathrm{u}^-}_{\left|r^{'}_k\right|-1,k}, y^{\mathrm{b}^+}_{\left|r^{'}_k\right|-1,k}, y^{\mathrm{b}^-}_{\left|r^{'}_k\right|-1,k}\right) \right)$, where $y^{\mathrm{u}^+}_{l,k}$ and $y^{\mathrm{u}^-}_{l,k}$ denote the number of usable bikes unloaded and loaded at node s_{k_l} respectively, and $y^{\mathrm{b}^+}_{l,k}$ and $y^{\mathrm{b}^-}_{l,k}$ represent the number of broken bikes unloaded and loaded at s_{k_l} $\left(0 \leq l \leq \left|r^{'}_k\right| - 1\right)$ respectively.

The solution that corresponds to the one shown in Fig. 3.1 in a network with ten stations and one depot with two repairers and two trucks can be represented as

$$\phi = (S_{\mathrm{R}}, S_{\mathrm{K}}, \Omega_{\mathrm{R}}, \Omega_{\mathrm{K}}) = \left((r_1, r_2), \left(r^{'}_1, r^{'}_2\right), (G_1, G_2), (Y_1, Y_2)\right)$$

where:

- $r_1 = (0, 6, 8, 0)$ and $G_1 = (0, 4, 12, 0)$;
- $r_2 = (0, 1, 3, 0)$ and $G_2 = (0, 3, 2, 0)$;
- $r'_1 = (0, 4, 6, 9, 0, 2, 0)$ and $Y_1 = ((0, 0, 0, 0), (0, 2, 0, 1), (0, 5, 0, 17), (7, 0, 0, 0), (0, 3, 18, 0), (3, 0, 0, 9), (0, 0, 9, 0)$;
- $r'_2 = (0, 5, 4, 0, 7, 5, 9, 0)$ and $Y_2 = ((0, 0, 0, 0), (0, 12, 0, 10), (0, 0, 0, 3), (8, 0, 13, 0), (0, 3, 0, 2), (0, 1, 0, 3), (8, 0, 0, 0), (0, 0, 5, 0))$.

Note that the zeros at the beginning and end of each route are included solely for representation purposes. In the implementation, these zeros are excluded during operations at each step, with only the sequence of the visited nodes in between being retained to enhance computational efficiency.

4.2. Overall algorithmic procedure

The algorithmic procedure for HGSADC-SBC is given in **Algorithm 1**. The algorithm starts with the initialization of a population *Pop* consisting of a feasible subpopulation Pop_f and an infeasible subpopulation Pop_i (Line 1). Then, the population of individuals iteratively evolves through successive operations (Line 3 to Line 29). The best fitness value and its corresponding solution up to each iteration are kept with *current_best_value* and *current_best_sol*, respectively. The algorithm terminates and returns the best known solution and fitness value when the number of iterations without improvement (denoted as *iter_no_imp*) reaches the limit It_{NI}, or the running time (denoted as *cpu_time*) reaches the limit T_{max}. A diversification process is activated every It_{div} iterations without improvement to maintain a high level of diversity among individuals (Line 26 to Line 28).

Algorithm 1: HGSADC-SBC

```
1   Initialization: Generate a population Pop consisting of a feasible subpopulation $Pop_f$ and an infeasible subpopulation $Pop_i$
2   Set $iter\_no\_imp = 0$ and $current\_best\_value = +\infty$
3   while $iter\_no\_imp < It_{NI}$ and $cpu\_time < T_{max}$ do
4   Randomly select two individuals $Ind_1$ and $Ind_2$ from Pop
5   Generate the routes of offspring $C_1$ and $C_2$ using ordered crossover on the routes of $Ind_1$ and $Ind_2$, then apply the SBC heuristic on the routes to determine
    the (un)loading and repairing quantities for $C_1$ and $C_2$
6   Educate on both $C_1$ and $C_2$ by applying local search procedures to generate new routes and using the SBC heuristic to determine the (un)loading and
    repairing quantities
7   if $C_i$ ($i = 1, 2$) is feasible then
8     Add $C_i$ to the feasible subpopulation $Pop_f$
9   else
10    Add $C_i$ to the infeasible subpopulation $Pop_i$ and repair $C_i$ with the probability $P_{repair}$ using local search procedures and the SBC heuristic
11  end if
12  if the individuals become feasible after repairs then
13    The repaired individuals are added to the feasible subpopulation $Pop_f$
14  else
15    The repaired individuals are discarded, and the original infeasible solution is kept in the infeasible subpopulation $Pop_i$
16  end if
17  if the size of $Pop_f$ or $Pop_i$ reaches the maximum subpopulation size then
18    Select survivors from the corresponding subpopulation
19  end if
20  Set $bs$ = the solution with the best fitness value from $Pop_f$ and set $bv$ = the fitness value of $bs$
21  if $bv \geq current\_best\_value$ then
22    Set $iter\_no\_imp = iter\_no\_imp + 1$
23  Else
24    Set $current\_best\_value = bv$, $current\_best\_sol = bs$, and $iter\_no\_imp = 0$
25  end if
26  if $iter\_no\_imp \geq It_{div}$ then
27    Diversify the population consisting of $Pop_f$ and $Pop_i$ and adjust the penalty parameter
28  end if
29  end while
30  return current_best_sol, current_best_value
```

The algorithm structure follows the one given by Vidal et al. (2012, 2013, 2014). The route representation, crossover operator, and education operator are based on the study by Li et al. (2016) with some modifications, mostly on extending their methods to address the scenario of multiple routes for multiple trucks and repairers from the scenario of a single route for only one truck. Moreover, in this algorithm, the crossover and local search operations in education and the repair process are applied to the routes of the solutions. Once new routes are generated through these operations, the SBC heuristic is subsequently applied to determine the loading, unloading, and repair quantities, completing each solution with both routes and associated quantities. The SBC heuristic is developed based on a novel station budget concept, and is detailed in Section 4.4.

4.3. Population initialization

To initialize a population, we generate solutions one by one until a specified population size is reached. The generation of each solution requires initializing routes for both repairers and trucks, followed by applying the SBC heuristic to determine the associated (un)loading and repairing quantities. Each solution then undergoes a feasibility check (as detailed in Section 4.6) and is added to the feasible or infeasible subpopulation based on its feasibility.

To initialize a route, the depot is set as the first node. Then, new nodes are iteratively added to the route based on the last node of the current route, denoted as n_{last}. Specifically, we first find the set of nodes that allow the truck/repairer to reach there from n_{last}, conduct at least one loading/unloading/repairing there, and return to the depot within the time budget. The set is defined as the *reachable list* of n_{last}. Note that n_{last} itself is not included in the set. We randomly choose one node from the *reachable list* and append it to the end of the route. The procedure repeats until the reachable list of n_{last} is empty, and we finally append the depot at the end. Other routes are generated in the same manner. Note that the routes are generated one by one, in the order of repairers first and trucks second to make the best use of the repairers to repair broken bikes before truck arrivals. It should also be noted that any station that has already been visited by a repairer is not considered when generating routes for other repairers.

4.4. Station budget constrained (SBC) heuristic for quantity determination

For each route of a truck or repairer, we have to determine the quantities of bikes loaded, unloaded, and repaired at designated stations to form complete solutions and calculate their fitness values. To determine these quantities, an SBC heuristic is proposed. Some notations and terminologies are defined first to better describe the SBC heuristic:

(1) q_s represents the target usable inventory level at station s ($s \in S$), which is determined as the usable bike inventory at station s that corresponds to the lowest EUDF value. It together with the initial inventory of usable bikes p_s^0 can be used to define the station type. A station is categorized into a deficit station, a surplus station, and a balanced station if $p_s^0 < q_s$, $p_s^0 > q_s$, and $p_s^0 = q_s$ hold, respectively.
(2) p_s and b_s are used to track the inventory of usable and broken bikes at station s ($s \in S$), respectively. At the beginning of the SBC heuristic, they are assigned to the initial inventories before the repositioning starts. During the procedure of the heuristic, they are updated when bikes are loaded or unloaded from the trucks or when repairers conduct repairs.
(3) $\psi_{l,k}^{\text{u}}$ and $\psi_{l,k}^{\text{b}}$ are used to track the inventory of usable and broken bikes on truck k ($k \in K$) when it leaves the l-th visited node $s'_{k_l}\left(l = 0, ..., |r'_k| - 1\right)$.
(4) $maxTime_{s,i}^{\text{r}}$ represents the station repairing budget assigned to station $s \in S$ for repairer $i \in R$.
(5) $maxTime_{s,k}^{\text{t}}$ represents the station (un)loading budget assigned to station $s \in S$ for truck $k \in K$.
(6) $maxTimeRemain_{s,k}^{\text{t}}$ represents the remaining station (un)loading budget assigned to station $s \in S$ for truck $k \in K$.
(7) $unsat_{s,k}^{\text{load,u}}$ represents the number of usable bikes not loaded onto truck k due to the limited station loading budget at station $s \in S$.
(8) $unsat_{s,k}^{\text{unload,u}}$ represents the number of usable bikes not unloaded to station $s \in S$ from truck k due to the limited station unloading budget.
(9) $unsat_{s,k}^{\text{load,b}}$ represents the number of broken bikes not loaded onto truck k due to the limited station loading budget at station $s \in S$.
(10) $maxTimeInit_{s,k}^{\text{t}}$ represents the initial station time budget assigned to station $s \in S$ for truck k before determining truck loading quantities.
(11) $maxTimeNew_{s,k}^{\text{t}}$ represents the updated station time budget assigned to station $s \in S$ for truck k after adjustment.

We point out that the variables presented above are global variables that are shared across all subsequent algorithms, which means any changes to these variables are effective in all algorithms. For the sake of brevity, we directly use and assign values to these variables within the algorithms without redefining them or including them as arguments in each algorithmic procedure. In addition, in the following algorithm description, we use a notation where a variable is enclosed in parentheses and subscripted by its index domains to represent a list of variables indexed by the subscripts. For example, $\left(maxTime_{s,i}^{\text{r}}\right)_{s\in S}$ represents a one-dimensional list that stores the station budget at each station in route r_i for repairer i.

The SBC heuristic procedure is formed by the following steps:

Step 1: Initialization. Set initial values for station inventories.

Step 2: Assign repairing quantities for repairers. For each repairer, conduct the following three subprocedures:

1. Assign a station repair budget to each visited station in the repairer's route.
2. Determine the number of bike repairs to be conducted at each station, constrained by the quantities allowed to be handled under the assigned station budget, the broken bike inventory at the station, and the deviation of the usable bike inventory to the station target.
3. Update the station's usable and broken bike inventories to reflect the completed repairs.

Step 3: First-round truck loading and unloading assignment. For each truck, conduct the following three subprocedures:

1. Allocate a station (un)loading budget for each station in the truck's route.
2. Assign the initial quantities for loading and unloading usable and broken bikes at each station, based on station status (surplus or deficit) and station budget.
3. Update truck and station inventories accordingly.

Step 4: Adjust station budgets based on unused time. Adjust the station (un)loading budgets to redistribute any unused time to stations where demand remains unmet.

Step 5: Second-round truck loading and unloading assignment. Revert station inventories to their values before the first round of loading/unloading assignment, and reassign the quantities for the second round using the adjusted station budgets obtained from **Step 4**.

Step 6: Final amendments to routes and quantities. Make final adjustments to the truck and repairer routes, along with their associated loading, unloading, and repairing quantities, to optimize the use of any time budget left.

For the detailed procedure of the SBC heuristic, readers can refer to **Algorithm 2 in Appendix B**.

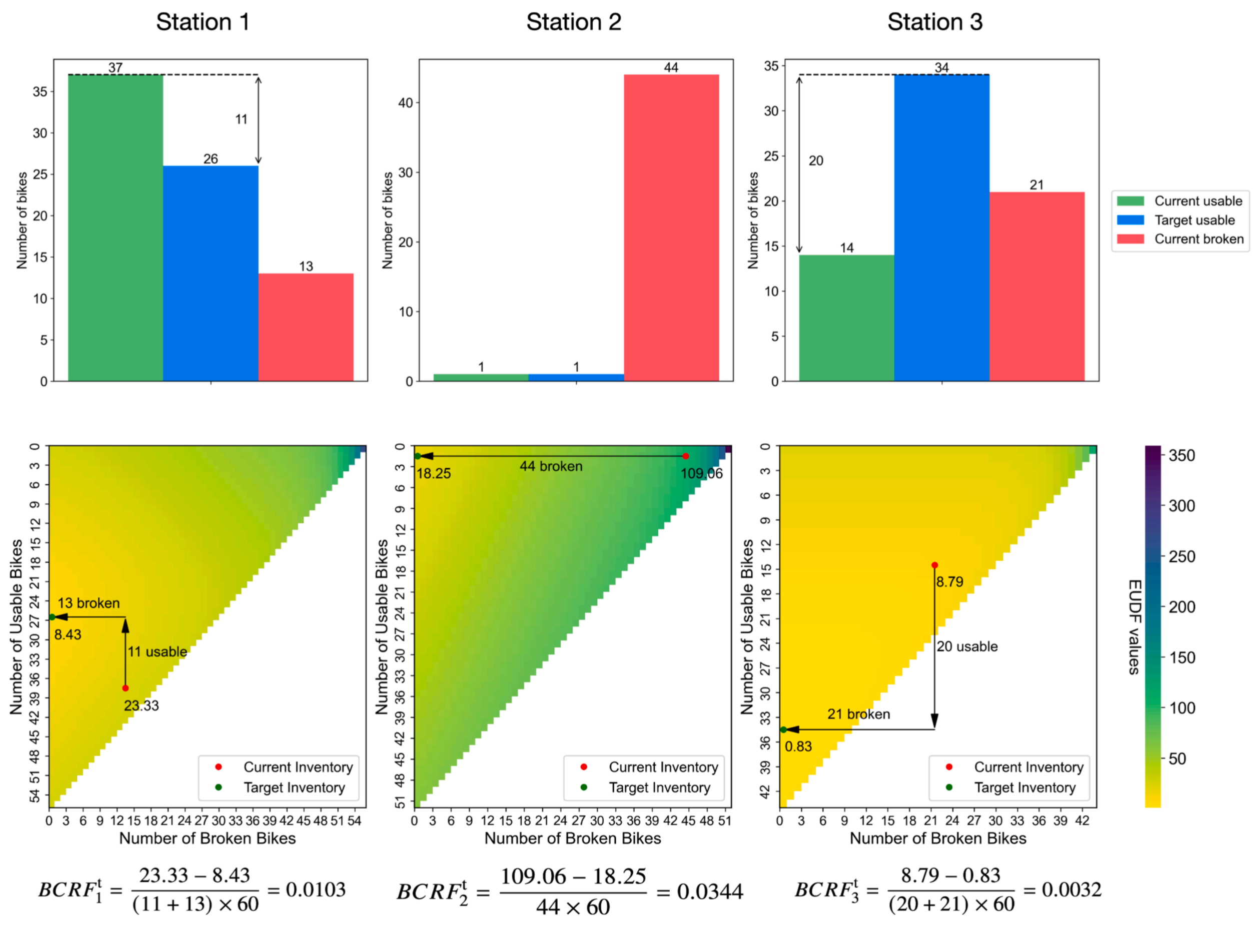
In the following subsections, we delve into the details of each component of the algorithm, including station budget assignment in Steps 2.1 and 3.1 (Section 4.4.1), repairer repairing quantity assignment in Step 2.2 and truck (un)loading quantity assignment in Step 3.2 and (Section 4.4.2), station (un)loading budget and truck (un)loading quantity adjustments in Steps 4 and 5 respectively (Section 4.4.3), and the final amendments to the routes, repairing, and (un)loading quantities in Step 6 (Section 4.4.4).

4.4.1. *Station budget allocation*

The SBC heuristic allocates truck (un)loading time and repairer repairing time at stations by leveraging the EUDF to define a Benefit-Cost Ratio Function (BCRF). The EUDF captures user dissatisfaction as a function of station inventories, offering a convex framework for assessing the impact of inventory adjustments. The BCRF builds upon the EUDF by quantifying the effectiveness of resource allocation at each station to reduce user dissatisfaction per unit of time spent. Specifically, the BCRF for a station is computed as the change in EUDF value—reflecting the benefit of truck (un)loading or repairs—divided by the operational time required to achieve that change.

Given the usable bike inventory p, the broken bike inventory b, and the target usable bike inventory q at station s, the BCRF value for a truck at that station (denoted as $BCRF_s^t(p,b)$) is defined as

$$BCRF_s^t(p,b) = \begin{cases} 0, & p = q, b = 0; \\ \dfrac{F_s(p,b) - F_s(q,0)}{t^{\text{load}} \cdot (|q-p| + b)}, & \text{Otherwise}. \end{cases} \tag{68}$$



$$BCRF_1^t = \frac{23.33 - 8.43}{(11+13)\times 60} = 0.0103 \qquad BCRF_2^t = \frac{109.06 - 18.25}{44\times 60} = 0.0344 \qquad BCRF_3^t = \frac{8.79 - 0.83}{(20+21)\times 60} = 0.0032$$

Fig. 4.1. An illustration of calculating the BCRF values of three stations based on the EUDF values.

For balanced stations without broken bikes, the BCRF value for a truck at that station is defined as zero, as indicated in the first condition of the piecewise function. In the second condition, the denominator is the time required to handle the usable and broken bikes at the station to achieve the target inventory and is considered to be a cost, assuming the value of time is fixed. The numerator is the penalty reduction and is considered to be a benefit. $F_s(q,0)$ is the minimum EUDF value so the numerator is always non-negative.

Fig. 4.1 shows an example illustrating the calculation of the values of the BCRF for trucks based on the EUDF. The three columns in this illustration represent Stations 1, 2, and 3, respectively.

In the first row, the bar charts display the current inventory of usable bikes, the target inventory for usable bikes, and the current inventory of broken bikes at each station. Arrows indicate the deviations of the current inventories of usable bikes from the corresponding target inventory levels. The second row contains heat maps showing the EUDF values for various combinations of usable and broken bike inventories at each station. Darker colors correspond to higher dissatisfaction, as captured by the EUDF. The red dots represent the current inventory levels, and the green dots indicate the target inventory levels. The arrows between these points show how the current inventory can be adjusted to match the target, specifying how many usable and broken bikes need to be loaded and unloaded.

For Station 3, the change in color between the current and target inventory levels is subtle, suggesting that spending time at this station has little effect on reducing the EUDF value. In contrast, Station 2 shows a significant color shift, indicating a large potential for reducing dissatisfaction by adjusting the inventory level.

The observations in the last paragraph are further quantified in the BCRF values on the last row, which are calculated by dividing the reduction in EUDF value by the time required to adjust the inventory from the current to the target level. Station 2′s BCRF value is nearly ten times greater than that of Station 3, emphasizing that more time should be spent on Station 2 to maximize the reduction in user dissatisfaction per unit of time.

Similarly, for each repairer, we also have $BCRF_s^r(p,b)$ defined as

$$BCRF_s^r(p,b)=\begin{cases}0, & p\geq q, b=0;\\ \dfrac{F_s(p,b)-F_s(p+\min\{b,\max\{q-p,0\}\},0)}{t^{\text{repair}}\cdot\min\{b,\max\{q-p,0\}\}}, & \text{Otherwise}.\end{cases}\tag{69}$$

For balanced and surplus stations without broken bikes, the BCRF value for a truck at that station is defined as zero, as indicated in the first condition of the piecewise function. In the second condition, the denominator $\min\{b,\max\{q-p,0\}\}$ is the maximum number of broken bikes repaired at the concerned station, which cannot be larger than the number of usable bikes required at that station because the repaired bikes become usable. That is why the repaired bikes are also included in the second term of the numerator.

The station repair budget is allocated based on the calculated $BCRF^r$ for each station, which is determined by prioritizing those with higher BCRF values that reflect a greater reduction in user dissatisfaction per unit of time. Specifically, the available repair time is distributed proportionally across stations according to the $BCRF^r$ of the station, ensuring a balanced allocation of resources while preventing excessive time spent at any single station. To make the best use of the time budget, any overly allocated time is redistributed to unsatisfied stations with unmet repair needs following the descending order of $BCRF^r$. The detailed algorithm for this is presented in **Algorithm 3** in Appendix C. The station (un)loading budget allocation algorithm for trucks is similar to the one for repairers, as presented in **Algorithm 4** in Appendix C, except that it considers the truck's loading and unloading operations instead of repair operations. To illustrate the main idea with a simple example, consider a route consisting of the above three stations 0–1-2–3-0, where 0 is the depot. After accounting for the route's travel time, the remaining time for bike loading and unloading is distributed proportionally across the stations based on their BCRF values. The results of this distribution are depicted in Fig. 4.2, which are calculated as follows:

$$\text{Station 1:}\frac{0.0103}{0.0103+0.0344+0.0032}=21.5\%,$$
$$\text{Station 2:}\frac{0.0344}{0.0103+0.0344+0.0032}=71.8\%,\ \text{and}$$
$$\text{Station 3:}\frac{0.0032}{0.0103+0.0344+0.0032}=6.7\%.$$

4.4.2. Loading, unloading, and repairing quantity assignment

With the station (un)loading budgets determined, we proceed to assign the quantities of usable bikes to be loaded or unloaded by trucks, broken bikes to be loaded by trucks, and broken bikes to be repaired by repairers. To improve the clarity of the main text, the detailed algorithms for these assignments (from **Algorithm 5** to **Algorithm 9**) are provided in Appendix D. Different from the traditional greedy heuristic, which assigns quantities based solely on the truck's inventory and the station's demand, our heuristic incorporates the station budget as an additional factor. The budget limits the maximum time a truck or repairer can spend at a station for operations. By considering this factor, we prevent excessive time consumption at early stations that may not necessarily justify such a large time investment, and ensure a more balanced and efficient allocation of resources across all stations.

4.4.3. Station (un)loading budget and truck (un)loading quantity adjustments

In **Algorithm 2** in Appendix B, after the assignment of loading and unloading quantities, it is observed that some stations may still have a loading budget unused while others encounter a shortfall. To make use of the extra time, an adjustment to the station (un)

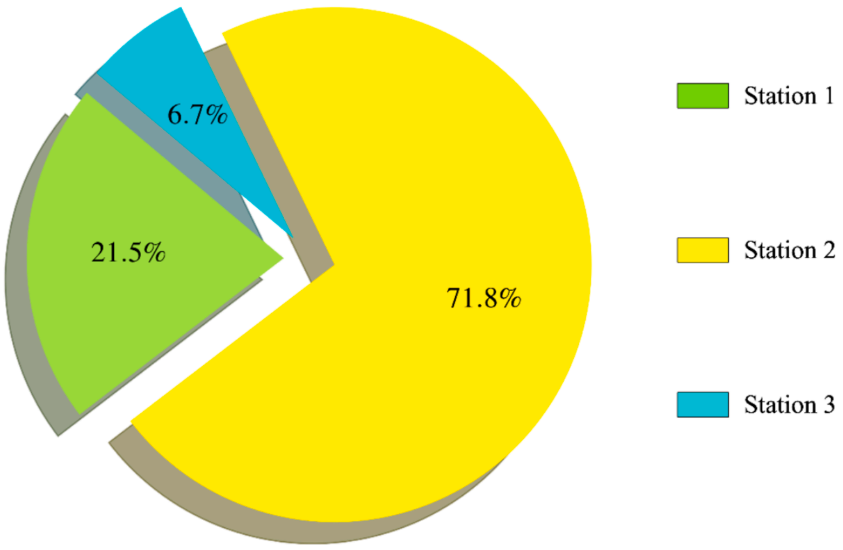


Fig. 4.2. Proportion of operational time allocated to each station based on their BCRF values.

loading budget allocations for trucks is necessary. The adjustment aims to redistribute the unused station (un)loading budgets from stations where the allocated time budget exceeds the actual loading and unloading time requirements to those stations where the budget is insufficient. The algorithm is given in Appendix E.

4.4.4. *Amendments to the routes and assigned quantities for trucks and repairers*

When assigning repair, loading, and unloading quantities, a conservative rounding-down strategy is applied (e.g., Line 3 in Algorithm 6 in Appendix D). This can lead to the availability of unused time budgets, which can be utilized for more bike repairs and loading. Therefore, this section provides amendments to the routes and assigned quantities for trucks and repairers.

The amendment procedure begins by removing stations that are not assigned with any loading, unloading, and repairing quantities, then calculating the leftover time for each truck and repairer based on the outputs from **Algorithm 2**. It then focuses on the stations which are still not balanced. For each repairer at every station, the amendment procedure explores the possible broken bike repairing quantity that ranges from zero to the maximum allowed by the extra time. It selects the repairing quantity that results in the lowest $BCRF_s^{\mathrm{r}}$. Meanwhile, each truck checks the surplus stations and redistributes as many surplus bikes as possible to deficit stations. The choice of the surplus and deficit stations prioritizes those with the highest $BCRF_s^{\mathrm{t}}$. For each surplus station checked by the truck, the candidate destinations are set as the stations between the current station and the next depot along the truck's route, with surplus bikes unloaded to the next depot if unloading at any candidate station would violate station or truck capacity constraints. After checking all surplus stations, if extra time remains, supplementary loadings of broken bikes are conducted similarly, with the distinction that broken bikes can only be unloaded at the depot.

4.5. *Feasibility check and solution evaluation*

After determining the loading, unloading, and repairing quantities, we can evaluate solution ϕ by

$$Z(\phi) = f(\phi) + \eta\varepsilon(\phi) \tag{70}$$

where $f(\phi)$ is the sum of the penalty costs of the total user dissatisfaction at all stations in the network and the total cost of carbon emissions during the rebalancing process. $\varepsilon(\phi)$ is related to the penalties for solution infeasibility and η is the penalty coefficient of $\varepsilon(\phi)$. If $\varepsilon(\phi)$ is 0, then solution ϕ is feasible.

The infeasibility we consider here originates from the fact that the heuristic proposed in Section 4.4 determines the quantities route by route separately, without considering the interdependencies between different routes and their effects on station inventories. This strategy is used to make the decision-making process more straightforward and easier to implement. It assumes that the arrival times at stations follow the visiting order of stations in the route, with all visits by trucks with lower indices occurring before any visits by trucks with higher indices. This assumption, however, can lead to invalid quantities being assigned.

Consider a problem where two trucks perform the repositioning task in a network with one depot. For simplicity, we assume there are no repairers in this example. The route for Truck 1 is (0–1–3–5–0), and the route for Truck 2 is (0–2–5–0). We notice that Station 5 is visited by both trucks. It is unclear which truck would arrive first at Station 5 when the arrival times at stations, which depend on quantity assignment and travel times, are not calculated. The method proposed in Section 4.4 presumed that the first truck arrived at Station 5 ahead of the second truck, updating the inventories and assigning the repositioning quantities accordingly, which can lead to an invalid assignment. For example, as shown in Fig. 4.3 **(a)**, suppose the capacity of the station is 20, with a usable bike inventory of 3, a broken bike inventory of 10, and a target usable bike inventory of 13. If Truck 1 arrives at Station 5 from Station 3 with a truckload of 0 usable bikes and 0 broken bikes, according to the strategy in Section 4.4, it collects 10 broken bikes (provided the remaining time is enough) at Station 5, and thus the broken bike inventory of Station 5 is updated to 0. Then, for Truck 2, if it arrives at Station 5 with 12 usable bikes, it delivers 10 usable bikes to achieve the target usable bike inventory. However, if Truck 1 actually arrives at Station 5 later than Truck 2, the decision to deliver 10 usable bikes by Truck 2 results in a total inventory of usable and broken bikes of 23 (3 + 10 + 10), which exceeds the station capacity (see Fig. 4.3 **(b)**). The scenario could become more complicated if the repairers also conduct repairs at the station. Therefore, we need to validate the feasibility of the solution by examining the inventories of the multiple visited stations after the operation of each visited truck or repairer.

After the operations at a node, the usable or broken bike inventory may be below 0, or the total inventory of usable and broken bikes may be over the capacity of the station. We set $\varepsilon(\phi)$ to the total number of bikes that exceeds the capacity or is below 0, which is calculated following the procedure below:

Step 1: Calculate the arrival time ar_s for every station $s \in S$ visited by each repairer and truck according to solution ϕ and the quantities determined in Section 4.4, and record the results as a list of tuples $\left(n, s, y_s^{b^-}, y_s^{u^+}, y_s^{u^-}, g_s, ar_s\right)$, where n represents the index of the repairer or truck operating at station s, and the number of loaded broken bikes, loaded usable bikes, unloaded usable bikes, and repaired broken bikes are represented as $y_s^{b^-}, y_s^{u^-}, y_s^{u^+}$, and g_s, respectively.
Step 2: Sort the list of tuples according to the arrival time ar_s.
Step 3: Select the nodes in the list one by one in the ascending order of arrival time and perform the loading/unloading/repairing to update the inventory of usable and broken bikes. Update $\varepsilon(\phi)$ if, after the operations at some node, the usable bike inventory p_s or the broken bike inventory b_s is below 0, or the total inventory of usable and broken bikes exceeds the capacity of the station C_s by

$$\varepsilon(\phi) \leftarrow \varepsilon(\phi) - (\min\{p_s, 0\} + \min\{b_s, 0\} + \min\{C_s - p_s - b_s, 0\}).$$

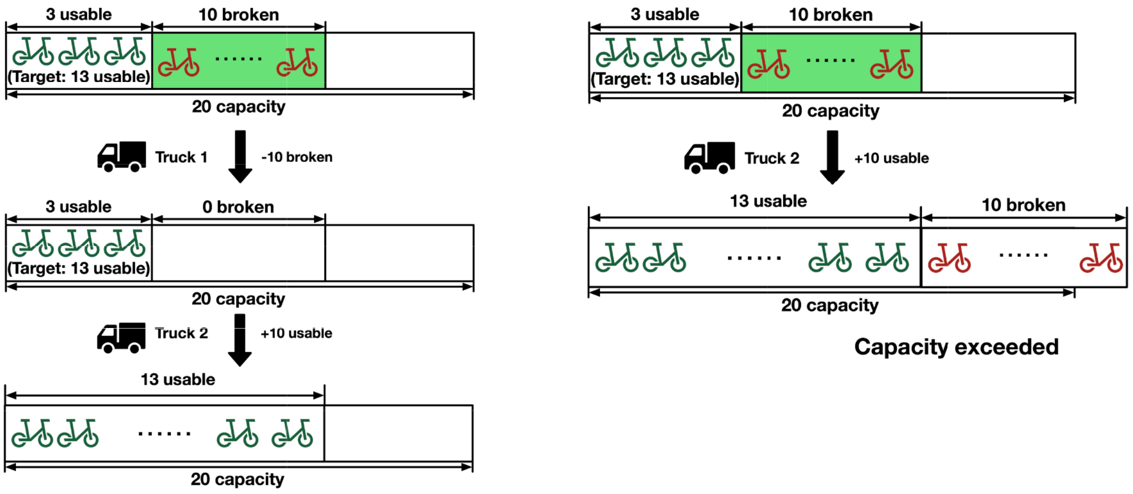
Next, we define the biased fitness value of a solution to consider the diversity in each subpopulation, which is calculated with

$$BF(\phi) = rank_{\text{fit}}(\phi) + \left(1 - \frac{N_{\text{elite}}}{N_{\text{indiv}}}\right) rank_{\text{sim}}(\phi) \tag{71}$$

where N_{elite} is the number of elite individuals that survive to the next generation, and N_{indiv} is the actual number of individuals in the subpopulation. $rank_{\text{fit}}(\phi)$ is the rank of solution ϕ in its subpopulation regarding $Z(\phi)$ sorted in the ascending order, and $rank_{\text{sim}}(\phi)$ is the rank of solution ϕ regarding the similarity with other solutions in the subpopulation.

In our problem, for each solution ϕ, we gather the arc set of all truck routes in ϕ to the set TA_ϕ, and the arc set of all repairer routes in ϕ to a set RA_ϕ. Let χ be the subpopulation that ϕ belongs to. Depending on the feasibility of ϕ, $\chi = Pop_{\text{f}}$ if ϕ is feasible, and $\chi = Pop_{\text{i}}$ otherwise. Then, the similarity of ϕ with respect to the subpopulation χ it belongs to is defined as

$$\left| TA_\phi \cap \left(\bigcup_{\phi' \in \chi \setminus \{\phi\}} TA_{\phi'} \right) \right| + \left| RA_\phi \cap \left(\bigcup_{\phi' \in \chi \setminus \{\phi\}} RA_{\phi'} \right) \right|. \tag{72}$$



(a) The assigned (un)loading quantities at Station 5, assuming Truck 1 arrives at this station first and Truck 2 later

(b) Executing the assigned (un)loading quantities in (a) but Truck 2 arrives at this station first

Fig. 4.3. An example of solution infeasibility in the quantity assignment.

which is the sum of the number of common arcs between the truck routes in the solution and the union of truck routes from every other solution in the corresponding subpopulation and the number of common arcs between repairer routes in the solution and the union of repairer routes from every other solution in the corresponding subpopulation.

4.6. *Selection, crossover, education, and repairing*

In each iteration, two individuals are selected using a binary tournament, which randomly picks two individuals from the population and reserves the one with a better-biased fitness value. The crossover and local search strategies for routes in education and repair are basically the same as those proposed by Li et al. (2016). Therefore, we omit the details here. The only difference is that in our solution, there can be multiple truck routes and multiple repairer routes. Hence, for each solution, the crossover, education, and repair are carried out route by route. For cases with more than one truck or repairer, we only consider the crossover of the routes with the same index. For example, the route of the first repairer in solution ϕ_1 only makes a crossover with the route of the first repairer in solution ϕ_2. The repair operation follows the method by Vidal (2022) except that the SBC heuristic is used after each new route is generated.

4.7. *Population management and search guidance*

The population management mechanism works with selection, crossover, and education operators to help propagate the characteristics of excellent solutions, increase population diversity, and provide guidance for a more efficient search. At the beginning of the algorithm, 4μ individuals are initialized (Li et al., 2016), where μ is the minimum subpopulation size. Each of them directly goes through an education process using local search procedures and the SBC heuristic, and then falls into a feasible or infeasible subpopulation based on the feasibility of the solution. The repair operation is activated for infeasible individuals with a probability of P_{repair}. The repair operation runs local search procedures and the SBC heuristic under a penalty factor of 10 times the original one, aiming to convert an infeasible solution into a feasible one (Vidal, 2022). If the repair is successful, the resulting individual is added to the feasible subpopulation. Otherwise, the infeasible one before repair is preserved in the infeasible subpopulation. The algorithm keeps generating new individuals under the process of selection, crossover, education, and repair until the time limit or the maximum number of iterations without improvement is reached.

During the above process, individuals generated through crossover, education, or repair are added to the corresponding subpopulation based on their feasibility. If the size of either subpopulation reaches $\mu + \lambda$, where λ is the number of offspring in a generation, survivor selection is conducted to remove the λ solutions with the highest biased fitness values to ensure that the size equals μ.

As in the study of Li et al. (2016), the penalty coefficient η in Eq. (70) is modified every 100 iterations to adjust the proportion of feasible solutions. Let ξ^{REF} be the target proportion of feasible individuals. If the proportion of feasible individuals with respect to $\varepsilon(\phi)$ is less than $\xi^{\text{REF}} - 5\%$ or greater than $\xi^{\text{REF}} + 5\%$, then we change η to 1.2η or 0.85η, respectively (Li et al., 2016).

The diversification operation is triggered if the best solution has not been improved for It_{div} iterations. In the diversification, the best $\mu/3$ solutions of each subpopulation are preserved (Li et al., 2016), and 4μ new individuals are generated, followed by the survivor selection to adjust the size of each subpopulation to μ.

5. Numerical studies

The proposed model was implemented in Python 3.11 using Gurobi Python API (Gurobi Optimization, LLC, 2023). The hybrid algorithm HGSDAC-SBC was implemented with C++. All numerical experiments were conducted on a computer equipped with an Intel Core i9-12900 processor (12 cores, 24 threads) and 64 GB of physical memory.

In Section 5.1, we introduce the parameter tuning and setting for HGSDAC-SBC. In Section 5.2, we demonstrate the effectiveness of

Table 5.1
Parameter setting of the numerical experiments.

Parameter		Value	References
ρ_0	The empty-load fuel consumption rate of each truck (liters/km)	0.252	Franceschetti et al. (2013); Shui and Szeto (2018)
ρ^*	The full-load fuel consumption rate of each truck (liters/km)	0.258	Franceschetti et al. (2013); Shui and Szeto (2018)
CER	The CO_2 emission rate (kg/liter)	2.61	Wang and Szeto (2018)
Q_k	The capacity of truck k (bikes)	25	Ho and Szeto (2017)
t^{load}	Time for loading/unloading a bike (seconds)	60	Liu et al. (2018)
t^{repair}	Time for repairing a bike on-site (seconds)	300	Jin et al. (2022)
v^{t}/v^{m}	The ratio of speed between a truck and an electric tricycle	1.68	Wu (2021); Fang et al. (2019)
c^{p}	The penalty cost for a user unable to rent/return a bike (CNY)	2	Chen et al. (2020)
c^{e}	The unit carbon trading cost (CNY/kg)	0.06	Y. Jia et al. (2020)
θ	The penalty coefficient used to eliminate unnecessary loading or unloading operations	1e-8	–

the proposed algorithm by presenting results and running times for small instances involving a single truck and repairer. In Section 5.3, we evaluate the algorithm's performance on larger networks with multiple trucks and repairers. Section 5.4 focuses on the impact of varying repair times on the cost-effectiveness of including a repairer through experiments on different proportions of broken bikes.

The parameters regarding the network are from the dataset used by Ho and Szeto (2017), which is available at https://www.dropbox.com/s/jbsfefi2wqsssrr/hlns-set3.zip?dl=0. The dataset contains the travel times between nodes, the capacity of each station, and the initial inventory of usable bikes at each station. The travel times used in the experiments are also from the dataset above, in which the asymmetric travel times (in seconds) between the stations were obtained from the Open Source Routing Machine project (https://project-osrm.org/) on the travel times of trucks between stations. However, since the project does not provide travel times for electric tricycles (the vehicle assumed to be used by repairers in this study), we assume the speed of the electric tricycles is proportional to that of the trucks. The proportionality constant is determined based on values from the existing literature that provide relevant speed information for both vehicles (see Table 5.1). As a result, the travel times for repairers between nodes can be calculated accordingly based on the constant.

The EUDF value of each station was calculated based on the historical data of Citi Bike, which consists of the number of bike rentals and returns at each station, recorded every 15 min from 6:00 a.m. to 12:00p.m., over the course of a single day.

All instances used in this section were generated by randomly selecting a certain number of stations from the 518-station dataset. Specifically, in parameter tuning in Section 5.1, a 60-station network was generated. In the experiment in Section 5.2, five different instances were generated for each network size (6, 10, and 15 stations). In the experiment in Section 5.3, instances with network sizes ranging from 60 to 500 stations were created, with one instance generated for each size. In the experiment in Section 5.4, one of the 15-station networks generated in Section 5.2 was used. As the dataset provides no information on the initial inventory of broken bikes, we set the initial quantity of broken bikes b_i^0 at station i as follows: in the experiments in Sections 5.1, 5.2, and 5.3, b_i^0 was assigned a pre-generated random value within the range $\left[0, C_i - p_i^0\right]$, while in the experiment in Section 5.4, it was set proportional to the residual capacity $C_i - p_i^0$. Note that the time-indexed formulation for the MILP converts all travel times into integer intervals by dividing the actual travel time by τ and rounding up to the nearest integer, which can lead to an inflated time representation caused by such an approximation. Consequently, the lower bounds obtained from Gurobi in this section may not be the accurate bounds for the actual (continuous-time) problem.

The setting for other parameters of the experiments is listed in Table 5.1. Note that except for the experiment in Section 5.3, in which we vary the number of trucks and repairers on larger networks, we fixed the number of trucks and repairers to one. For the repositioning time budget, we uniformly set it to two hours in Sections 5.1, 5.2, and 5.4. In Section 5.3, for larger instances, experiments were conducted under both two-hour and three-hour time budgets. In the time discretization, we used 600 s as the length of each period τ for most experiments, except in Section 5.4, where repair times vary between 300 and 1200 s. To accommodate the largest repair time in these settings, in Section 5.4, we set the length of each period τ to 1200 s, ensuring that repair times do not exceed the period as defined in the parameter table in Section 3. The codes and the generated datasets used in the experiments are available at https://github.com/rqhu1995/brpwr.

5.1. Parameter tuning and setting for the algorithm

The parameter setting mostly follows that of Li et al. (2016), who referred to the research conducted by Vidal et al. (2012, 2013, 2014). The only parameter that differs from theirs and requires tuning is η, the one related to the penalty of infeasible solutions. η was set to take values from the set {10, 50, 100, 150, 200, 300, 400, 500}. We ran the algorithm, setting η to be each value from the candidate set. The result of the average objective value in 20 runs under different values of η is given in Fig. 5.1., and 100 was chosen because it yields the best average result within a shorter CPU time. The detailed parameter setting is summarized in Table 5.2.

5.2. Comparison of the results between HGSDAC-SBC and Gurobi[a]

This section aims to compare the results obtained from HGSDAC-SBC with the optimal solutions obtained by Gurobi for small network instances. Across all instances, the optimal solutions could be found using both the solver and the heuristic for all the tested instances, and when applied with different random seeds 20 times to a single instance, the heuristic consistently found the optimal solution. Therefore, we only report the CPU times using Gurobi and the average CPU times over 20 runs with HGSDAC-SBC in Table 5.3.

In most instances, HGSDAC-SBC outperformed Gurobi in terms of CPU times, producing optimal solutions in significantly less time. Noticeably, in one of the 10-station instances (instance 3), HGSDAC-SBC completed the computations in as little as 3.02 s on average, whereas Gurobi required over 13,000 s (nearly four hours) for the same instance. However, in a smaller 6-station instance (instance 4), Gurobi required slightly less CPU time than HGSDAC-SBC. This is likely because a significant portion of the computing time of the heuristic was spent on the time required to check the convergence. Despite this, the overall computational benefits of HGSDAC-SBC are

[a] We used four threads and set the parameter *Heuristics* in Gurobi to 0. To avoid no feasible solution found by Gurobi within the time limit for large networks, for all the experiments run with Gurobi in Section 5, we used Gurobi's "*MIP starts*" feature that enables users to provide it with initial solutions. We formed the solution by setting trucks to travel to no station and stay at the depot, and each repairer visits only one station (Station 1 if $|R|=1$, stations 1 and 2 if $|R|=2$).

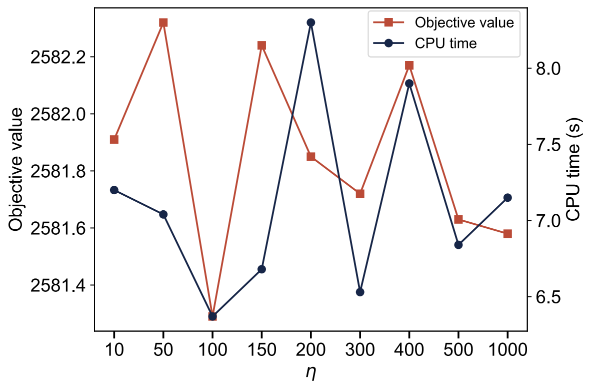


Fig. 5.1. Parameter tuning results for the penalty coefficient η ($|S| = 60$, $T = 2$ h, $|K| = |R| = 1$).

Table 5.2
The parameter setting for HGSDAC-SBC.

Parameter	Value
The minimum subpopulation size μ	25
The number of offspring in a generation λ	40
The target proportion of feasible individuals ξ^{REF}	0.2
The maximum number of iterations without improvement It_{NI}	5000
The number of successive iterations that triggers the diversification operation It_{div}	200 ($0.4It_{\mathrm{NI}}$)
The time limit for executing the algorithm $T_{\max}$	7200 s
Probability for repairing an infeasible solution P_{repair}	0.5
The number of elite individuals that survive to the next generation N_{elite}	10
Penalty coefficient for solution infeasibility η	100

Table 5.3
A comparison between the CPU times (s) of Gurobi and HGSDAC-SBC on small networks (Repositioning time budget $T = 2$ h, $|K| = |R| = 1$, $\tau = 600$ s).

$\lvert S\rvert$	instance	Gurobi	HGSDAC-SBC
6	1	13.41	6.29
	2	13.85	5.75
	3	66.97	6.22
	4	2.29	3.63
	5	22.29	4.84
10	1	928.50	3.87
	2	1409.89	5.27
	3	13768.72	3.02
	4	2404.03	5.16
	5	4172.75	4.17
15	1	2602.24	7.80
	2	4209.67	4.89
	3	6283.58	4.72
	4	3374.38	5.32
	5	2371.40	5.08

evident as the network size increases, making it a more efficient choice for large-scale problems. It is also worth noting that the time-index formulation can make the model computationally hard, even for instances comprising just a small number of stations, further showing the necessity of using heuristics to solve the problem efficiently.

5.3. Comparison of results from HGSDAC-SBC and Gurobi on larger networks

In this section, we proceed to compare the results obtained from HGSDAC-SBC with the best feasible solutions found by Gurobi for larger network instances with different numbers of trucks and repairers. Gurobi was given a two-hour time limit for computation in all instances in this section. In some cases, Gurobi terminated earlier due to running out of memory. Hence, no CPU times are reported for those instances (marked as N/A). The results are reported in Table 5.4.

As shown in both Table 5.4 and Table 5.5, Gurobi was unable to find optimal solutions within the two-hour CPU time limit in any of the instances. For all instances where memory did not run out, Gurobi used the entire two-hour time limit to provide the best feasible solution (UB). The difficulty in solving these larger instances again highlights the computational complexity of the problem.

On the other hand, HGSDAC-SBC consistently outperformed Gurobi by finding better feasible solutions (as reflected by a positive gap) in significantly less time. As the repositioning time budget increased from two hours to three hours in the model, the performance gap between HGSDAC-SBC and Gurobi widened, especially for instances comprising 60 stations. For example, in the 60-station instance with one truck and two repairers, the gap increased from 1.55 % under the two-hour budget to 32.58 % under the three-hour budget. This indicates that the increased complexity of the model with longer time budgets makes it more difficult for Gurobi to find better solutions, while HGSDAC-SBC is still efficient in obtaining better solutions. Additionally, we notice that the computation for some very large instances ($|S| = 400$ and 500) used up the 64 GB memory of the computer before the two-hour time limit was reached. This suggests that as network size increases and resources become a limitation for Gurobi, HGSDAC-SBC shows an additional advantage by requiring less memory and less CPU time, making it more suitable for real-world BRP with large networks.

5.4. Cost-effectiveness of introducing a repairer under varying repair times and broken bike inventory levels

In this section, we analyze the cost-effectiveness of involving a repairer under different repair times and broken bike quantities. The experiment was conducted on a fixed 15-station network, with the number of broken bikes at each station being proportional to the station's residual capacity with a proportional coefficient denoted by ℓ. The initial broken bike quantity at station i was set to $b_i^0 = \lfloor \ell \cdot (C_i - p_i^0) \rfloor$. ℓ ranges from 10 % to 100 % in an increment of 10 %. For each value of ℓ, the repair time per bike varies from 300 s to 1200 s, with a step size of 100 s. The length of period τ is set to 1200 s in this section to guarantee $t^{\text{repair}} < \tau$.

To capture the cost-effectiveness of a repairer, an indicator ν is introduced, which is defined as the ratio of the reduction cost in user dissatisfaction (when a repairer is introduced) to the wage paid to the repairer, which is calculated as

$$\nu = \frac{c^{\text{p}} \sum_{i \in S} \left(\widehat{F}_i - \widehat{F}_i^{'} \right)}{\varpi \cdot T/3600} \tag{73}$$

where $\widehat{F}_i$ and $\widehat{F}_i^{'}$ are the user dissatisfaction after the repositioning at station i with and without a repairer, respectively, c^{p} is the unit cost of the dissatisfaction, ϖ is the hourly wage of a repairer, and T is the time budget for the repositioning in seconds. The division by 3600 converts the time from seconds to hours, ensuring consistency with the wage unit in CNY/h. The numerator is the reduction in the dissatisfaction cost, while the denominator is the cost paid to the repairer. Note that the dissatisfaction after the repositioning with no repairers $\widehat{F}_i^{'}$ was calculated by setting $|R| = 0$ in the optimization model. The hourly wage was determined based on a piece of job-hunting information online,[b] and was set to 33.33 CNY/h.

For each setting of ℓ, experiments under varying repair times were conducted, and the results of the variations in ν over repair times are plotted in Fig. 5.2. The results show a clear trend across all instances: As repair time increases, the cost-effectiveness ratio decreases consistently, meaning that on-site repairs become less cost-effective. This implies that a longer repair time leads to a smaller reduction in user dissatisfaction, as fewer repairs can be completed by the repairer within the available time. In the extreme case, when the repair time exceeds a certain threshold, introducing on-site repairs is cost-ineffective.

One key observation is the existence of a critical point at which the cost-effectiveness ratio ν equals 1.0 at some level of broken bikes. For lower levels of broken bikes ($10\% \leq \ell \leq 20\%$), this ratio remains below 1.0 regardless of the repair times, indicating that the cost of employing a repairer outweighs the benefits. In such cases, the system can still be effectively managed by trucks alone for repositioning, as the low number of broken bikes does not justify the additional cost of involving a repairer. However, for systems with higher levels of broken bikes ($\ell \geq 30\%$), a turning point emerges where the cost-effectiveness ratio shifts from above to below 1.0. This signifies that as the proportion of broken bikes increases, the benefit of reducing user dissatisfaction gradually begins to outweigh the cost of employing a repairer at lower ranges of repair times. Larger proportions of broken bikes shift the incorporation of a repairer from being cost-ineffective to cost-effective at lower ranges of repair times, reflecting the growing necessity for on-site repairs as the average damage level within the system rises. The changes in the cost-effectiveness ratio over the broken bike inventory levels suggest the necessity for a dynamic assignment of repairers based on the actual level of damage within the system, which includes varying the number of repairers in response to the dynamic change in the broken bike levels in the system to ensure the cost-effectiveness. Specifically, part-time repairer scheduling can serve as a flexible strategy to further enhance resource allocation, allowing systems to adjust the input for repairs dynamically.

Another important observation is the shift in the repair time at which the cost-effectiveness ratio ν equals 1.0. As the level of broken bikes increases, this critical point moves toward longer repair times. For example, when the broken bike inventory reaches 50 %, the point at which $\nu = 1.0$ occurs at approximately 492 s, whereas for the 90 % level, it occurs at around 580 s. This suggests that with a higher number of broken bikes, longer repair times can still yield a favorable cost-effectiveness ratio before the repairer's cost outweighs the benefit. However, once repair times exceed around 645 s, employing a repairer becomes less cost-effective across all instances, regardless of the level of broken bike inventory. Although our model assumes the same repair time for all bikes, this can be viewed as an average repair time, representing the overall damage level of the system. Hence, such a relatively long repair time suggests that the system is experiencing significant damage. Therefore, it may be more effective to shift resources from reactive repairs to preventive maintenance. This strategy can be adapted based on the system's changing damage degrees, helping to reduce the

[b] https://www.zhipin.com/job_detail/81bcad2cd24453bf1HF73NW4FlJU.html.

Table 5.4
A comparison between the results of Gurobi and HGSDAC-SBC on larger networks (Repositioning time budget $T = 2$ h, $\tau = 600$ s).

$\lvert S\rvert$	$\lvert K\rvert$	$\lvert R\rvert$	Gurobi			HGSDAC-SBC				[5]Gap (%)
			UB	LB	CPU (s)	[1]Obj. Avg.	[2]Obj. Best.	[3]Obj. Std.	[4]CPU (s)	
60	1	1	2604.63	2517.12	7200.00	2581.29	2579.82	0.86	6.37	0.90
	1	2	2533.44	2420.32	7200.00	2494.24	2486.32	4.09	9.20	1.55
	2	1	3581.80	2284.28	7200.00	2400.65	2379.21	10.18	9.87	32.98
	2	2	3517.71	2202.55	7200.00	2324.50	2302.98	11.76	10.90	33.92
90	1	1	4510.58	3893.04	7200.00	4038.28	4014.71	13.59	8.81	10.47
	1	2	4159.43	3782.32	7200.00	3941.78	3915.27	14.03	13.40	5.23
	2	1	4510.58	3656.81	7200.00	3862.44	3829.09	19.77	9.56	14.37
	2	2	4488.64	3557.92	7200.00	3768.58	3733.90	21.59	13.28	16.04
120	1	1	5113.72	4247.68	7200.00	4303.05	4293.75	3.58	9.13	15.85
	1	2	5103.87	4169.64	7200.00	4237.52	4226.39	4.17	14.96	16.97
	2	1	5113.72	4024.31	7200.00	4167.75	4129.41	16.19	11.45	18.50
	2	2	5103.87	3951.26	7200.00	4097.19	4071.74	11.87	15.98	19.72
200	1	1	8657.30	2919.70	7200.00	7821.12	7793.53	19.35	12.36	9.66
	1	2	8657.30	7563.69	7200.00	7719.58	7694.50	12.08	23.96	10.83
	2	1	8657.30	7378.11	7200.00	7641.82	7573.84	28.23	14.19	11.73
	2	2	8657.30	7286.86	7200.00	7561.16	7511.34	22.96	23.90	12.66
300	1	1	11717.13	10898.80	7200.00	11169.22	11134.10	15.16	16.55	4.68
	1	2	11716.99	10757.43	7200.00	11057.76	10990.90	19.38	25.67	5.63
	2	1	11717.13	10561.75	7200.00	10989.39	10953.90	19.84	19.59	6.21
	2	2	11716.99	10442.63	7200.00	10888.38	10848.40	20.63	33.70	7.07
400	1	1	15068.74	6026.88	7200.00	14280.67	14267.10	5.89	25.27	5.23
	1	2	15060.58	6026.88	7200.00	14178.63	14149.00	12.33	34.05	5.86
	2	1	15068.74	6026.59	7200.00	14169.14	14104.30	21.84	25.92	5.97
	2	2	15060.58	6026.59	7200.00	14059.18	14033.60	15.29	43.77	6.65
500	1	1	19793.68	7716.08	7200.00	18961.58	18927.40	18.77	27.36	4.20
	1	2	19785.07	7716.08	7200.00	18793.14	18734.00	22.68	41.60	5.01
	2	1	19793.68	7715.69	N/A	18752.12	18618.10	50.56	33.65	5.26
	2	2	19785.07	N/A	N/A	18612.62	18505.40	37.57	51.13	5.93

[1] the average objective value for each instance in 20 runs.
[2] the best objective value for each instance in 20 runs.
[3] the standard deviation of the objective value for each instance in 20 runs.
[4] average computational time for each instance in 20 runs in seconds.
[5] (UB – Obj. Avg) / UB.

number of broken bikes and the average repair time over time and preventing the system from reaching a state where repairs become less cost-effective.

6. Conclusion

We investigate a new and practical static BRP with broken bikes and on-site repairers. The problem is to find the best routes and loading/unloading instructions of both usable and broken bikes for the repositioning trucks while determining the best routes for on-site repairers and the number of broken bikes repaired by each repairer at each station. The goal is to reduce the sum of the penalty costs of user dissatisfaction and the cost of total CO_2 emissions. The problem is formulated as a time-index Mixed Integer Linear Programming (MILP) model. A hybrid algorithm HGSDAC-SBC is proposed to solve the problem efficiently for large instances. The algorithm relies on a novel concept of station budget, which utilizes the Benefit-Cost Ratio Function (BCRF) to determine (un)loading and repair quantities at stations, ensuring that time is spent effectively across the network.

Numerical experiments on small networks of up to 15 stations demonstrated that HGSDAC-SBC consistently achieves optimal solutions, matching Gurobi's results while requiring significantly less computational time in most cases. For larger networks, ranging from 60 to 500 stations, HGSDAC-SBC outperformed Gurobi in finding a feasible solution obtained under a two-hour time limit, demonstrating the algorithm's computational efficiency. Additionally, we analyzed the cost-effectiveness of introducing a repairer under varying repair times and broken bike inventory levels. The findings show that long repair times reduce cost-effectiveness. In the extreme case, when the repair time exceeds a certain level, introducing on-site repairs is cost-ineffective. Moreover, when the number of broken bikes in the system is small, introducing on-site repairs can become detrimental. The results also underscore the need for a dynamic repairer allocation strategy based on the system's actual damage level and suggest that preventive maintenance strategies can reduce the average number of broken bikes in the system and the average repair time over time.

It is worth noting that we conducted additional experiments comparing a full station set with a reduced station set, where stations with no broken bicycles and inventory levels that (nearly) minimize the EUDF were removed. The results showed no statistically significant difference in terms of objective values or CPU times between the two approaches. This indicates that the heuristic's natural allocation of negligible station budgets to such stations achieves a similar effect to explicitly removing them from the problem, supporting the efficiency and robustness of the current approach. Since the manuscript is already lengthy, we omit the detailed

Table 5.5
A comparison between the results of Gurobi and HGSDAC-SBC on larger networks (Repositioning time budget $T = 3$ h, $\tau = 600$ s).

$\|S\|$	$\|K\|$	$\|R\|$	Gurobi			HGSDAC-SBC				Gap (%)[5]
			UB	LB	CPU (s)	[1]Obj. Avg.	[2]Obj. Best.	[3]Obj. Std.	[4]CPU (s)	
60	1	1	3553.62	2337.38	7200.00	2433.29	2412.09	3.14	10.50	31.53
	1	2	3507.20	2221.33	7200.00	2364.56	2352.33	5.67	12.33	32.58
	2	1	3581.80	2035.17	7200.00	2264.57	2248.72	8.76	13.65	36.78
	2	2	3507.20	1930.24	7200.00	2214.40	2196.06	6.44	12.99	36.86
90	1	1	4503.66	3705.60	7200.00	3903.04	3865.03	16.13	10.06	13.34
	1	2	4481.06	3563.25	7200.00	3778.54	3735.60	20.69	14.04	15.68
	2	1	4503.66	3398.88	7200.00	3705.04	3652.58	20.54	15.00	17.73
	2	2	4481.06	3264.80	7200.00	3606.92	3563.30	21.66	19.82	19.51
120	1	1	5113.72	4085.94	7200.00	4203.49	4180.03	9.81	8.84	17.80
	1	2	5103.87	3982.62	7200.00	4106.56	4087.78	8.83	18.28	19.54
	2	1	5113.72	3784.05	7200.00	4023.59	4007.56	10.96	13.59	21.32
	2	2	5103.87	3685.32	7200.00	3952.46	3909.70	17.55	17.09	22.56
200	1	1	8657.30	7457.23	7200.00	7695.07	7636.22	22.75	14.47	11.11
	1	2	8657.30	7322.35	7200.00	7575.25	7539.85	14.29	25.61	12.50
	2	1	8657.30	2916.37	7200.00	7471.22	7406.93	32.94	19.45	13.70
	2	2	8657.30	2916.37	7200.00	7393.52	7333.03	30.51	27.84	14.60
300	1	1	11715.76	4568.18	7200.00	11024.28	10973.30	21.87	18.06	5.90
	1	2	11715.62	10462.88	7200.00	10901.05	10858.60	16.88	38.59	6.95
	2	1	11715.76	4568.16	7200.00	10830.35	10756.00	23.79	27.03	7.56
	2	2	11715.62	4568.16	7200.00	10692.76	10638.50	25.19	44.10	8.73
400	1	1	15068.74	6026.59	7200.00	14137.72	14090.00	20.16	21.88	6.18
	1	2	15060.58	6026.59	7200.00	14009.58	13974.40	15.25	45.86	6.98
	2	1	15068.74	6026.58	N/A	13968.64	13926.70	21.06	32.79	7.30
	2	2	15060.58	N/A	N/A	13851.78	13822.50	12.96	55.96	8.03
500	1	1	19793.68	7715.71	7200.00	18818.29	18716.10	37.12	21.88	4.93
	1	2	19785.07	N/A	N/A	18596.33	18539.40	26.59	56.24	6.01
	2	1	19793.68	N/A	N/A	18572.48	18493.30	38.87	35.34	6.17
	2	2	19785.07	N/A	N/A	18407.06	18250.20	44.37	62.35	6.96

[1] the average objective value for each instance in 20 runs.
[2] the best objective value for each instance in 20 runs.
[3] the standard deviation of the objective value for each instance in 20 runs.
[4] average computational time for each instance in 20 runs in seconds.
[5] (UB – Obj. Avg) / UB.

experimental results in the manuscript.

While this study addresses key aspects of the problem, two limitations must be acknowledged to provide a comprehensive perspective on the findings. The time-indexed formulation, although necessary for the MILP model to handle multiple visits to stations by both trucks and repairers, introduces a discrepancy between the discrete and continuous-time problems by approximating travel times through integer conversion, limiting the accuracy of the model in reflecting the continuous-time problem. Additionally, the SBC heuristic assumes that bicycles are (un)loaded or repaired directly at their respective stations without intermediate stops or second-time transfers. This simplification streamlines implementation but restricts the heuristic from leveraging potential efficiency gains through temporary inventories, which could yield better solutions. Future research will develop methodologies to address these limitations.

Declarations of interest: This research was jointly supported by grants from the National Natural Science Foundation of China (No. 71771194), the Research Grants Council of the Hong Kong Special Administrative Region of China (17206322), and a grant from the University Research Committee of the University of Hong Kong (2202100693).

CRediT authorship contribution statement

Runqiu Hu: Writing – original draft, Visualization, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **W.Y. Szeto:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition. **Sin C. Ho:** Writing – review & editing, Formal analysis, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

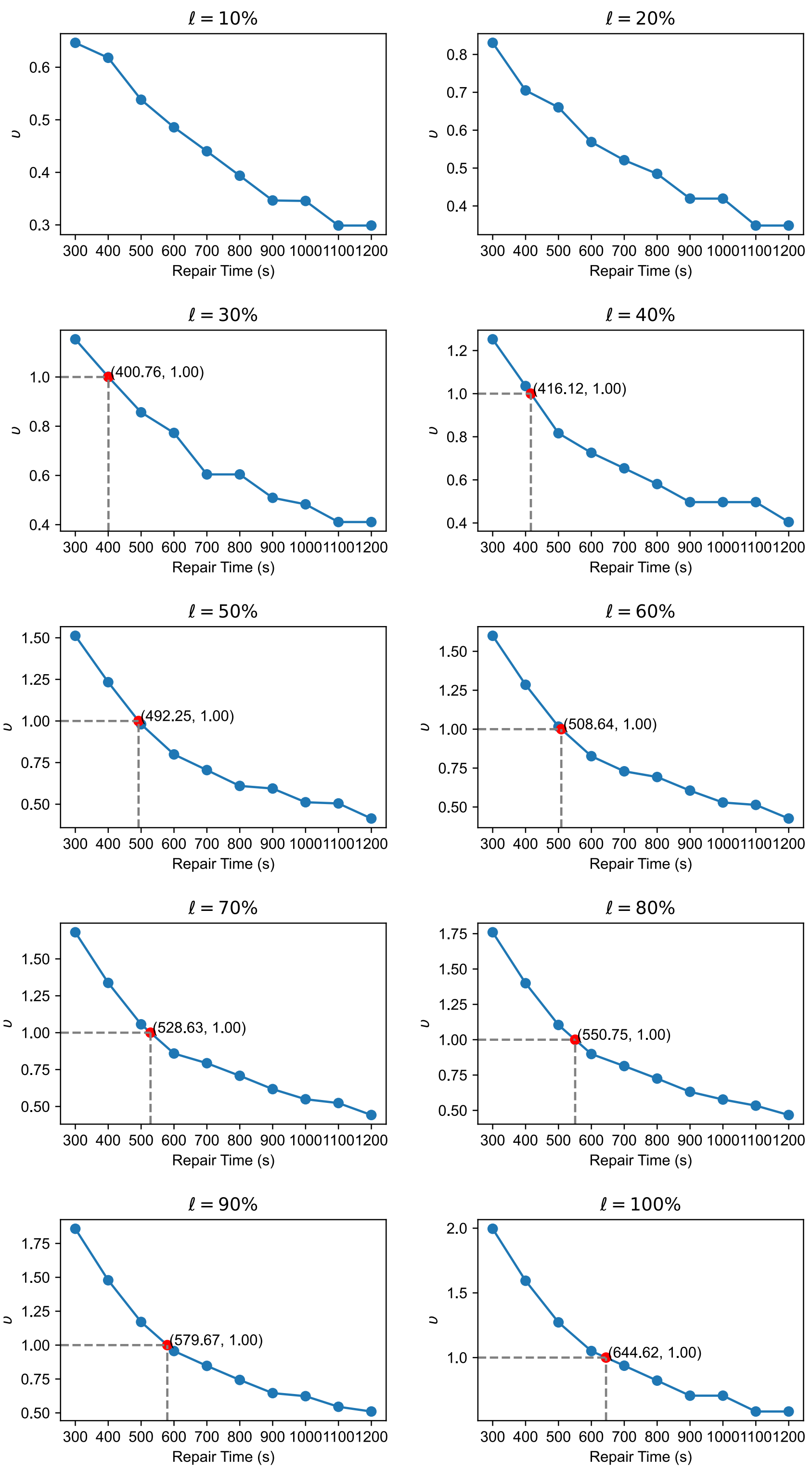


Fig. 5.2. Variations in the cost-effectiveness ratio over repair times under different broken bike inventories.

Acknowledgments

This research was jointly supported by grants from the National Natural Science Foundation of China (No. 71771194), the Research Grants Council of the Hong Kong Special Administrative Region of China (17206322), and a grant from the University Research Committee of the University of Hong Kong (2202100693). The authors are grateful to the constructive comments from the reviewers.

Appendix A. Linearization of the EUDF

The Extended User Dissatisfaction Function (EUDF) is a bivariate function that represents a weighted sum of the expected shortages of usable bikes and the expected shortages of lockers at a bike-sharing station in a day. It is calculated based on the quantities of both usable and unusable bikes at the station. The calculation of the EUDF values follows a Markov process, which captures the dynamics of bicycle inventory levels over time. For detailed steps for how the EUDF values are calculated, readers can refer to the approach outlined in the paper of Kaspi et al. (2017).

Let $F_i(p_i', b_i')$ be the EUDF, where p_i' and b_i' are the usable and broken inventory levels at the station, respectively. The domain of EUDF at each station i is $\Theta_i = \{(p_i', b_i') | p_i' \geq 0, b_i' \geq 0, p_i' + b_i' \leq C_i\}$, where C_i is the station capacity. For each point $(p_i', b_i') \in \Theta_i$, a plane $\alpha^i_{p_i,b_i} \cdot p_i' + \beta^i_{p_i,b_i} \cdot b_i' + \gamma^i_{p_i,b_i}$ that passes as close as possible to $F_i(p_i', b_i')$ can be fitted, where $\alpha^i_{p_i,b_i}, \beta^i_{p_i,b_i}$, and $\gamma^i_{p_i,b_i}$ are the coefficients of the plane that can be computed for every $(p_i, b_i) \in \Theta_i$. These coefficients are obtained by solving a linear programming (LP) model which ensures that the polyhedral function$\widetilde{F}_i(p_i', b_i') = \max_{(p_i,b_i)\in\theta}\left(\alpha^i_{p_i,b_i} \cdot p_i' + \beta^i_{p_i,b_i} \cdot b_i' + \gamma^i_{p_i,b_i}\right)$remains below or equal to the EUDF values at all points except for $(p_i, b_i) \in \theta_i$, with the objective of minimizing the gap between the fitted function and the actual EUDF at each point. The LP model is given as follows:

Decision variables:

$\alpha^i_{p_i,b_i}, \beta^i_{p_i,b_i}, \gamma^i_{p_i,b_i}$ Coefficients of the fitted plane

s Gap at the point (p_i, b_i) between the fitted plane and the EUDF value

The LP formulation:

Minimize s (A.74)

s.t.

$$\alpha^i_{p_i,b_i} \cdot p_i' + \beta^i_{p_i,b_i} \cdot b_i' + \gamma^i_{p_i,b_i} \leq F(p_i', b_i') \quad \forall (p_i', b_i') \in \theta \backslash \{(p_i, b_i)\} \tag{A.75}$$

$$\alpha^i_{p_i,b_i} \cdot p_i + \beta^i_{p_i,b_i} \cdot b_i + \gamma^i_{p_i,b_i} = F(p_i, b_i) - s \tag{A.76}$$

$$\alpha^i_{p_i,b_i}, \beta^i_{p_i,b_i}, \gamma^i_{p_i,b_i} \quad \text{free,} \tag{A.77}$$

$$s \geq 0 \tag{A.78}$$

The objective (A.74) is to minimize the gap between the fitted plane and the EUDF value. Constraints (A.75) ensure that the fitted plane passes under the EUDF at all integer points except for (p_i, b_i). Constraint (A.76) defines the gap at the point (p_i, b_i) between the fitted plane and the EUDF value. Constraints (A.77) and (A.78) are the domain constraints for the decision variables.

After the polyhedral function $\widetilde{F}_i(p_i, b_i)$ is formed at each point $(p_i, b_i) \in \Theta_i$, we replace the EUDF $F_i\left(p_{i,T'}, b_{i,T'}\right)$ in the objective function with an auxiliary variable $\widehat{F}_i$ and add the following constraints to the model:

$$\widehat{F}_i \geq \alpha^i_{p_i,b_i} \cdot p_{i,T'} + \beta^i_{p_i,b_i} \cdot b_{i,T'} + \gamma^i_{p_i,b_i}, \forall (p_i, b_i) \in \{(p_i, b_i) \in \mathbb{Z}^2 | p_i \geq 0, b_i \geq 0, p_i + b_i \leq C_i\}, \forall i \in S. \tag{A.79}$$

Appendix B. SBC heuristic for the quantity assignment

Algorithm 2: $SBCAssignment(S_R, S_K)$

```
1   $p_s = p_s^0, b_s = b_s^0, \forall s \in S$// initialization
    // lines 2 to 9: repairing quantity assignment
2   for $i = 1$ to $|R|$ do
3       // assign the repairing budget for repairer $i$ at each visited station (Algorithm 3)
        $\left(maxTime^r_{s,i}\right)_{s\in S} = allocateRepairBudget(r_i)$
        // determine the repairing quantity for repairer $i$ at each visited station in route $r_i$ (Algorithm 5)
4       $\left(g_{s,i}\right)_{s\in S} = assignRepair\left(r_i, \left(maxTime^r_{s,i}\right)_{s\in S}\right)$
8       /* lines 5 to 8: station inventory update (create a duplicate of the usable and broken bike inventory after repairs in $p_s'$ and $b_s'$, respectively, in preparation for restoring the bike inventories to the state after repairs before the second-round truck (un)loading assignment (Line 37) */
```

(continued on next page)

(*continued*)

Algorithm 2: *SBCAssignment*$(S_{\mathrm{R}}, S_{\mathrm{K}})$

```
5         for $s = 1$ to $|S|$ do
6             $p_s = p_s + g_{s,i}, p'_s = p_s$
7             $b_s = b_s - g_{s,i}, b'_s = b_s$
8         end for
9     end for
      // lines 10 to 36: truck (un)loading quantity assignment (the first round)
10    for $k = 1$ to $|K|$ do
11        $unsat^{\text{load,u}}_{s,k} = unsat^{\text{unload,u}}_{s,k} = unsat^{\text{load,b}}_{s,k} = 0, \forall s \in S$// initialization
          /* assign the station (un)loading budget for truck $k$ at each station (Algorithm 4) and record it in $\left(maxTimeInit^{\text{t}}_{s,k}\right)_{s\in S}$ */
12        $\left(maxTimeInit^{\text{t}}_{s,k}\right)_{s\in S} = allocateLoadingBudget\left(r'_k\right)$
13        $\left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S} = \left(maxTimeInit^{\text{t}}_{s,k}\right)_{s\in S}$ // initialization
          /* at the first departure from the depot, we load no usable bikes by default, and the adjustment on depot loading quantities is made in Algorithm 8 when we assign unloading quantities at subsequent deficit stations */
14        $y^{\text{u}^+}_{0,k} = y^{\text{u}^-}_{0,k} = y^{\text{b}^+}_{0,k} = y^{\text{b}^-}_{0,k} = 0, \psi^{\text{u}}_{0,k} = \psi^{\text{b}}_{0,k} = 0$
          // lines 15 to 33: assign loading and unloading quantities for each node in the route
15        for $l = 1$ to $|r'_k| - 1$ do
16            $\psi^{\text{u}}_{l,k} = \psi^{\text{b}}_{l,k} = 0$
17            $s' = s'_{k_l}$
18            $y^{\text{u}^+}_{l,k} = y^{\text{u}^-}_{l,k} = y^{\text{b}^+}_{l,k} = y^{\text{b}^-}_{l,k} = 0$// all (un)loading quantities are initialized with 0
19            if $s' = 0$ then
                  /* at intermediate visits to the depot, we load no usable bikes by default, and the adjustment on the depot (un)loading quantities is made in Algorithm 8 when we assign unloading quantities at subsequent deficit stations */
20                $y^{\text{u}^+}_{l,k} = \psi^{\text{u}}_{l-1,k}, y^{\text{b}^+}_{l,k} = \psi^{\text{b}}_{l-1,k}$
21            else
22                if $p_{s'} > q_{s'}$ then // for surplus stations, load usable bikes (Algorithm 6).
23                    $\left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{load,u}}_{s,k}\right)_{s\in S}, y^{\text{u}^-}_{l,k} = assignLoadU\left(r'_k, l, \left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{load,u}}_{s,k}\right)_{s\in S}\right)$
24                else
25                    if $p_{s'} < q_{s'}$ then // for deficit stations, unload usable bikes (Algorithms 7–8)
26                        $\left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{load,u}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{unload,u}}_{s,k}\right)_{s\in S}, y^{\text{u}^+}_{l,k} = assignUnloadU\left(r'_k, l, \left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{load,u}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{unload,u}}_{s,k}\right)_{s\in S}\right)$
27                    end if
28                end if
                  // station and truck inventory update
29                $p_{s'} = p_{s'} - y^{\text{u}^-}_{l,k} + y^{\text{u}^+}_{l,k}, \psi^{\text{u}}_{l,k} = \psi^{\text{u}}_{l,k} + y^{\text{u}^-}_{l,k} - y^{\text{u}^+}_{l,k}$
                  // for both types of stations, load broken bikes (Algorithm 9)
30                $\left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{load,b}}_{s,k}\right)_{s\in S}, y^{\text{b}^-}_{l,k} = assignLoadB\left(r'_k, l, \left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{unload,b}}_{s,k}\right)_{s\in S}\right)$
31                $b_{s'} = b_{s'} - y^{\text{b}^-}_{l,k}, \psi^{\text{b}}_{l,k} = \psi^{\text{b}}_{l,k} + y^{\text{b}^-}_{l,k}$// station and truck inventory update
32            end if
33        end for
          // $BCRF^{\text{t}}_s$ is updated here in preparation for the reassignment of the station (un)loading budget
34        Update $BCRF^{\text{t}}_s$ for each station based on the current station inventories
          /* adjust the station un(loading) budget of truck $k$ by distributing the unused time to unbalanced stations with bikes unable to be (un)loaded due to insufficient station (un)loading budgets (Algorithm 10) */
35        $\left(maxTimeNew^{\text{t}}_{s,k}\right)_{s\in S} = adjustLoadingBudget\left(\left(maxTimeInit^{\text{t}}_{s,k}\right)_{s\in S}, \left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{load,u}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{unload,u}}_{s,k}\right)_{s\in S}, \left(unsat^{\text{load,b}}_{s,k}\right)_{s\in S}\right)$
36    end for
      /* restore the station inventories to the inventories before the first-round truck (un)loading quantity assignment */
37    $p_s = p'_s, b_s = b'_s$
38    for $k = 1$ to $|K|$ do
          /* assign the quantities for bike (un)loading for the second round using the updated station (un)loading budget */
39        Set $\left(maxTimeRemain^{\text{t}}_{s,k}\right)_{s\in S} = \left(maxTimeNew^{\text{t}}_{s,k}\right)_{s\in S}$ and repeat the procedure from Line 15 to Line 33
40    end for
```

(*continued on next page*)

(*continued*)

Algorithm 2: *SBCAssignment*(S_R, S_K)

```
41  Conduct amendments to both trucks and repairers regarding the routes and their associated repairing or (un)loading quantities using the algorithm described in Section 4.4.4.
42  return $G_R$, $Y_K$
```

The algorithm begins by assigning quantities of repairs for each repairer and updating the station inventories (Line 1 to Line 9). Following this, the algorithm iteratively traverses the route of each truck, determining the quantities of loading/unloading based on the status of each station (Line 10 to Line 40). A key feature of this SBC heuristic is setting a maximum time for bike loading/unloading for every truck and a maximum time for bike repairing for every repairer, referred to as the *station budget*. Specifically, we refer to the budget at each station as the *station (un)loading budget* for trucks and the *station repairing budget* for repairers. The introduction of the station budget prevents the truck/repairer from excessively spending time at the initial stations, which can lead to insufficient time for the latter stations. For each truck, the first round of (un)loading quantity assignments is conducted (Line 15 to Line 33), followed by an adjustment on the station (un)loading budgets based on unsatisfied demand and unused time budgets in the first round (Lines 34 and 35). Afterward, the algorithm restores the station inventories to the values before the truck (un)loading quantity assignment (Line 37) and conducts a second-round assignment to adjust the truck (un)loading quantities (Line 38 to Line 40). Next, amendments to both trucks and repairers regarding the routes and their associated repairing or (un)loading quantities are made to utilize the unused time budget resulting from the rounding-down strategies applied during quantity assignments (Line 41). Finally, the algorithm returns the assigned repairing and (un)loading quantities for usable and broken bikes (Line 42).

Appendix C. Assignment algorithms for station repairing budget and station (un)loading budget

Algorithm 3: *allocateRepairBudget*(r_i)

```
1   Initialization: $maxTime^{r}_{s,i} = 0, \forall s \in S$
2   $T^{op} = T - \sum_{j=0}^{|r_i|-1} t^{r}_{s_{ij}, s_{ij+1}}$ // total time budget left for repairing bikes
3   while $T^{op} < t^{repair}$ do
        // remove the station with the lowest benefit-cost ratio function value from the route
4       Remove station $s' = \text{argmin}_{s \in S} BCRF^{r}_{s}$ from $r_i$
5       $T^{op} = T - \sum_{j=0}^{|r_i|-1} t^{r}_{s_{ij}, s_{ij+1}}$
6   end while
    /* the set unsat stores the stations where broken bikes cannot be all repaired due to insufficient station repairing budget */
7   $unsat = \emptyset$
    // record the overly assigned station repairing budget
8   $extraTime = 0$
    // lines 9 to 18: assign the maximum repairing quantity at each station ($1 \le j \le |r_i| - 1$)
9   for $j$ in 1 to $|r_i| - 2$ do
        // distribute the total station repairing budget using the ratio of the BCRF value of each station
10      $maxTime^{r}_{s_{ij}, i} = \dfrac{T^{op} \times BCRF^{r}_{s_{ij}}\left(p_{s_{ij}}, b_{s_{ij}}\right)}{\sum_{j'=1}^{|r_i|-2} BCRF^{r}_{s_{ij'}}\left(p_{s_{ij'}}, b_{s_{ij'}}\right)}$
11      $repair^{max}_{s_{ij}} = \min\left\{\max\left\{q_{s_{ij}} - p_{s_{ij}}, 0\right\}, b_{s_{ij}}\right\}$ // maximum repairing quantity
12      if $maxTime^{r}_{s_{ij}} < repair^{max}_{s_{ij}} \cdot t^{repair}$ then
            // record stations with insufficient repairing budgets to repair all broken bikes
13          $unsat = unsat \cup \{s_{ij}\}$
14      else
            // gather the excess repairing budget and adjust the repairing budget
15          $extraTime = extraTime + maxTime^{r}_{s_{ij}, i} - repair^{max}_{s_{ij}} \cdot t^{repair}$
16          $maxTime^{r}_{s_{ij}} = repair^{max}_{s_{ij}} \cdot t^{repair}$
17      end if
18  end for
    // lines 19 to 25: reassign the total excess time to nodes with insufficient budgets
19  while $extraTime > t^{repair}$ and $unsat \neq \emptyset$ do
20      $\hat{s}$ = The node in unsat with the highest $BCRF^{r}_{s}$ (break ties by choosing the smallest $\hat{s}$)
21      $added = \min\left\{extraTime, repair^{max}_{\hat{s}} \cdot t^{repair} - maxTime^{r}_{\hat{s},i}\right\}$
22      $maxTime^{r}_{\hat{s},i} = maxTime^{r}_{\hat{s},i} + added$
23      $extraTime = extraTime - added$
24      $unsat = unsat \setminus \{\hat{s}\}$
25  end while
26  return $\left(maxTime^{r}_{s,i}\right)_{s \in S}$
```

The algorithm begins by computing the total time available for repairs T^{op} by deducting travel time between stations from the total time budget T (Line 2). It then conducts a feasibility check and route pruning to ensure that T^{op} is sufficient for at least one repair (Line 3 to Line 6). Then, the algorithm assigns the total time available to each station based on $BCRF^{r}_{s}$, and determines the maximum possible number of repairs within the allocated time. Stations where the time is insufficient are added to *unsat*, while the excess time budget is added to *extraTime*(Line 7 to Line 18). If there is extra time left (i.e., $extraTime > 0$) and *unsat* is not empty, the algorithm redistributes the extra time to stations in *unsat*, prioritizing those with the highest $BCRF^{r}_{s}$ until all *extraTime* is allocated or *unsat* is empty (Line 19 to Line 25). Finally, the algorithm returns the station repairing budget for each station.

```
Algorithm 4: allocateLoadingBudget($r'_k$)
1   Initialization:$maxTime^{t}_{s,k} = 0, \forall s \in S$
2   $T^{op} = T - \sum_{l=0}^{|r'_k|-1} t^{t}_{s'_{k_l}, s'_{k_{l+1}}}$ // The total time budget left for truck $k$ to load and unload bikes
3   while $T^{op} < 2t^{load}$ do
        // remove the station with the lowest benefit-cost ratio function value from the route
4       Remove station $s' = \mathrm{argmin}_{s \in S} BCRF^{t}_{s}$ from $r'_k$
5       $T^{op} = T - \sum_{l=0}^{|r'_k|-1} t^{t}_{s'_{k_l}, s'_{k_{l+1}}}$
6   end while
7   $unsat = \emptyset$
8   $extraTime = 0$
9   $NS$ = The set of stations in $r'_k$
10  for $s$ in $NS$ do
11      $maxTime^{t}_{s,k} = \frac{T^{op} \times BCRF^{t}_{s}(p_s, b_s)}{\sum_{s' \in NS} BCRF^{t}_{s'}(p_{s'}, b_{s'})}$
12      $load^{max}_{s} = |p_s - q_s| + b_s$
13      if $maxTime^{t}_{s,k} < 2t^{load} \cdot load^{max}_{s}$ then
14          $unsat = unsat \cup \{s\}$
15      else
16          $extraTime = extraTime + maxTime^{t}_{s,k} - 2t^{load} \cdot load^{max}_{s}$
17          $maxTime^{t}_{s,k} = 2t^{load} \cdot load^{max}_{s}$
18      end if
19  end for
20  while $extraTime > 2t^{load}$ and $unsat \neq \emptyset$ do
21      $\hat{s}$ = The node in $unsat$ with the highest $BCRF^{t}_{s}$
22      $added = \min\left\{extraTime, 2t^{load} \cdot load^{max}_{\hat{s}} - maxTime^{t}_{\hat{s},k}\right\}$
23      $maxTime^{t}_{\hat{s},k} = maxTime^{t}_{\hat{s},k} + added$
24      $extraTime = extraTime - added$
25      $unsat = unsat \setminus \{\hat{s}\}$
26  end while
27  return $\left(maxTime^{t}_{s,k}\right)_{s \in S}$
```

Note that for a visited surplus station, we must consider that each bike loaded onto the truck also needs to be unloaded at another node (either the depot or a deficit station). Consequently, we set the station (un)loading budget as twice the loading time for each bike at this station. For a visited deficit station, at the current stage, we assume that all bikes being unloaded are additionally loaded from the depot. Therefore, we also set the station (un)loading budget as twice the loading time for each bike at this station. It is obvious that this approach leads to an overestimation, as it double-counts the time for bikes that are simply being transferred from surplus to deficit stations. To correct this, we make appropriate adjustments to the station (un)loading budgets during the detailed station-by-station assignment process, as outlined in **Algorithms 6, 7**, and **8** in subsequent sections.

Appendix D. Repairing, loading, and unloading quantity assignment algorithms

We start by introducing the repairing quantity assignment. The algorithm takes the route of repairman *i* and the station repairing budget as the input and returns the number of broken bikes repaired by repairman *i* at each station. For surplus stations, no bike is repaired at the current stage. For deficit stations, the repairing quantity is constrained by the deviation of usable bike inventory from the target level and the station repairing budget.

```
Algorithm 5: assignRepair($r_i, \left(maxTime^{r}_{s,i}\right)_{s \in S}$)
1   Initialization:$g_{s,i} = 0, \forall s \in S$
2   for $j = 1$ to $|r_i| - 2$ do
3           $g_{s_{ij},i} = \min\left\{\max\left\{q_{s_{ij}} - p_{s_{ij}}, 0\right\}, b_{s_{ij}}, \lfloor maxTime^{r}_{s_{ij},i} / t^{repair} \rfloor\right\}$
4   end for
5   return $\left(g_{s,i}\right)_{s \in S}$
```

Next, we present the algorithms for (un)loading quantity assignment of trucks, including usable bike loading (**Algorithm 6**), usable bike unloading (**Algorithms 7 and 8**), and broken bike loading (**Algorithm 9**). Note that we use *l* to represent the order index for defining when the concerned node is visited by truck and assigned a loading quantity in these four algorithms.

Algorithm 6: $assignLoadU\left(r'_k, l, \left(maxTimeRemain^{t}_{s,k}\right)_{s\in S}, \left(unsat^{load,u}_{s,k}\right)_{s\in S}\right)$

```
1   $s = s'_{k_l}$
2   $n = \min\left\{Q_k - \psi^{u}_{l-1,k} - \psi^{b}_{l-1,k}, p_s - q_s\right\}$ // the quantity without the station loading budget
3   $y^{u^-}_{l,k} = \min\left\{n, \lfloor maxTimeRemain^{t}_{s,k}/2t^{load} \rfloor\right\}$ // the quantity with the station loading budget
    // Record the number of bikes not loaded due to the limited station loading budget
4   $unsat^{load,u}_{s,k} = unsat^{load,u}_{s,k} + \left(n - y^{u^-}_{l,k}\right)$
5   $maxTimeRemain^{t}_{s,k} = maxTimeRemain^{t}_{s,k} - 2t^{load} \cdot y^{u^-}_{l,k}$
6   return $\left(maxTimeRemain^{t}_{s,k}\right)_{s\in S}, \left(unsat^{load,u}_{s,k}\right)_{s\in S}, y^{u^-}_{l,k}$
```

Algorithm 6 calculates the loading quantity of usable bikes for truck *k* at a given node. The number of bikes loaded is constrained by the truck's capacity, the station's surplus inventory, and the station loading budget (Line 3). The algorithm records the unsatisfied loading quantity due to the limited station loading budget (Line 4) and updates the station loading budget (Line 5). Finally, it returns the updated remaining station (un)loading budget, unsatisfied loading quantity due to the limited station loading budget, and the number of bikes loaded onto truck *k* (Line 6).

Algorithm 7: $assignUnloadU\left(r'_k, l, \left(maxTimeRemain^{t}_{s,k}\right)_{s\in S}, \left(unsat^{load,u}_{s,k}\right)_{s\in S}, \left(unsat^{unload,u}_{s,k}\right)_{s\in S}\right)$

```
1   $s = s'_{k_l}$
2   $y^{u^+}_{l,k} = \min\left\{\psi^{u}_{l-1,k}, C_s - p_s - b_s, q_s - p_s\right\}$
    /* add back the double-counted loading time as these unloaded bikes are not from the depot but from the previously visited surplus stations */
3   $maxTimeRemain^{t}_{s,k} = maxTimeRemain^{t}_{s,k} + 2t^{load} \cdot y^{u^+}_{l,k}$
    // target inventory not reached due to insufficient bike supply on the truck
4   if $y^{u^+}_{l,k} < q_s - p_s$ and $y^{u^+}_{l,k} \neq C_s - p_s - b_s$ then
        /* line 5 to 20: load more bikes from the visited but still surplus stations and unload them to the current station until the target inventory is reached or the station loading budget is used up */
5       while $\exists \hat{s} \in S : unsat^{load,u}_{\hat{s},k} > 0$ and $maxTimeRemain^{t}_{s,k} \geq 2t^{load}$ do
6           $s'$ = the station with the highest $BCRF^{t}_{\hat{s}}$ and $unsat^{load,u}_{s',k} > 0$ (break ties by choosing the smallest $s'$)
7           $l'$ = the index of station $s'$ in the route $r'_k$ (break ties by choosing the one with the largest order index)
            // minimum truck residual capacity from the chosen node to the current node
8           $rc_k = \min_{l''=l',l'+1,\dots,l-1} Q_k - \psi^{u}_{l'',k} - \psi^{b}_{l'',k}$
            // the extra loaded quantity with no station loading budget
9           $n = \min\left(q_s - p_s - y^{u^+}_{l,k}, rc_k, unsat^{load,u}_{s',k}\right)$
            // the extra loaded quantity with the remaining station loading budget
10          $\Lambda^{load,u} = \min\left\{n, \lfloor maxTimeRemain^{t}_{s,k}/2t^{load} \rfloor\right\}$
11          $unsat^{unload,u}_{s,k} = unsat^{unload,u}_{s,k} + \left(n - \Lambda^{load,u}\right)$// update the unsatisfied unloading quantity
            // update the remaining station loading budget of the current station
12          $maxTimeRemain^{t}_{s,k} = maxTimeRemain^{t}_{s,k} - 2t^{load} \cdot \Lambda^{load,u}$
13          $unsat^{load,u}_{s',k} = unsat^{load,u}_{s',k} - \Lambda^{load,u}$// update the unsatisfied loading quantity
14          $p_{s'} = p_{s'} - \Lambda^{load,u}$// update the station inventory after extra loading of bikes
15          for $l''$ in $l'$ to $l-1$ do // update the truck inventories
16              $\psi^{u}_{l'',k} = \psi^{u}_{l'',k} + \Lambda^{load,u}$
17          end for
18          $y^{u^-}_{l',k} = y^{u^-}_{l',k} + \Lambda^{load,u}$// update the loading quantity
19          $y^{u^+}_{l,k} = y^{u^+}_{l,k} + \Lambda^{load,u}$// update the unloading quantity
20      end while
21      if $y^{u^+}_{l,k} < q_s - p_s$ then// the target inventory is still not reached
22          $dv = \max\left\{dv \middle| dv < l, s'_{k_{dv}} = 0\right\}$ // order index of the last visited depot
23          // adjust the loading at the previously visited depot (Algorithm 8)
24          $h, maxTimeRemain^{t}_{s,k}, unsat^{unload,u}_{s,k} = adjustDepotLoading\left(dv, l, maxTimeRemain^{t}_{s,k}, unsat^{unload,u}_{s,k}\right)$
```

(continued on next page)

(*continued*)

```
Algorithm 7: assignUnloadU$\left(r'_k, l, \left(maxTimeRemain^{t}_{s,k}\right)_{s\in S}, \left(unsat^{load,u}_{s,k}\right)_{s\in S}, \left(unsat^{unload,u}_{s,k}\right)_{s\in S}\right)$
25            $y^{u^+}_{l,k} = y^{u^+}_{l,k} + h$
26        end if
27    end if
28    return $\left(maxTimeRemain^{t}_{s,k}\right)_{s\in S}, \left(unsat^{load,u}_{s,k}\right)_{s\in S}, \left(unsat^{unload,u}_{s,k}\right)_{s\in S}, y^{u^+}_{l,k}$
```

Algorithm 7 determines the unloading quantity by considering the truck's usable bike inventory, the station's residual capacity, and the deviation of the inventory from the target inventory levels (Line 2). If the usable bike shortage cannot be satisfied due to insufficient usable bikes on the truck, the algorithm utilizes the unsatisfied loading quantities from previously visited stations that are still in a surplus state and tries to load more bikes at those stations to satisfy the shortage at the current station (Lines 5 to Line 20). Note that during this process, we also capture the number of usable bikes unable to be loaded from other nodes by truck k due to the insufficient station loading budget (Line 9 to Line 11). If the station's demand remains unmet, a final adjustment is made on the previous depot visit (Line 21 to Line 26) by reducing the number of usable bikes unloaded or loading more usable bikes there. The detailed procedure for adjustment at the depot is presented in **Algorithm 8**. Note that we use dv to denote the order index for the last visited depot in r'_k that requires (un)loading adjustment, h to denote the increased loading of usable bikes at the station that requires (un)loading adjustment, and s is the station that requires (un)loading adjustment from the previous depot.

```
Algorithm 8: adjustDepotLoading$\left(dv, l, maxTimeRemain^{t}_{s,k}, unsat^{unload,u}_{s,k}\right)$
1    $rc_k = \min_{l'=dv,dv+1,\dots,l-1} Q_k - \psi^{u}_{l',k} - \psi^{b}_{l',k}$
2    $n = \min\left\{rc_k, q_s - p_s - y^{u^+}_{l,k}\right\}$
3    $h = \min\left\{n, \lfloor maxTimeRemain^{t}_{s,k}/2t^{load} \rfloor\right\}$
4    $unsat^{unload,u}_{s'_{k_l},k} = unsat^{unload,u}_{s'_{k_l},k} + (n-h)$
5    if $y^{u^+}_{dv,k} > 0$ then// the original operation at the depot is unloading
6        $y^{u^+}_{dv,k} = \max\left\{y^{u^+}_{dv,k} - h, 0\right\}$ // reduce the unloading of usable bikes at the depot
         // if the reduction exceeds the unloading quantity, the final decision at the depot becomes loading
7        $y^{u^-}_{dv,k} = \max\left\{h - y^{u^+}_{dv,k}, 0\right\}$
8        $maxTimeRemain^{t}_{s'_{k_l},k} = maxTimeRemain^{t}_{s'_{k_l},k} - 2t^{load}\cdot y^{u^-}_{dv,k}$
9    end if
10   if $y^{u^-}_{dv,k} > 0$ then// the original operation at the depot is loading
11       $y^{u^-}_{dv,k} = y^{u^-}_{dv,k} + h$// increase the loading quantity
12       $maxTimeRemain^{t}_{s,k} = maxTimeRemain^{t}_{s,k} - 2t^{load}\cdot h$
13   end if
14   for $l' = dv$ to $l-1$ do
15       $\psi^{u}_{l',k} = \psi^{u}_{l',k} + h$
16   end for
17   return $h$, $maxTimeRemain^{t}_{s,k}$, $unsat^{unload,u}_{s,k}$
```

Algorithm 8 initiates by calculating the minimum residual capacity of the truck from the last visited depot to the upcoming station $s = s'_{k_l}$ (Line 1). It then computes the additional number of bikes that need to be loaded at the depot, which is the lowest value among the truck's minimum residual capacity, the shortage of usable bikes at the upcoming station, and the number of additional bikes that can be loaded using the station loading budget of the upcoming deficit station (Lines 2–3). If the initial operation at the depot is to unload bikes, the algorithm decreases the unloading quantity by the newly calculated amount. If the adjustment leads to a surplus, the operation at the depot is switched from unloading to loading. Conversely, if the depot was initially set to load bikes, the algorithm simply increases the loading quantity by the computed adjustment (Line 5 to Line 13). The algorithm terminates by updating the truck inventory along the route from the depot to the upcoming deficit station and returns the adjustment value. During the process, the unsatisfied unloading demand is limited by station unloading budgets and the budgets are updated accordingly (Line 4 and Line 12).

After usable bikes are loaded or unloaded, we continue to assign the quantities of broken bikes to be loaded using **Algorithm 9**. The assignment follows a similar procedure as **Algorithm 6**, with the loading quantities of broken bikes constrained by the residual capacity of the truck, the broken bike inventory at the station, and the station loading budget.

Algorithm 9: *assignLoadB*$\left(r'_k, l, \left(maxTimeRemain^{\mathrm{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\mathrm{load,b}}_{s,k}\right)_{s\in S}\right)$

```
1  $s = s'_{k_l}$
2  $n = \min\left\{b_s, Q_k - \psi^{\mathrm{u}}_{l-1,k} - \psi^{\mathrm{b}}_{l-1,k}\right\}$
3  $y^{\mathrm{b}^-}_{l,k} = \min\left\{n, \lfloor maxTimeRemain^{\mathrm{t}}_{s,k}/2t^{\mathrm{load}} \rfloor\right\}$
4  // The number of broken bikes not loaded due to the limited station loading budget
   $unsat^{\mathrm{load,b}}_{s,k} = unsat^{\mathrm{load,b}}_{s,k} + \left(n - y^{\mathrm{b}^-}_{l,k}\right)$
5  $maxTimeRemain^{\mathrm{t}}_{s,k} = maxTimeRemain^{\mathrm{t}}_{s,k} - 2t^{\mathrm{load}} \cdot y^{\mathrm{b}^-}_{l,k}$
6  return $\left(maxTimeRemain^{\mathrm{t}}_{s,k}\right)_{s\in S}, \left(unsat^{\mathrm{load,b}}_{s,k}\right)_{s\in S}, y^{\mathrm{b}^-}_{l,k}$
```

Appendix E. (Un)loading budget adjustment algorithm for trucks

Algorithm 10 shows the procedure of the (un)loading budget adjustment.

Algorithm 10: *adjustLoadingBudget*$\left(\begin{array}{c}\left(maxTimeInit^{\mathrm{t}}_{s,k}\right)_{s\in S}, \left(maxTimeRemain^{\mathrm{t}}_{s,k}\right)_{s\in S}, \\ \left(unsat^{\mathrm{load,u}}_{s,k}\right)_{s\in S}, \left(unsat^{\mathrm{unload,u}}_{s,k}\right)_{s\in S}, \left(unsat^{\mathrm{load,b}}_{s,k}\right)_{s\in S}\end{array}\right)$

```
1  $unusedTime_k = 0$// the variable that stores the time budget that is assigned but not used
   /* lines 2 to 6: add up all the bikes not loaded/unloaded at each station due to insufficient station (un)loading budgets and store the result in $unsat_{\hat{s},k}$, subtract the overly assigned station (un)loading budgets, and accumulate them into $unusedTime_k$*/
2  for $\hat{s}$ in $S$ do
3      $unsat_{\hat{s},k} = unsat^{\mathrm{load,u}}_{\hat{s},k} + unsat^{\mathrm{load,b}}_{\hat{s},k} + unsat^{\mathrm{unload,u}}_{\hat{s},k}$
4      $maxTimeNew^{\mathrm{t}}_{\hat{s},k} = maxTimeInit^{\mathrm{t}}_{\hat{s},k} - maxTimeRemain^{\mathrm{t}}_{\hat{s},k}$
5      $unusedTime_k = unusedTime_k + maxTimeRemain^{\mathrm{t}}_{\hat{s},k}$
6  end for
7  while $unusedTime_k \geq 2t^{\mathrm{load}}$ and $\exists \hat{s} \in S : unsat_{\hat{s},k} > 0$ do
8      $s'$ = The station that satisfies $unsat_{\hat{s},k} > 0$ with the highest $BCRF^{\mathrm{t}}_{\hat{s}}$ (break ties by choosing the smallest $\hat{s}$)
9      $\Delta = \min\left\{unusedTime_k, 2t^{\mathrm{load}} \cdot unsat_{s',k}\right\}$
10     $maxTimeNew^{\mathrm{t}}_{s',k} = maxTimeNew^{\mathrm{t}}_{s',k} + \Delta$
11     $unusedTime_k = unusedTime_k - \Delta$
12     $unsat_{s',k} = unsat_{s',k} - \lfloor \Delta/2t^{\mathrm{load}} \rfloor$
13 end while
14 return $\left(maxTimeNew^{\mathrm{t}}_{s,k}\right)_{s\in S}$
```

Algorithm 10 begins with a traversal of each station in the network, counting the total number of bikes not (un)loaded at each station due to insufficient (un)loading budgets, reducing the overly assigned budget from the initial station (un)loading budget, and sum up these unused times (Line 1 to Line 6). The algorithm then enters a loop where it redistributes unused time budget to stations with unsatisfied loading or unloading needs, i.e., the bikes that could not be loaded or unloaded due to limited station (un)loading budgets (Line 7 to Line 13). Within each iteration, the algorithm selects the station with the highest $BCRF^{\mathrm{t}}_{\hat{s}}$ and allocates additional time from the unused budget to make up for this insufficiency (Line 8 to Line 12). This process is repeated until there is either no more unused time or all the demands have been met. Finally, the updated station (un)loading budgets are returned (Line 14).

Data availability

Data will be made available on request.

References

Alvarez-Valdes, R., Belenguer, J.M., Benavent, E., Bermudez, J.D., Muñoz, F., Vercher, E., Verdejo, F., 2016. Optimizing the level of service quality of a bike-sharing system. Omega 62, 163–175.

Brinkmann, J., Ulmer, M.W., Mattfeld, D.C., 2019. Dynamic lookahead policies for stochastic-dynamic inventory routing in bike sharing systems. Comput. Oper. Res. 106, 260–279.

Caggiani, L., Camporeale, R., Ottomanelli, M., Szeto, W.Y., 2018. A modeling framework for the dynamic management of free-floating bike-sharing systems. Transp. Res. Part C: Emerg. Technol. 87, 159–182.

Caggiani, L., Ottomanelli, M., 2012. A modular soft computing based method for vehicles repositioning in bike-sharing systems. Procedia-Soc. Behav. Sci. 54, 675–684.

Caggiani, L., Ottomanelli, M., 2013. A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems. Procedia-Soc. Behav. Sci. 87, 203–210.

Cai, Y., Ong, G.P., Meng, Q., 2022. Dynamic bicycle relocation problem with broken bicycles. Transp. Res. Part E: Logist. Transp. Rev. 165, 102877.
Castro Fernández, A., 2011. The contribution of bike-sharing to sustainable mobility in Europe. Vienna University of Technology. Ph.D. Thesis.
Chang, S., Song, R., He, S., Qiu, G., 2018. Innovative bike-sharing in China: Solving faulty bike-sharing recycling problem. J. Adv. Transp. https://doi.org/10.1155/2018/4941029.
Chang, X., Wu, J., Sun, H., Correia, G., Chen, J., 2021. Relocating operational and damaged bikes in free-floating systems: A data-driven modeling framework for level of service enhancement. Transp. Res. Part A: Policy Pract. 153, 235–260.
Chang, X., Wu, J., Sun, H., Yan, X., 2023. A smart predict-then-optimize method for dynamic green bike relocation in the free-floating system. Transp. Res. Part C: Emerg. Technol. 153, 104220.
Chen, M., Szeto, W.Y., 2023. A static green bike repositioning problem with heavy and light carriers. Transp. Res. Part D: Transp. Environ. 118, 103711.
Chen, M., Wang, D., Sun, Y., Waygood, E.O.D., Yang, W., 2020. A comparison of users' characteristics between station-based bikesharing system and free-floating bikesharing system: case study in Hangzhou. China. Transportation 47 (2), 689–704.
Chen, Q., Ma, S., Li, H., Zhu, N., He, Q.-C., 2024. Optimizing bike rebalancing strategies in free-floating bike-sharing systems: An enhanced distributionally robust approach. Transp. Res. Part E: Logist. Transp. Rev. 184, 103477.
Cheng, Y., Wang, J., Wang, Y., 2021. A user-based bike rebalancing strategy for free-floating bike sharing systems: A bidding model. Transp. Res. Part E: Logist. Transp. Rev. 154, 102438.
Chiariotti, F., Pielli, C., Zanella, A., Zorzi, M., 2020. A bike-sharing optimization framework combining dynamic rebalancing and user incentives. ACM Trans. Auton. Adapt. Syst. 14 (3), 1–30.
Citi Bike, 2024. June 2024 monthly report. Available at: https://mot-marketing-whitelabel-prod.s3.amazonaws.com/nyc/June-2024-Citi-Bike-Monthly-Report.pdf (Accessed: 8 September 2024).
Contardo, C., Morency, C., Rousseau, L.-M., 2012. Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09. Montréal.
Dell'Amico, M., Iori, M., Novellani, S., Stützle, T., 2016. A destroy and repair algorithm for the bike sharing rebalancing problem. Comput. Oper. Res. 71, 149–162.
Dell'Amico, M., Iori, M., Novellani, S., Subramanian, A., 2018. The bike sharing rebalancing problem with stochastic demands. Transp. Res. Part b: Methodol. 118, 362–380.
Di Gaspero, L., Rendl, A., Urli, T., 2013a. A hybrid ACO+CP for balancing bicycle sharing systems. In: M.J. Blesa., C. Blum., P. Festa., A. Roli., M. Sampels. (Eds.), Hybrid Metaheuristics. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 198–212.
Di Gaspero, L., Rendl, A., Urli, T., 2013b. Constraint-based approaches for balancing bike sharing systems. In: C. Schulte. (Ed.), Principles and Practice of Constraint Programming. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 758–773.
Du, M., Cheng, L., Li, X., Tang, F., 2020. Static rebalancing optimization with considering the collection of malfunctioning bikes in free-floating bike sharing system. Transp. Res. Part E: Logist. Transp. Rev. 141, 102012.
Fang, W., Yang, X., Zhao, W., Liu, B., 2019. Stability analysis of articulated dump truck based on eigenvalue method. IOP Conf. Ser. Mater. Sci. Eng. 688 (2), 022024.
Fishman, E., Washington, S., Haworth, N., 2014. Bike share's impact on car use: Evidence from the United States, Great Britain, and Australia. Transp. Res. Part d: Transp. Environ. 31, 13–20.
Forma, I.A., Raviv, T., Tzur, M., 2015. A 3-step math heuristic for the static repositioning problem in bike-sharing systems. Transp. Res. Part B: Methodol. 71, 230–247.
Franceschetti, A., Honhon, D., Van Woensel, T., Bektaş, T., Laporte, G., 2013. The time-dependent pollution-routing problem. Transp. Res. Part B: Methodol. 56, 265–293.
Freund, D., 2018. Models and algorithms for transportation in the sharing economy. Cornell University. Ph.D. Thesis.
Guo, Y., Li, J., Xiao, L., Allaoui, H., Choudhary, A., Zhang, L., 2024. Efficient inventory routing for bike-sharing systems: A combinatorial reinforcement learning framework. Transp. Res. Part E: Logist. Transp. Rev. 182, 103415.
Gurobi Optimization, LLC, 2023. Gurobi optimizer reference manual. Available at: https://www.gurobi.com.
Ho, S.C., Szeto, W.Y., 2014. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. Transp. Res. Part E: Logist. Transp. Rev. 69, 180–198.
Ho, S.C., Szeto, W.Y., 2017. A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem. Transp. Res. Part B: Methodol. 95, 340–363.
Hu, R., Zhang, Z., Ma, X., Jin, Y., 2021. Dynamic rebalancing optimization for bike-sharing system using priority-based MOEA/D algorithm. IEEE Access 9, 27067–27084.
Huang, D., Chen, X., Liu, Z., Lyu, C., Wang, S., Chen, X., 2020. A static bike repositioning model in a hub-and-spoke network framework. Transp. Res. Part E: Logist. Transp. Rev. 141, 102031.
Jia, H., Miao, H., Tian, G., Zhou, M., Feng, Y., Li, Z., Li, J., 2020a. Multiobjective bike repositioning in bike-sharing systems via a modified artificial bee colony algorithm. IEEE Trans. Autom. Sci. Eng. 17 (2), 909–920.
Jia, Y., Zeng, W., Xing, Y., Yang, D., Li, J., 2020b. The bike-sharing rebalancing problem considering multi-energy mixed fleets and traffic restrictions. Sustainability 13 (1), 270.
Jin, Y., Ruiz, C., Liao, H., 2022. A simulation framework for optimizing bike rebalancing and maintenance in large-scale bike-sharing systems. Simul. Model Pract. Theory 115, 102422.
Kaspi, M., 2015. Bike-sharing systems: parking reservation policies and maintenance operations. Tel Aviv University. Ph.D. Thesis.
Kaspi, M., Raviv, T., Tzur, M., 2016. Detection of unusable bicycles in bike-sharing systems. Omega 65, 10–16.
Kaspi, M., Raviv, T., Tzur, M., 2017. Bike-sharing systems: User dissatisfaction in the presence of unusable bicycles. IISE Trans. 49 (2), 144–158.
Lee, E., Son, B., Han, Y., 2020. Optimal relocation strategy for public bike system with selective pick-up and delivery. Transp. Res. Rec.: J. Transp. Res. Board 2674 (4), 325–336.
Li, Y., Szeto, W.Y., Long, J., Shui, C.S., 2016. A multiple type bike repositioning problem. Transp. Res. Part B: Methodol. 90, 263–278.
Liu, Y., Szeto, W.Y., Ho, S.C., 2018. A static free-floating bike repositioning problem with multiple heterogeneous vehicles, multiple depots, and multiple visits. Transp. Res. Part C: Emerg. Technol. 92, 208–242.
Lu, H., Zhang, M., Su, S., Gao, X., Luo, C., 2019. Broken bike recycling planning for sharing bikes system. IEEE Access 7, 177354–177361.
Lu, L., Zhao, S., He, Q.-C., Zhu, N., 2022. Task assignment in predictive maintenance for free-float bicycle sharing systems. Comput. Ind. Eng. 169, 108214.
Lu, Y., Benlic, U., Wu, Q., 2020. An effective memetic algorithm for the generalized bike-sharing rebalancing problem. Eng. Appl. Artif. Intell. 95, 103890.
Luo, H., Zhao, F., Chen, W.-Q., Cai, H., 2020. Optimizing bike sharing systems from the life cycle greenhouse gas emissions perspective. Transp. Res. Part c: Emerg. Technol. 117, 102705.
O'Mahony, E., Shmoys, D.B., 2015. Data analysis and optimization for (Citi)bike sharing. In: In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 687–694.
Pan, L., Cai, Q., Fang, Z., Tang, P., Huang, L., 2019. A deep reinforcement learning framework for rebalancing dockless bike sharing systems. In: In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, pp. 1393–1400.
Rainer-Harbach, M., Papazek, P., Raidl, G.R., Hu, B., Kloimüllner, C., 2015. PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. J. Glob. Optim. 63 (3), 597–629.
Raviv, T., Kolka, O., 2013. Optimal inventory management of a bike-sharing station. IIE Trans. 45 (10), 1077–1093.
Raviv, T., Tzur, M., Forma, I.A., 2013. Static repositioning in a bike-sharing system: models and solution approaches. Eur. J. Transp. Logist. 2 (3), 187–229.
Regue, R., Recker, W., 2014. Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem. Transp. Res. Part E: Logist. Transp. Rev. 72, 192–209.
Shi, Z., Xu, M., Song, Y., Zhu, Z., 2024. Multi-Platform dynamic game and operation of hybrid Bike-Sharing systems based on reinforcement learning. Transp. Res. Part E: Logist. Transp. Rev. 181, 103374.

Shui, C.S., Szeto, W.Y., 2018. Dynamic green bike repositioning problem – a hybrid rolling horizon artificial bee colony algorithm approach. Transp. Res. Part D: Transp. Environ. 60, 119–136.

Singla, A., Santoni, M., Bartók, G., Mukerji, P., Meenen, M., Krause, A., 2015. Incentivizing users for balancing bike sharing systems. In: In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 723–729.

Szeto, W.Y., Liu, Y., Ho, S.C., 2016. Chemical reaction optimization for solving a static bike repositioning problem. Transp. Res. Part D: Transp. Environ. 47, 104–135.

Usama, M., Shen, Y., Zahoor, O., 2019. A free-floating bike repositioning problem with faulty bikes. Procedia Comput. Sci. 151, 155–162.

Usama, M., Shen, Y., Zahoor, O., Bao, Q., 2020. Dockless bike-sharing system: Solving the problem of faulty bikes with simultaneous rebalancing operation. J. Transp. Land Use 13 (1), 491–515.

Vidal, T., 2022. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. Comput. Oper. Res. 140, 105643.

Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. 60 (3), 611–624.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput. Oper. Res. 40 (1), 475–489.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2014. A unified solution framework for multi-attribute vehicle routing problems. Eur. J. Oper. Res. 234 (3), 658–673.

Wang, J., Wang, Y., 2021. A two-stage incentive mechanism for rebalancing free-floating bike sharing systems: Considering user preference. Transp. Res. Part F: Traffic Psychol. Behav. 82, 54–69.

Wang, Y., Szeto, W.Y., 2018. Static green repositioning in bike sharing systems with broken bikes. Transp. Res. Part D: Transp. Environ. 65, 438–457.

Wang, Y., Szeto, W.Y., 2021. An enhanced artificial bee colony algorithm for the green bike repositioning problem with broken bikes. Transp. Res. Part C: Emerg. Technol. 125, 102895.

Wu, C., 2021. Determination of tort liability in traffic accidents based on performance parameter standard of electric bicycle. In: 2021 2nd international conference on computers, information processing and advanced education. New York, USA: Association for Computing Machinery (CIPAE 2021), pp. 1485–1489.

Xu, G., Li, Y., Xiang, T., Zhao, D., 2019. A bike repositioning problem with broken bikes. Syst. Eng. 37 (02), 91–99.

You, P.-S., 2019. A two-phase heuristic approach to the bike repositioning problem. Appl. Math. Model 73, 651–667.

Zhang, D., Xu, W., Ji, B., Li, S., Liu, Y., 2020. An adaptive tabu search algorithm embedded with iterated local search and route elimination for the bike repositioning and recycling problem. Comput. Oper. Res. 123, 105035.

Zhang, D., Yu, C., Desai, J., Lau, H.Y.K., Srivathsan, S., 2017. A time-space network flow approach to dynamic repositioning in bicycle sharing systems. Transp. Res. Part B: Methodol. 103, 188–207.

Zhang, S., Xiang, G., Huang, Z., 2018. Bike-sharing static rebalancing by considering the collection of bicycles in need of repair. J. Adv. Transp. https://doi.org/10.1155/2018/8086378.