Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 3040567, 15 pages
https://doi.org/10.1155/2020/3040567

Research Article

An Improved Iterated Local Search Algorithm for the Static Partial Repositioning Problem in Bike-Sharing System

Qiong Tang,[1,2,3] **Zhuo Fu**,[1,3] **Dezhi Zhang**,[1,3] **Meng Qiu**,[4] **and Minyi Li**[5]

[1]*School of Traffic and Transportation Engineering, Central South University, Changsha 410075, China*
[2]*College of Economics and Management, Hengyang Normal University, Hengyang 421002, China*
[3]*Smart Transport Key Laboratory of Hunan Province, Changsha 410075, China*
[4]*Institute for Data and Decision Analytics, The Chinese University of Hong Kong, Shenzhen 518172, China*
[5]*School of Science, RMIT University, Melbourne VIC 3000, Australia*

Correspondence should be addressed to Zhuo Fu; zhfu@csu.edu.cn

Received 4 February 2020; Revised 7 August 2020; Accepted 27 October 2020; Published 17 November 2020

Academic Editor: Dilum Dissanayake

In this paper, a single-vehicle static partial repositioning problem (SPRP) is investigated, which distinguishes the user dissatisfaction generated by different stations. The overall objective of the SPRP is to minimize the weighted sum of the total operational time and the total absolute deviation from the target number of bikes at all stations. An iterated local search is developed to solve this problem. A novel loading and unloading quantity adjustment operator is proposed to further improve the quality of the solution. Experiments are conducted on a set of instances from 30 to 300 stations to demonstrate the effectiveness of the proposed customized solution algorithm as well as the adjustment operator. Using a small example, this paper also reveals that the unit penalty cost has an effect on the repositioning strategies.

1. Introduction

As urban traffic congestion and environmental pollution problems become more and more serious, bike-sharing systems (BSSs), as a means of sustainable transportation, can effectively solve the "last mile" problem of urban public transportation and provide benefits for users in terms of finances, health, and a low-carbon lifestyle [1]. However, when the BSS is in operation, the demand and supply of bikes in each station often become unbalanced: at some stations, users might not be able to find available bikes, and in some others, there are not enough lockers for users to return their bikes. As a result, user satisfaction decreases in such situations. As pointed out by Laporte et al. [2], several related problems arising at the strategic, tactical, and operational levels can be studied to improve the BSS. Among those, one of the well-known, yet challenging program is the bike rebalancing problem or bike repositioning problem (BRP). Solutions to BRP aim to determine the optimal vehicle routes and the loading and unloading quantities at stations in order to minimize the sum of travel cost and user dissatisfaction subject to operational constraints. As a common practice, the BSS operators redistribute bikes among unbalanced stations accordingly using a dedicated fleet of trucks.

Existing research on BRP are mainly divided into two categories: dynamic and static. Dynamic BRP considers daytime operations where the user demand is changing over time, while static BRP focuses on nighttime operations in which the temporal user demand change can be ignored. Despite that many of the recent works have focused on dynamic settings, there are certainly practical advantages and importance of investigating the static repositioning problem. For instance, the static repositioning allows the repositioning truck to travel quickly in the city without aggravating the traffic congestion and parking problems. Hence, most existing works that take into account the next day bike inventory arrangement in BSS have focused on static settings [3]. In terms of the number of trucks used, static BRP studies are mainly divided into two categories:

single and multiple vehicle problems. However, compared with multiple-vehicle BRP [3–21], it is more realistic to consider a single-vehicle BRP for a particular street in the city [22–32]. In this paper, we focuses on the single-vehicle static BRP.

Often in the operation process of static BRP, considering various constraints of resource including operation time, capacity of the relocation trucks, and the inherent imbalance between supply and demand of the BSS (i.e., the number of surplus bikes is not equal to the number of deficit bikes at all stations), there is no way to achieve a complete or perfect balance among all stations [3]. Therefore, in practice, only a partial balance among some stations can be achieved. This problem is defined as the static partial repositioning problem (SPRP) [13]. In the SPRP, it is necessary to determine which station needed to be rebalanced, and the loading and unloading quantities at those stations, as well as the route of trucks to minimize the sum of travel cost and user dissatisfaction.

Existing models to SPRP define user dissatisfaction either as the sum of the unmet demand (only consider stations with too few bikes) [4–9] or as the sum of the absolute differences between the current inventory and the target inventory at all stations (including stations with too many bikes and stations with too few bikes) [10, 11, 32]. They commonly ignore the different level of dissatisfaction caused by different stations. However, existing study demonstrates that user dissatisfaction caused by not satisfying one unit of demand may vary from station to station [12]. For instance, at locations with high density of bike stations or convenient transportation, the level of user dissatisfaction may be low, because users can easily walk to other nearby stations or use other modes of transportation. However, it can be significantly higher in stations where bike sharing is the only option available to users. Therefore, it is important to distinguish the level of user dissatisfaction generated by not satisfying one unit of demand at different stations. It is not very scientific to quantify the total dissatisfaction of all stations by simply using the sum of the number of bikes that are not satisfied at all stations.

In this paper, we build an SPRP on the model proposed by Di Gaspero et al. [9] and Szeto et al. [32] and develop customized solution to the SPRP. Different from existing work, our proposed model will distinguish the impact of different stations on user dissatisfaction by defining different unit penalty costs associated with unsatisfied one bike at different stations. To the best of our knowledge, we are among the first to develop customized solutions to SPRP that could take into account the differentiation of user dissatisfaction at different stations.

The main contribution of this paper on the SPRP literature can be summarized as follows:

(1) A comprehensive review of the static BRP literature is provided, including the complete and partial repositioning aspects

(2) A new SPRP is introduced, where the level of user dissatisfaction could be different at different stations

(3) A heuristic algorithm based on iterated local search is proposed to solve the problem

(4) A novel loading and unloading quantity adjustment operator is designed to further improve the quality of the solution

The remainder of this paper is organized as follows: Section 2 reviews the BRP literature. Section 3 describes the mathematical formulation of the problem as well as the nonlinear programming model. Then, in Section 4, we present the improved iterated local search algorithm to solve the model. We present experimental results in Section 5 and conclude this paper with discussion on potential future research in Section 6.

2. Literature Review

The bike-sharing system has become an increasingly studied problem in the last two decades. Si et al. [33] provided a complete and insightful picture of all 208 relevant articles about bike-sharing research published from 2010 to 2018. As mentioned before, this paper focuses on the static bike repositioning problem (BRP), and hence, this section will mainly analyze the current literature on the static BRP.

The research on the static BRP can be traced back to the seminal paper of Benchimol et al. [22], in which the authors introduced the static single-vehicle BRP and proved it to be NP-hard. For the first time, Pal and Zhang [13] categorized the static BRP into complete and partial repositioning problem based on the rigorous of a repositioning that needs to be performed. Table 1 presents a summary of the literature on the static BRP in terms of the type of repositioning (complete or partial), the number of trucks used ($|V|$), and the objective function.

2.1. Complete Repositioning Problem. In the complete repositioning problem, achieving a perfect balance among stations is considered as a hard constraint. In this case, the ideal inventory of each station must be met after the repositioning operation. The objective of this problem is mostly to study how to arrange the order of stations' visits so that the travel cost is minimized, see, e.g., [15–17, 22–27]. The exceptions are Pal and Zhang [13] and Wang and Szeto [14] where the objective is to minimize the makespan of the rebalancing fleet and the total CO_2 emissions, respectively.

Various solutions have been proposed to solve the static single-vehicle complete repositioning problem, for instances, Benchimol et al. [22] developed approximation algorithms, Chemla et al. [24] presented a branch-and-cut method embedded Tabu search, and Erdoğan et al. [26] proposed the first exact algorithm that consists of a branch-and-cut algorithm based on Benders combinatorial cuts. Later on, Cruz et al. [25] addressed the same problem by proposing an iterated local search algorithm. Compared with the algorithms proposed by Chemla et al. [24] and Erdoğan et al. [26], the iterated local search developed by Cruz et al. [25] has more advantages in term of solution time

Table 1: Summary of the static BRP in the literature.

References	Type of repositioning	$\|V\|$	Objective function
Benchimol et al. [22]	Complete repositioning	1	Minimize total travel cost
Bruck et al. [23]			
Chemla et al. [24]			
Cruz et al. [25]			
Erdoğan et al. [26]			
Lahoorpoor et al. [27]			
Bulhões et al. [15]			
Dell'Amico et al. [16, 17]		≥1	
Pal and Zhang [13]		≥1	Minimize the makespan of the rebalancing fleet
Wang and Szeto [14]			
Raviv et al. [3]		≥1	Minimize the total CO_2 emissions
Forma et al. [19]			
Ho and Szeto [12]		≥1	Minimize the weighted sum of the penalty cost and total travel time
Ho and Szeto [28]	Partial repositioning	1	Minimize total penalty cost
Tang et al. [29]			
Di Gaspero et al. [5, 8]		1	The upper-level model minimizes the total penalty cost; the lower-level model minimizes the travel time
Raidl et al. [6]			
Rainer-Harbach et al. [4, 7]		≥1	Minimize the weighted sum of the total absolute deviation from the target number of bikes, the total number of loading/unloading quantities, and the total travel time
Di Gaspero et al. [9]		≥1	Minimize the weighted sum of the total absolute deviation from the target number of bikes, and the total travel time
Alvarez-Valdes et al. [18]		≥1	Minimizing the overall cost of unsatisfied demands
Erdoğan et al. [30]		1	Minimize total travel cost and handling cost
Li et al. [31]		1	Minimize the sum of travel, imbalance, substitution, and occupancy costs
Liu et al. [10]		≥1	Minimize the weighted sum of the total unmet demand, the inconvenience of getting a bike and total operational time
Szeto et al. [32]		1	Minimize the weighted sum of the total unmet demand and the total operational time
Schuijbroek et al. [20]		≥1	Minimize maximum the sum of total travel, loading and unloading costs
Szeto and Shui [20]		≥1	Minimize the positive deviation from the tolerance of total demand dissatisfaction first and then service time in form of the total service time or maximum service time
You [21]		≥1	Minimize the sum of the total unmet demand and travel costs
This paper		1	Minimize the weighted sum of the total deviation from the target number of bikes and the total operational time

and quality. Bruck et al. [23] introduced static single-vehicle BRP with forbidden temporary operations and proposed three exact algorithms based on different mathematical formulations.

There are also studies on the static multivehicle complete repositioning problems, for instance, Bulhões et al. [15] presented a branch-and-cut algorithm and an iterated local search. Dell'Amico et al. [16] presented branch-and-cut algorithm. Dell'Amico et al. [17] developed a destroy and repair metaheuristic algorithm, which makes use of effective constructive heuristic and of several local search procedures.

It is worth mentioning that there are three BRP articles occurring in the BSS a bit different from most of those complete repositioning problems studied in this survey. Pal and Zhang [13] modeled a static BRP in a free-floating bike-sharing system and presented a hybrid nested large neighborhood search with a variable neighborhood descent algorithm. Wang and Szeto [14] introduced a static green BRP with broken bikes, and applied CPLEX to determine the routing and loading decisions. Lahoorpoor et al. [27] proposed a bottom-up spatial cluster-based method to solve the static BRP with consideration of the users' behavior in the BSS.

2.2. Partial Repositioning Problem. In the partial repositioning problem, achieving a perfect balance among stations is considered as a soft constraint; that is to say, it is not necessary to achieve the ideal inventory at each station after repositioning operation. As can be seen from the Table 1, with the exceptions of Ho and Szeto [28], and Alvarez-Valdes et al. [18], the partial repositioning problems in the literature are modeled as multiobjective optimization problems. The goal of these problems is not only to pursue the minimum transportation cost by determining routing decisions, but also to minimize user dissatisfaction by determining loading and unloading decisions at each station. User dissatisfaction is commonly measured by the penalty costs [3, 12, 19, 28, 29], the absolute deviation from the target number of bikes [4–9], and the unmet demand [10, 11, 32].

Raviv et al. [3] creatively defined the user dissatisfaction as penalty function which represents the expected number of shortages at any inventory level at each station. The authors addressed a multivehicle SPRP, and proposed a two-phase heuristic to first solve the routing subproblem then to solve the loading and unloading subproblem. Then, Forma et al. [19] presented a 3-step math heuristic, and later on, Ho and

Szeto [12] developed a hybrid large neighborhood search algorithm (proven to be more effective than that proposed in [19]) to solve the same problem as in [3]. Ho and Szeto [28] presented an iterated Tabu search to solve a single-vehicle SPRP. Tang et al. [29] proposed a bilevel programming model by taking the outsourcing transportation mode into account and presented an iterated local search algorithm and Tabu search algorithm to solve the optimization problem.

In [4], Rainer-Harbach et al. measured the user dissatisfaction based on the deviation from the target number of bikes, and addressed the multivehicle SPRP. They proposed a heuristic approach for the static BRP where the routing decisions are calculated by a variable neighborhood search metaheuristic and the loading decisions are computed by a helper algorithm. There are a list of other solutions proposed for this problem setting, for instance, in [5], Di Gaspero et al. introduced a combination of constraint programming and ant colony optimization approach; in [6], Raidl et al. provided an efficient method for determining optimal loading operations based on two maximum flow computations; in [7], Rainer-Harbach et al. proposed PILOT, a greedy randomized adaptive search procedure, along with a variable neighborhood search algorithm to solve this problem; and in [8], Di Gaspero et al. employed constraint programming to tackle this problem. Last but not least, in [9], Di Gaspero et al. described a multivehicle SPRP with the objective of minimizing both the weighted sum of the total absolute deviation from the target number of bikes, as well as the total travel time. They proposed a constraint programming approach that utilizes a smart branching strategy and large neighborhood search algorithm to solve the problem.

For the first time, Szeto et al. [32] measured the user dissatisfaction based on the unmet demand, and addressed a single-vehicle SPRP by proposing an enhanced chemical reaction optimization algorithm. Liu et al. [10] studied a multivehicle SPRP for the free-floating bike-sharing system. They took into account the different convenience level of getting bikes, and proposed an enhanced chemical reaction optimization algorithm to solve the problem. You [11] studied a multivehicle SPRP under a minimum service requirement over a planning horizon, and proposed a two-phase heuristic algorithm to solve the problem based on linear programming.

In this section, there are five articles differentiate themselves from the above other studies. Alvarez-Valdes et al. [18] addressed a static BRP to optimize the level of service quality by a two-stage algorithm. Erdoğan et al. [30] addressed a static BRP where the target inventory at each station must lie in a predetermined interval. They developed and implemented a Benders decomposition scheme and a branch-but-cut algorithm to solve the problem. Li et al. [31] investigated a static BRP with multiple bike types, and proposed a combined hybrid genetic algorithm. Schuijbroek et al. [20] unified dual-bounded service level constraints (with added inventory flexibility) and vehicle routing for static BRP and proposed a cluster-first route-second heuristic. Szeto and Shui [21] developed an enhanced artificial bee colony algorithm that incorporated a new set of optimal loading and unloading strategies to solve the static BRP to minimize first the positive deviation from the tolerance of the total demand dissatisfaction and then the total service time of all trucks.

As demonstrated above, in complete repositioning problem, the loading and unloading quantities to a station are given in advance, as opposed to our partial repositioning problem where they are decision variable. The increase of decision variables also increased the computational complexity of solving partial positioning problems which is more in line with the operation of the actual BSS. When it comes to partial positioning problems, most researchers have focused their objective on minimizing the user dissatisfaction measured by the sum of unmet demand or deviations from the target fill levels at all stations [4–11, 32]. However, as mentioned previously, the different level of user dissatisfaction caused by unmet demands or deviations at different stations is another important aspect to take into account although it has been ignored in most of the literature. In this paper, we do not only pursue the overall minimization of operational costs and user dissatisfaction costs but also consider the differences in user dissatisfaction at different stations. As summary above, two works have solved the complete repositioning problem by iterated local search [23, 25], only one work has used iterated local search to solve the bilevel partial repositioning problem [29]. It is obvious that the iterated local search algorithm in the above three papers cannot effectively solve the partial repositioning problem studied in this paper. Considering the characteristics and solution quality of the problem, we improve the traditional iterated local search algorithm and propose a customized algorithm for solving the problem in this paper. Computational results show that the improved algorithm is effective.

3. Mathematical Formulation

The bike-sharing system studied in this paper comprises a depot, a set of stations, and a truck. Each station has a repositioning demand. If the repositioning demand at a station is greater than 0, the station is defined as a pickup station and can provide bikes to the delivery stations. The set of pickup stations is denoted by P. On the other hand, if the repositioning demand at a station is smaller than 0, then the station is defined as a delivery station and should be supplied with bikes from pickup stations. The set of delivery stations is denoted by D. If the demand at a station is equal to 0, then the station is called a balanced station. Balanced stations can be ignored in the partial repositioning problem as there is no visiting or loading/unloading operations needed.

We consider the scenario that only a single truck with a given capacity is available. The truck collects bikes from pickup stations and transports them to delivery stations. It starts from and returns to the depot empty and operates within a given repositioning time constraint. Moreover, each station is allowed to be visited at most once, and not all stations need to be visited. The objective is to determine the route of the truck and the loading and unloading quantities at each visited station to minimize the weighted sum of the

total operational time as well as the total absolute deviation from the target number of bikes at all stations.

3.1. Notations

3.1.1. Sets

N: set of stations

N_0: set of nodes, including the stations (indexed by i, $i \in N$) and the depot (indexed by 0)

P: set of pickup stations, where $P \subset N$

D: set of delivery stations, where $D \subset N$

R: set of stations visited by the truck, where $R \subset N$

U: set of stations not visited by the truck, where $U \subset N$

3.1.2. Parameters

d_i: demand of station i at the beginning of the repositioning operation, if $d_i > 0$, then $i \in P$; if $d_i < 0$, then $i \in D$

t_{ij}: operational time that includes the travel time from station i to station j and the expected time required to load or unload bikes at station j

Q: truck capacity

T: repositioning time

M: a very large number

w_i: the unit penalty cost associated with unsatisfied one bike at station i, $w_i \in (0, 1], \forall i \in N$

α: the weight associated with the truck's total operational time

3.1.3. Decision Variables

x_{ij}: binary variable that equal 1 if truck travels directly from station i to station j, and 0 otherwise

y_{ij}: the number of bikes onto the truck when it travels directly from station i to station j

y_i^P: the number of bikes loaded on the truck at station $i, y_i^P \geq 0, \forall i \in P$, if $i \notin P$, then $y_i^P = 0$

y_i^D: the number of bikes unloaded from truck at station $i, y_i^D \geq 0, \forall i \in D$, if $i \notin D$, then $y_i^D = 0$

3.2. Mathematical Formulation. The formulation for the SPRP addressed in this study is given as follows:

$$\min \alpha \sum_{i \in N_0} \sum_{j \in N_0, i \neq j} t_{ij} x_{ij} + \sum_{i \in P} w_i \left(d_i - y_i^P\right) + \sum_{i \in D} w_i \left(-d_i - y_i^D\right), \tag{1}$$

which is subject to

$$\sum_{j \in N} x_{0j} = \sum_{j \in N} x_{j0} = 1, \tag{2}$$

$$\sum_{j \in N} y_{0j} = \sum_{j \in N} y_{j0} = 0, \tag{3}$$

$$\sum_{j \in N_0, j \neq i} x_{ij} \leq 1, \quad \forall i \in N, \tag{4}$$

$$\sum_{j \in N_0, j \neq i} x_{ij} = \sum_{j \in N_0, j \neq i} x_{ji}, \quad \forall i \in N, \tag{5}$$

$$\sum_{i, j \in N_0, i \neq j} t_{ij} x_{ij} \leq T, \tag{6}$$

$$q_j \geq q_i + 1 - M\left(1 - x_{ij}\right), \quad \forall i, j \in N_0, i \neq j, \tag{7}$$

$$\sum_{j \in N_o, j \neq i} y_{ij} - \sum_{j \in N_o, j \neq i} y_{ji} = y_i^P - y_i^D, \quad \forall i \in N, \tag{8}$$

$$y_{ij} \leq Q x_{ij}, \quad \forall i, j \in N_0, i \neq j, \tag{9}$$

$$\sum_{i \in N} y_i^P - \sum_{i \in N} y_i^D = 0, \tag{10}$$

$$y_i^D \leq \min\left(\max\left(-d_i, 0\right) \cdot \sum_{j \in N_0, j \neq i} x_{ij}, \sum_{j \in N_0, j \neq i} y_{ji}\right), \quad \forall i \in N, \tag{11}$$

$$y_i^P \le \min\left(\max(d_i, 0) \cdot \sum_{j \in N_0, j \ne i} x_{ij}, \left(Q - \sum_{j \in N_0, j \ne i} y_{ji}\right)\right), \quad \forall i \in N, \tag{12}$$

$$q_i \ge 0, \quad \forall i \in N_0, \tag{13}$$

$$y_i^P \ge 0, \quad \forall i \in N, \tag{14}$$

$$y_i^D \ge 0, \quad \forall i \in N, \tag{15}$$

$$y_{ij} \ge 0, \quad \forall i, j \in N_0, i \ne j, \tag{16}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in N_0, i \ne j. \tag{17}$$

Objective function (1) is defined as the weighted sum of the total operational time and the total weighted sum of deviation from the target number of bikes. Constraint (2) states that the truck departs from the depot only once and returns to the depot. Constraint (3) ensures that the truck starts from and returns to the depot empty. Constraints (4) ensure that the truck can visit a station at most once. Constraints (5) ensure that if the truck visits a station, it must leave that station. Constraint (6) ensures that total operational time of the truck does not exceed the repositioning time available. Constraints (7) are the subtour elimination constraint [34]. Constraints (8) require that the number of bikes loaded onto or unloaded from the truck at a given station equals the difference between the truck load before and after the station visit. Constraints (9) ensure that the load on the vehicle cannot be greater than the truck capacity. Constraint (10) ensures that bikes picked up from pickup stations by the truck are eventually delivered to delivery stations. Constraints (11) require that the unloading quantities at a station should not be greater than the number of bikes needed by that station and greater than the number of bikes on the truck when the vehicle reaches station. Constraints (12) require that the loading quantities at a station should not be greater than the number of bikes supplied by that station and greater than the available spaces on the truck when it arrives at the station. Constraints (13) ensure the auxiliary variable is nonnegative. Constraints (14)–(16) ensure that loading and unloading quantities at each station and the number of bikes on the truck are nonnegative integers. Constraints (17) define x_{ij} to be a binary variable.

4. Solution Method

Similar to other existing works (e.g., [15, 25, 29]), we use iterated local search (ILS) to solve the BRP problem. The ILS algorithm is a metaheuristic algorithm. It obtains the optimal solution through two mechanisms of local search (see Section 4.3) and perturbation search (see Section 4.4). In this paper, local search is executed through the randomized variable neighborhood decent (RVND) procedure. Algorithm 1 illustrates the ILSSPRP algorithm for the SPRP. According to Algorithm 1, after constructing the initial solution s_0 by applying the initialization procedure (see Section 4.2), the procedure RVND (see Section 4.3) and perturbation search (see Section 4.4) are called repeatedly until the predefined termination criterion (the maximum consecutive iterations without improvement limit in our case) is satisfied. The local search including 6 neighborhood structures, such as Swap (N1), Insertion (N2), Delete (N3), Exchange (N4), 2-Opt (N5), and Reinsertion (N6) neighborhood structures, is executed using an RVND procedure, whereas the perturbation procedure performs Swap (P1), Insertion (P2), Exchange (P3), and Reinsertion (P4) neighborhood structures.

According to the characteristics of the SPRP, we improve the traditional ILS algorithm by adding a novel adjustment operator after RVND and perturbation search. Details of the adjustment operator can be found in Section 4.5, and we also verify the impact of the proposed adjustment operator on the ILSSPRP in Section 5.2. Let s represent the current solution for the SPRP and s* represent the best solution that can be searched currently. For a solution s, TC (s) gives the objective value of solution s. We also denote *ConsIter* as the current consecutive iterations without improvement, and *MaxConsIter* as the maximum consecutive iterations without improvement, respectively.

4.1. Solution Representation. A solution w is represented by $w = (x, y, q)$ as shown in Figure 1, where x is a vector that stores the routing sequence of the truck $(x_0, x_1, ..., x_n, x_{n+1})$, where $x_0 = x_{n+1} = 0$ represents the depot, x_h $(x_h \in N, h \in \{1, \ldots, n\})$ represents the station visited by the truck, the subscript h indicates the order of stations being visited by the truck. y is a vector that stores the demand that has been met at the station visited by the truck, where positive and negative values indicate the quantities of bikes collected and delivered, respectively. q is a vector used to store the cumulative load after the truck visits the station. According to equations (3) and (9), if a solution is feasible, then $q_h = 0, \forall h \in \{0, n+1\}$ and $0 \le q_h \le Q$, $\forall h \in \{0, 1, \ldots, n, n+1\}$ where Q is the truck capacity.

s_0: = **Initialization;**
s^*: = s_0; s: = s_0; *ConsIter*: = 0;
while *ConsIter* ≤ *MaxConsIter* do
  s' := **RVND** (s, N^k); {$1 \le k \le 6$}
  s := **Adjustment** (s');
  if TC (s) < TC (s^*), then
    s^*: = s;
    ConsIter: = 0;
  else
    ConsIter: = *ConsIter* + 1;
  end
  s'': = **Perturbation search** (s^*, P^m); {$1 \le m \le 4$}
  s: = Adjustment (s'');
End
Return s^*

Algorithm 1: ILS_{SPRP} for SPRP.

h	0	1	2	3	4	5	6	7	8	9
x_h	0	8	9	6	2	1	3	5	4	0
y_h	0	5	4	−6	−2	5	−4	6	−8	0
q_h	0	5	9	3	1	6	2	8	0	0

Figure 1: Solution representation.

4.2. Initialization. In this paper, the initial feasible solution is efficiently generated by a construction heuristic algorithm (see [12, 28, 29]). The basic construction heuristic algorithm was first proposed by Ho and Szeto [28] to solve the single-vehicle BRP, and Ho and Szeto [12] developed it to solve the multiple-vehicle BRP, and then Tang et al. [29] improved it to solve the bilevel BRP.

The construction heuristic algorithm can be divided into two main steps:

Step one, Sort: we first divide stations into two categories: pickup stations and delivery stations. Then, stations are sorted in descending order by the ranking criteria. Similar to Tang et al. [29], in this paper, the ranking criteria are $|w_i d_i^Q|$, where $d_i^Q = \{\min(Q, d_i), i \in P;\ \max(-Q, d_i), i \in D\}$.

Step two, Insert: we first insert the pickup stations in turn into the route, if no pickup station can be inserted into the current route without exceeding the capacity limits; then, delivery stations are inserted in turn in the route. We repeat this process until no station can be inserted without violating the predetermined repositioning time constraint.

4.3. Local Search. After generating the initial solution, we improve this initial solution with local search technique. We consider two kinds of neighborhood structures: intersets and intraroute. The former performs moves between set R (the set of stations visited by the truck) and set U (the set of stations not visited by the truck), whereas the latter only considers moves within the route.

4.3.1. Interset Neighborhood Structures

(i) Swap-N^1: randomly select a station i from the route (set R), then select another station j ($|d_i| \le |d_j|$) of the same type from set U, and exchange them.

(ii) Insertion-N^2: randomly select a station from the set U to insert into the route (set R).

(iii) Delete-N^3: randomly select a station from the route (set R) and delete it.

4.3.2. Intraroute Neighborhood Structures

(i) Exchange-N^4: randomly select two stations from the route and exchange them.

(ii) 2-Opt-N^5: randomly select two stations from the route and exchange the access order of the stations between the two stations.

(iii) Reinsertion-N^6: randomly select a station and reinsert it into the route.

(iv) The new neighborhood solutions obtained by applying above moves may no longer remain feasible. In order to ensure that a sufficient number of feasible solutions can be obtained, we selectively repair some infeasible solutions. Here, we repair infeasible solutions obtained by applying N^2 and N^3 (details to be discussed in Sections 4.6.1 and 4.6.2). In terms of the other moves, only feasible solutions can be accepted as the new neighborhood solutions.

4.4. Perturbation Search. In order to avoid falling into local optimum and to realize global search, perturbation is used to constantly change the current local optimum solution. The following four kinds of moves are used to implement perturbations:

(i) Swap-P^1: randomly select a station i from the route, then select another station j of the same type from set U, and exchange them.

(ii) Insertion-P^2: randomly select a station from the set U and insert it into the route. For the infeasible neighborhood solution, when using the repair operator to repair, Insertion-P^2 randomly selects a feasible solution after repair as the new neighborhood solution, while Insertion-N^2 chooses the best feasible solution after repair as the new neighborhood solution.

(iii) Exchange-P^3: randomly select two adjacent stations from the route and swap them with two other adjacent stations from the route.

(iv) Reinsertion-P^4: randomly select two adjacent stations from the route, delete them, and reinsert them elsewhere on the route.

(v) Here, we repair infeasible solutions obtained by applying P^2 (details to be discussed in Section 4.6.3). For other moves P^1, P^3, and P^4, only feasible solutions are accepted as the new neighborhood solutions.

4.5. Adjustment Operator. The basic idea of the adjustment operator is that a solution can be improved by properly adjusting the loading and unloading quantities between stations in a heuristically way [10, 12, 28, 32].

Similar to the pickup-delivery cycle proposed by Lei and Ouyang [35], we first define a pickup-delivery pair as a sequence of consecutive visits that starts from a pickup station (preceded by the depot or delivery station) and ends at the corresponding delivery station (followed by the depot or pickup station). Figure 2 illustrates an example in which the truck visits two pickup-delivery pairs (with $Q = 10$). The first pickup-delivery pair contains one pickup station (station 1) and one delivery station (station 2), and the second pickup-delivery pair also contains one pickup station (station 4) and one delivery station (station 5). The truck starts from the depot with empty, picks up 5 bikes at station 1, and delivers directly to station 2, where 3 bikes are unloaded. Then, 6 bikes are further collected at station 4, delivers 8 bikes at station 5, and finally ends at the depot empty.

In this paper, we divide the adjustment operator into two categories: adjustment of the quantities of different types of stations in one pickup-delivery pair and adjustment of the quantities of the same type of stations in different pickup-delivery pairs. We first adjust the quantities of stations within each pickup-delivery pair, and then adjust the quantities of the same type of stations across different pickup-delivery pairs.

4.5.1. Adjustment of the Quantities of Different Types of Stations in One Pickup-Delivery Pair. In this case, on the current route, we adjust the quantities of different type of stations within each pickup-delivery pair, until all pickup-delivery pairs are adjusted. For one pickup-delivery pair, we consider one pickup station x_i and one delivery station x_j at a time (i and j indicate the order of stations being visited by the truck, respectively), with their loading and unloading quantities increased r at the same time, which is formulated by $r = \min\left\{d_{x_i} - y^P_{x_i}, |d_{x_j}| - y^D_{x_j}, Q - q_i\right\}$, subject to capacity and loading constraints. Then, the loading and unloading quantities on station x_i and station x_j are updated at the same time by $y^P_{x_i} = y^P_{x_i} + r$; $y^D_{x_j} = y^D_{x_j} + r$.

As shown in Figure 3(a), in the first pickup-delivery pair, pickup station 1 and delivery station 2 are selected (with $Q = 10$) to adjust the loading and unloading quantities. In this case, the increased loading/unloading quantities $r = \min\{d_1 - y^P_1, |d_2| - y^D_2,\ Q - q_1\} = \min\{7 - 5, 6 - 3, 10 - 5\} = 2$. Then, $y^P_1 = y^P_1 + 2 = 5 + 2 = 7$, $y^D_2 = y^D_2 + 2 = 3 + 2 = 5$. In the second pickup-delivery pair, pickup station 4 and delivery station 5 are selected as shown in Figure 3(b), $r = 1$; then, $y^P_4 = y^P_4 + 1 = 6 + 1 = 7$, $y^D_5 = y^D_5 + 1 = 8 + 1 = 9$.

4.5.2. Adjustment of the Quantities of the Same Type of Stations in Different Pickup-Delivery Pairs. For two stations of the same type, adjusting the loading (for pickup stations) and unloading (for delivery stations) quantities between them can improve the current solution, under the condition of ensuring the feasibility of the solution after the transformation. Each time, two stations of the same type in different pickup-delivery pairs are selected, and the loading and unloading quantities of one station are increased by 1 unit, while that of the other station is reduced by 1 unit, resulting in a decrease in the value of the objective function (i.e., the weighted sum of the operational time and deviation from target). The procedure is repeated until there is no loading and unloading quantity adjustment between stations to improve the current solution.

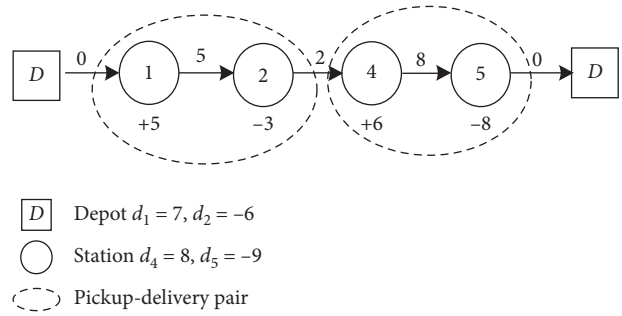

Figure 2: Illustration of pickup-delivery pair.

Consider a current feasible solution shown in Figure 2. Figures 4(a) and 4(b) show an example where pickup station 1 (in the first pickup-delivery pair) and pickup station 4 (in the second pickup-delivery pair) are selected; then, we shift two unit quantity between them. For instance, in Figure 4(a), we increase the loading quantities at station 1 by two, and reduce the loading quantities at station 4 by two. We do the opposite, and it is shown in Figure 4(b). Under the two transformations, the new solution that could reduce the objective value will be accepted.

Similarly, as shown in Figures 4(c) and 4(d), delivery station 2 in the first pickup-delivery pair and delivery station 5 in the second pickup-delivery pair are selected; then, we shift two unit quantities between them. Figure 4(c) shows the operation where we increase the unloading quantities at station 2 by two, and decrease the unloading quantities at station 5 by two. And Figure 4(d) shows the opposite operation. Similarly, between the two newly generated solutions, we accept the one that could reduce the objective value as the current solution.

4.6. Repair Operator. There is a couple of constraints for a solution to be feasible. First of all, the sum of the total satisfied demand of stations visited by the truck on the route is 0 (according to equation (10)). For instance, a feasible solution is shown in Figure 5, suppose $Q = 10$, and the total satisfied demand of stations visited by the truck is 0, that is, $4-3+2+2-5=0$. Therefore, if we want to ensure that the neighbor solution obtained by applying the Insertion and Delete is feasible, we must make sure that the sum of the total satisfied demand of stations visited by the truck is 0. Besides, we also need to make sure that the solution meets the capacity and loading constraints

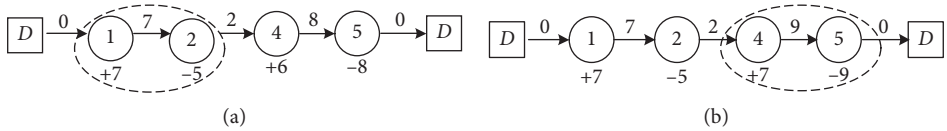

Figure 3: Neighbor solutions after applying the adjustment operator in one pickup-delivery pair. (a) Adjustment of the loading and unloading quantities between stations in the first pickup-delivery pair. (b) Adjustment of the loading and unloading quantities between stations in the second pickup-delivery pair.

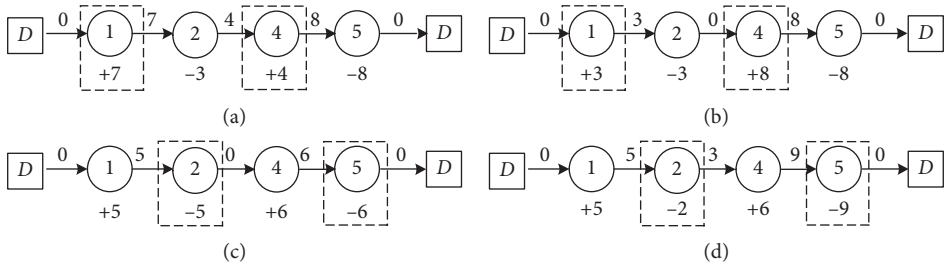

Figure 4: Neighbor solutions after applying the adjustment operator between two pickup-delivery pairs. (a) Adjustment of the loading quantities between station 1 and station 4. (b) Adjustment of the loading quantities between station 1 and station 4. (c) Adjustment of the unloading quantities between station 2 and station 5. (d) Adjustment of the unloading quantities between station 2 and station 5.

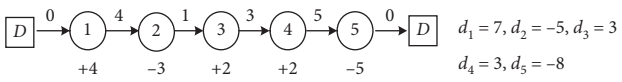

Figure 5: A feasible solution.

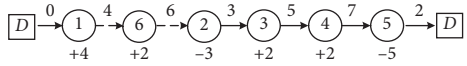

Figure 6: An infeasible solution with station 6 inserted.

(according to equations (3) and (9)). In the following sections, we describe the repair procedure for infeasible solutions obtained by local search and perturbation search, respectively.

4.6.1. Repairing Infeasible Solutions after Applying N2. Given a feasible solution, if we insert a station into the solution, the obtain new neighbor solution is certainly not feasible. However, the feasibility of the new neighbor solution can be ensured by adjusting the loading/unloading quantity of a certain station. Assuming that the current satisfied demand of the inserted station is Δ^I, then it is necessary to reduce Δ^I on the basis of the current satisfied demand of a certain station on the route, in order to balance out the sum of the total satisfied demand on the route to be 0. In this case, the new neighbor solution after applying the repair procedure is feasible if the capacity and loading constraints are satisfied. There might be more than one feasible adjustment scheme, under which the solution with the lowest objective value is selected as the new neighbor solution.

As shown in Figure 6, if we insert station 6 ($\Delta^I = y_6^P = 2$) into the current feasible solution (as shown in Figure 5), the total of all satisfied demands is 2 but not 0. Then, we need to choose one station from the route and reduce its satisfied demand by Δ^I. As shown in Figure 7, there are three feasible neighbor solutions can be obtained by applying the repair procedure, and we choose the neighbor solution with the lowest objective value as the new feasible solution.

4.6.2. Repairing Infeasible Solutions after Applying N^3. Assuming that the current satisfied demand for the deleted station is Δ^D, then it is necessary to increase Δ^D on the basis of the current satisfied demand of a certain station on the route, to ensure that the sum of the total satisfied demand on the route is 0. Similarly, if there are more than one feasible neighbor solutions, the solution with the lowest objective value is selected as the new neighbor solution.

As shown in Figure 8, if we delete station 4 ($\Delta^D = y_4^P = 2$) from the current feasible solution (Figure 5), the total of all satisfied demand is −2. Therefore, we need to choose one station from the route and increase its satisfied demand by Δ^D. As shown in Figure 9, there are three feasible neighbor solutions can be obtained from the repair procedure, and we choose the neighbor solution with the lowest objective value as the new neighbor solution.

4.6.3. Repairing Infeasible Solutions after Applying P2. In this case, the repair operation would be the same as in Section 4.6.1, the only difference is that when there is more than one feasible neighborhood solution generated, we randomly select one as the new neighbor solution.

5. Computational Results

In this section, we present experiments conducted to verify the effectiveness of the proposed ILS_{SPRP} algorithm as well as the proposed adjustment operator. We compare the performance of our proposed solution with the optimal solution of Lingo 18. We also analyze through experiments the

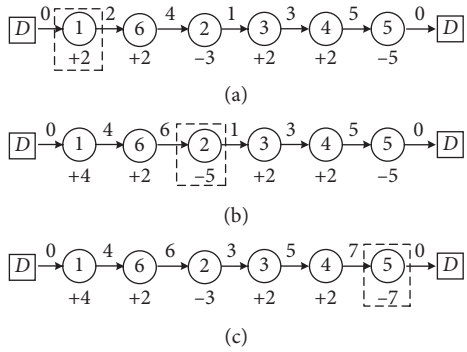

Figure 7: Feasible solutions after applying repair operator. (a) Adjustment of the loading quantities of station 1. (b) Adjustment of the unloading quantities of station 2. (c) Adjustment the unloading quantities of station 5.

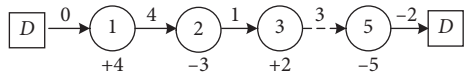

Figure 8: An infeasible solution with station 4 deleted.

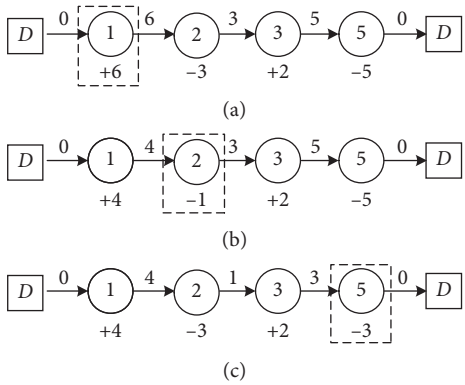

Figure 9: Feasible solutions after applying repair operator. (a) Adjustment of the loading quantities of station 1. (b) Adjustment of the unloading quantities of station 2. (c) Adjustment of the unloading quantities of station 5.

influence of the change of weight coefficient α on the proposed ILS_{SPRP} and the change of unit penalty cost w_i on the repositioning strategies.

We follow the instances proposed by Rain-Harbach et al. [7] to conduct the experiments, where the size of the instances is ranging from 30 to 300 stations. For each instance, we select the first seed of the 30 independent seeds. The initial objective value, the total pickup, and total drop-off quantities of each instances are listed in Table 2. We consider instances with 30 and 60 stations as the small-size instances; instances contain 120 and 180 stations as the medium-size instances; and instances with 240 and 300 stations as the large-size instances. In this study, unless otherwise specified, we follow the setting in Szeto et al. [32], $\alpha = 0.00001$. Three repositioning time constraints (in seconds) were used, $T = 14400$ s, $T = 21600$ s, and $T = 28800$ s. Two truck capacities (in terms of the number of bikes) were considered: $Q = 10$ and $Q = 20$, respectively. The unit penalty cost w_i associated with one unsatisfied bike at station i was randomly generated in the interval (0, 1], $\forall i \in N$.

The experiments are conducted in Matlab, on a computer with Intel Core i5-4590 CPU @ 3.30 GHz and 4 GB RAM. We run each instance for 20 times, the best and average of the solutions and the average computing time of those 20 runs are used to evaluate the performance of the ILS_{SPRP}.

The main parameter of ILS_{SPRP} to be calibrated is the maximum consecutive iterations without improvement (*MaxConsIter*). Here, we set its value as a function of the instance, $MaxConsIter = \max\{100, |N|\}$, similar to Bulhões et al. [15].

5.1. Comparison between Lingo and ILS_{SPRP}. In terms of the quality of the solution and the calculation efficiency, Tables 3–5 show the comparison results obtained by Lingo and ILS_{SPRP} under different repositioning times ($T = 14400$ s, $T = 21600$ s, and $T = 28800$ s), truck capacity ($Q = 10$ and $Q = 20$), and number of stations ($|N|$). Gap_{Best} and Gap_{Avg} (%) indicate the performance of the ILS_{SPRP} relative to that of Lingo based on the best and average objective values, respectively. The CPU (in seconds) indicates the average computing time of the ILS_{SPRP} or the computing time for Lingo, respectively.

From Tables 3–5, we can see that Lingo is not able to obtain any feasible solutions for the large-size problems (where $|N| = 240$ and $|N| = 300$). It also generally fails to obtain any global optimal solutions within 3600 seconds even for small-size problems. In contrast, for the various size of problems from small to large, the average computing time of the ILS_{SPRP} varies from 7.12 s to 33.71 s.

As shown in the Tables 3–5, almost all of the values of Gap_{Best} and Gap_{Avg} are negative (with only one exception for the results obtained under $|N| = 60$, $Q = 10$, $T = 14400$ s). The absolute values of the average Gap_{Best} and Gap_{Avg} values obtained by the ILS_{SPRP} for the scenarios with long repositioning time are larger than those with short repositioning time. Gap_{Best} and Gap_{Avg} are, respectively, −3.53 and −3.31 for $T = 14400$ s, and −5.46 and −4.45 for $T = 21600$ s, but −23.59 and −20.81 for $T = 28800$ s. This implies that the ILS_{SPRP} performs better than Lingo when solving the SPRP with a longer repositioning time.

Therefore, we can draw the conclusion that the proposed ILS_{SPRP} can obtain better feasible solutions much faster than Lingo.

5.2. The Effectiveness of Introducing Adjustment Operator into ILS_{SPRP}. In this study, the adjustment operator (presented in Section 4.5), which considers the characteristics of the SPRP, is introduced to improve the solution quality obtained by the proposed ILS_{SPRP}. In this section, we study through experiments the effectiveness of the proposed adjustment operator by

Table 2: Characterization of the instances.

$\|N\|$	Initial objective value	Total pickup quantities	Total drop-off quantities
30	99.2	107	107
60	170.3	179	179
120	330.7	354	354
180	606.4	557	557
240	760.4	785	785
300	852.2	883	883

Table 3: Result comparison of Lingo and ILSSPRP ($T = 14400$ s).

Instance		Lingo		ILS_{SPRP}			Gap_{Best} (%)	Gap_{Avg} (%)
$\|N\|$	Q	Optimal value	CPU	Best value	Average value	CPU		
30	10	48.6440	233	48.6440	48.6440	4.27	0.00	0.00
	20	43.5440	3600	41.4386	41.4386	4.58	−5.08	−5.08
60	10	117.1420	3600	117.6428	117.7424	5.33	0.43	0.51
	20	113.4440	3600	111.6422	112.9824	4.86	−1.61	−0.41
120	10	273.7440	3600	271.2440	271.9101	6.65	−0.92	−0.67
	20	296.8180	3600	256.9386	257.0192	6.74	−15.52	−15.48
180	10	554.1440	3600	538.4374	538.7209	8.02	−2.92	−2.86
	20	542.0440	3600	528.2434	528.8430	8.11	−2.61	−2.50
240	10			690.0386	690.0386	8.47		
	20			671.9338	671.9338	8.91		
300	10			796.3374	796.9097	8.92		
	20			789.1392	792.1236	10.63		
Avg			3179.13			7.12	−3.53	−3.31

Table 4: Result comparison of Lingo and ILSSPRP ($T = 21600$ s).

Instance		Lingo		ILS_{SPRP}			Gap_{Best} (%)	Gap_{Avg} (%)
$\|N\|$	Q	Optimal value	CPU	Best value	Average value	CPU		
30	10	28.8148	3600	28.5178	28.7040	8.32	−1.04	−0.39
	20	23.5148	3600	21.2130	21.7398	9.42	−10.85	−8.16
60	10	94.2160	3600	93.2130	93.7275	8.12	−1.08	−0.52
	20	84.4150	3600	81.7226	83.9474	12.92	−3.29	−0.56
120	10	249.3160	3600	243.5154	244.1531	15.46	−2.38	−2.11
	20	229.4154	3600	222.4112	224.3142	14.45	−3.15	−2.27
180	10	513.2160	3600	506.4052	506.4052	11.77	−1.34	−1.34
	20	594.4610	3600	493.0178	494.4761	22.22	−20.58	−20.22
240	10			656.0130	656.4622	17.69		
	20			629.4130	630.3757	32.69		
300	10			764.4004	764.4005	17.22		
	20			743.1136	744.1450	46.51		
Avg			3600			18.07	−5.46	−4.45

Table 5: Result comparison of Lingo and ILSSPRP ($T = 28800$ s).

Instance		Lingo		ILS_{SPRP}			Gap_{Best} (%)	Gap_{Avg} (%)
$\|N\|$	Q	Optimal value	CPU	Best value	Average value	CPU		
30	10	20.0888	3600	14.2862	15.3446	14.00	−40.62	−30.92
	20	13.5862	3600	8.6850	9.1928	18.10	−56.43	−47.79
60	10	79.5874	3600	74.7808	75.3967	16.96	−6.43	−5.56
	20	65.7868	3600	63.5886	64.1407	19.10	−3.46	−2.57
120	10	223.8880	3600	220.4964	221.8205	24.22	−1.54	−0.93
	20	243.3166	3600	193.2868	195.1978	33.08	−25.88	−24.65
180	10	593.4432	3600	475.4784	476.1721	32.82	−24.81	−24.63
	20	588.8660	3600	454.5760	454.9920	36.97	−29.54	−29.42
240	10			623.5814	625.9040	41.91		
	20			582.9796	583.6207	56.05		
300	10			719.6832	720.3621	53.98		
	20			711.2862	712.2615	57.28		
Avg			3600			33.71	−23.59	−20.81

Table 6: Result comparison of ILSSPRP with and without the adjustment operator ($T = 14400$ s).

Instance		ILS_{SPRP} without adjustment			ILS_{SPRP}			Gap_{Best} (%)	Gap_{Avg} (%)
$\|N\|$	Q	Best value	Average value	CPU	Best value	Average value	CPU		
30	10	51.6440	51.9440	2.06	48.6440	48.6440	4.27	−6.17	−6.78
	20	48.3386	48.9039	2.23	41.4386	41.4386	4.58	−16.65	−18.02
60	10	118.1410	118.1410	1.49	117.6428	117.7424	5.33	−0.42	−0.34
	20	125.8314	125.8315	2.11	111.6422	112.9824	4.86	−12.71	−11.37
120	10	277.6434	278.0650	2.20	271.2440	271.9101	6.65	−2.36	−2.26
	20	262.0434	263.8493	2.42	256.9386	257.0192	6.74	−1.99	−2.66
180	10	551.5374	552.8316	3.86	538.4374	538.7209	8.02	−2.43	−2.62
	20	536.1428	537.9650	4.44	528.2434	528.8430	8.11	−1.50	−1.72
240	10	691.0392	691.0392	3.92	690.0386	690.0386	8.47	−0.15	−0.15
	20	675.6338	675.6338	4.77	671.9338	671.9338	8.91	−0.55	−0.55
300	10	797.3254	797.3254	4.09	796.3374	796.9097	8.92	−0.12	−0.05
	20	794.0284	794.0284	4.48	789.1392	792.1236	10.63	−0.62	−0.24
Avg				3.17			7.12	−3.81	−3.90

Table 7: Result comparison of ILSSPRP with and without the adjustment operator ($T = 21600$ s).

Instance		ILS_{SPRP} without adjustment			ILS_{SPRP}			Gap_{Best} (%)	Gap_{Avg} (%)
$\|N\|$	Q	Best value	Average value	CPU	Best value	Average value	CPU		
30	10	32.9112	33.4601	3.31	28.5178	28.7040	8.32	−15.41	−16.57
	20	27.3046	28.1255	4.38	21.2130	21.7398	9.42	−28.72	−29.37
60	10	103.0052	103.8758	3.51	93.2130	93.7275	8.12	−10.51	−10.83
	20	95.4034	95.4044	4.59	81.7226	83.9474	12.92	−16.74	−13.65
120	10	252.6992	252.9956	4.13	243.5154	244.1531	15.46	−3.77	−3.62
	20	234.0136	235.9209	5.58	222.4112	224.3142	14.45	−5.22	−5.17
180	10	509.4076	509.4078	6.83	506.4052	506.4052	11.77	−0.59	−0.59
	20	495.8064	496.6537	10.66	493.0178	494.4761	22.22	−0.57	−0.44
240	10	659.7034	659.7034	7.88	656.0130	656.4622	17.69	−0.56	−0.49
	20	643.8136	644.3996	12.00	629.4130	630.3757	32.69	−2.29	−2.22
300	10	764.5106	765.7674	7.95	764.4004	764.4005	17.22	−0.01	−0.18
	20	757.1046	759.4416	15.69	743.1136	744.1450	46.51	−1.88	−2.06
Avg				7.21			18.07	−7.19	−7.10

Table 8: Result comparison of ILSSPRP with and without the adjustment operator ($T = 28800$ s).

Instance		ILS_{SPRP} without adjustment			ILS_{SPRP}			Gap_{Best} (%)	Gap_{Avg} (%)
|N|	Q	Best value	Average value	CPU	Best value	Average value	CPU		
30	10	17.8850	19.3378	5.24	14.2862	15.3446	14.00	−25.19	−26.02
	20	12.2778	13.6011	5.49	8.6850	9.1928	18.10	−41.37	−47.95
60	10	78.4790	79.5503	4.82	74.7808	75.3967	16.96	−4.95	−5.51
	20	67.8736	69.4746	5.74	63.5886	64.1407	19.10	−6.74	−8.32
120	10	227.0844	229.5996	7.66	220.4964	221.8205	24.22	−2.99	−3.51
	20	202.2880	204.2200	9.13	193.2868	195.1978	33.08	−4.66	−4.62
180	10	477.6772	477.8671	10.48	475.4784	476.1721	32.82	−0.46	−0.36
	20	463.6724	465.0630	12.46	454.5760	454.9920	36.97	−2.00	−2.21
240	10	624.5778	626.3524	13.92	623.5814	625.9040	41.91	−0.16	−0.07
	20	604.0832	606.0604	14.53	582.9796	583.6207	56.05	−3.62	−3.84
300	10	732.4820	734.7626	15.30	719.6832	720.3621	53.98	−1.78	−2.00
	20	723.5718	725.3612	17.74	711.2862	712.2615	57.28	−1.73	−1.84
Avg				10.21			33.71	−7.97	−8.85

comparing the performance of the ILS_{SPRP} with and without adjustment operator using the various instances described in Section 5.1.

Tables 6–8 present the results obtained by the proposed ILS_{SPRP} algorithm with and without the adjustment operator, under different repositioning times, respectively. CPU denotes the average computing time in seconds. Gap_{Best} and Gap_{Avg} here are the deviations of the best/average of the objective values from 20 runs obtained by the ILS_{SPRP} with the adjustment operator from the corresponding optimal solution obtained by ILS_{SPRP} without the adjustment operator, respectively. From Tables 6–8, we can see ILS_{SPRP} without the adjustment operator resulting in a shorter computation time than the ILS_{SPRP} with the adjustment operator.

Table 9: Sensitivity analysis on α.

Instance			Lingo		ILS_{SPRP}			Gap_{Best} (%)	Gap_{Avg} (%)
α	T	Q	Optimal value	CPU	Best value	Average value	CPU		
0	14400	10	48.5000	261	48.5000	48.5000	2.95	0.00	0.00
		20	43.2000	3600	40.6000	41.1300	2.98	−6.40	−5.03
	21600	10	29.8000	3600	29.5000	30.0200	3.29	−1.02	0.73
		20	22.3000	3600	20.5000	21.2000	5.05	−8.78	−5.19
	28800	10	14.4000	3600	13.0000	13.9750	5.79	−10.77	−3.04
		20	12.7000	3600	9.1000	10.0444	7.20	−39.56	−26.44
0.00001	14400	10	48.6440	233	48.6440	48.6440	4.27	0.00	0.00
		20	43.5440	3600	41.4386	41.4386	4.58	−5.08	−5.08
	21600	10	28.8148	3600	28.5178	28.7040	8.32	−1.04	−0.39
		20	23.5148	3600	21.2130	21.7398	9.42	−10.85	−8.16
	28800	10	20.0888	3600	14.2862	15.3446	14.00	−40.62	−30.92
		20	13.5862	3600	8.6850	9.1928	18.10	−56.43	−47.79
0.0001	14400	10	49.9400	181	49.9400	49.9400	2.61	0.00	0.00
		20	45.1400	3600	41.9860	42.8651	2.85	−7.51	−5.31
	21600	10	32.3360	3600	28.3420	30.3527	9.39	−14.09	−6.53
		20	23.8300	3600	22.9360	23.3009	11.71	−3.90	−2.27
	28800	10	18.4800	3600	16.6260	16.9609	9.78	−11.15	−8.96
		20	16.3140	3600	10.8320	11.7420	16.54	−50.61	−38.94
0.001	14400	10	62.9000	96	62.9000	62.9000	3.35	0.00	0.00
		20	57.5400	3600	54.4600	55.3964	4.31	−5.66	−3.87
	21600	10	50.6800	3600	47.2000	48.8240	7.33	−7.37	−3.80
		20	44.0000	3600	41.2200	42.4707	10.74	−6.74	−3.60
	28800	10	42.4000	3600	39.5800	40.5733	11.91	−7.12	−4.50
		20	36.3000	3600	36.2400	36.2550	22.24	−0.17	−0.12
Avg				3032.13			8.28	−12.29	−8.72

Table 10: Parameter setting.

Station parameter	1	2	3	4	5	6
d_i	9	6	−6	−6	8	−5
w_i	0.8	0.6	0.8	0.6	0.2	0.7

Table 11: The value of objective function and the route of the truck in different scenarios.

Scenario	The value of objective function	Route and loading/unloading quantities at stations
1	18.0648	$0 \longrightarrow 2(+5) \longrightarrow 6(-5) \longrightarrow 1(+6) \longrightarrow 4(-6) \longrightarrow 0$
2	7.6708	$0 \longrightarrow 1(+9) \longrightarrow 3(-6) \longrightarrow 2(+2) \longrightarrow 6(-5) \longrightarrow 0$

However, the Gap_{Best} and Gap_{Avg} values for all discussed instances are negative, which implies that the adjustment operator is necessary to improve the quality of the solution. As expected, when T increases, the absolute average value of Gap_{Best} and Gap_{Avg} increases. This also indicates that the ILS_{SPRP} with the adjustment operator is more effective than the ILS_{SPRP} without the adjustment operator to solve the SPRP with long repositioning time. Therefore, it can be concluded that it is beneficial to incorporate adjustment operator into the ILS_{SPRP} to solve the SPRP.

5.3. Sensitivity Analysis on α. In this set of experiments, we analyze the sensitivity of the performance of the proposed approach to the values of α. Note that increasing the value of α means that we are increasing the impact of operational time on the whole weighted objective function. When we set $\alpha = 0$, it means that the objective function only considers user dissatisfaction.

We test the sensitivity of α with a small-size instance consisting of 30 stations ($|N| = 30$) and consider the three repositioning time constraints ($T = 14400$ s, $T = 21600$ s and $T = 28800$ s) and two truck capacity constraints ($Q = 10$ and $Q = 20$).

Different values of α from 0 to 0.001 were considered, and for each value, we perform 20 runs of the experiment. The results obtained with Lingo 18 and the ILS_{SPRP} were compared in Table 9. CPU (in seconds) is the computing time for Lingo or the average computing time of 20 runs for the ILS_{SPRP}. Gap_{Best} and Gap_{Avg} are the deviations of the best/average of the objective values from 20 runs obtained by the ILS_{SPRP} from the corresponding optimal solution obtained by Lingo, respectively.

As shown in Table 9, almost all of the values of Gap_{Best} and Gap_{Avg} are negative (with only one exception for the results obtained under $\alpha = 0$, $T = 21600$ s, $Q = 10$). The average values of Gap_{Best} and Gap_{Avg} are −12.29 and −8.72, respectively. This then indicates that the proposed ILS_{SPRP}

can always obtain better solutions than Lingo under different values of α, repositioning time and truck capacity. The average computing time of the ILS_{SPRP} is 8.28 s, which is significantly less than that of Lingo (3032.13 s). Therefore, we can conclude that the ILS_{SPRP} works well under different values of α.

5.4. Sensitivity Analysis on w_i. In this section, a small example is constructed to investigate the sensitivity of the proposed model to the parameter w_i. We consider 2 scenarios and investigate the repositioning strategies, respectively:

(i) Scenario 1: we solve the model without taking into account the different values of w_i

(ii) Scenario 2: we consider different values of w_i and solve the model accordingly

We select the first six stations in the instance of $|N| = 30$ mentioned in Section 5.1, and set $T = 7200$ s and $Q = 10$. The tested values of the parameters including d_i and w_i are shown in Table 10. Note that when $d_i > 0$, station i is a pickup station; otherwise, if $d_i < 0$, station i is a delivery station. The results are displayed in Table 11, including the value of the objective function, the route of the truck, and the loading/unloading quantities at each station.

As shown in Table 11, the experimental results of scenario 1 and scenario 2 are significantly different. For pickup station 1 and 2, in scenario 2, the value of w_1 is relatively large. This means that the level of user dissatisfaction (for one bike) of station 1 is higher than that of station 2. Therefore, in order to reduce the total cost, it is necessary to increase the loading quantity at station 1 to meet the demand of station 1 as much as possible. The loading quantity of station 1 increases from 6 bikes in scenario 1 (where we disregard the different values of w_i) to 9 bikes in scenario 2 ($d_1 = 9$, the maximum loading quantity of station 1 is 9 bikes); similarly, the loading quantity of station 2 decreases from 5 bikes in scenario 1 to 2 bikes in scenario 2.

Regarding the delivery stations 3, 4, and 6, in scenario 2, because $w_3 > w_6 > w_4$, the priorities of meeting the demand are first given to station 3, followed by station 6, and last to station 4. From the results, we can see that the unloading quantity of station 3 increases from 0 bike in scenario 1 to 6 bikes in scenario 2 ($d_3 = -6$, the maximum unloading quantity of station 3 is 6 bikes), while the unloading quantity of station 4 decreases from 6 bikes in scenario 1 to 0 bike in scenario 2.

From the above results, it can be seen that different values of w_i can directly impact the loading/unloading quantities of repositioning truck at each station, as well as the route of the truck, i.e., the repositioning strategy of the BSS operators. Therefore, it can be concluded that it is necessary to take distinguishing the user dissatisfaction generated by different stations into account when modeling the bike repositioning problem, which is the key factor to the optimization of the bike repositioning.

6. Conclusions

In this paper, we study the static partial repositioning problem where the level of user dissatisfaction differs in different stations. We propose an iterated local search algorithm ILS_{SPRP} to find solutions to the problem as well as an adjustment operator to further improve the obtained solutions. We conduct a comprehensive set of experiments to study the performance of the proposed approach in terms of solution quality and computational time, compared with Lingo. The experimental results show that the proposed ILS_{SPRP} almost always outperforms Lingo from small- to large-size problems, as well as under different relative importance set to the two objectives (i.e., operational time and deviation from the target number of bikes). We also use a small example to illustrate that the unit penalty cost w_i has an effect on the repositioning route and the loading/unloading quantities at each station.

There are potential directions for future research. For instance, unbalance demands in the bike-sharing system will not only increase user dissatisfaction, but also aggravate urban traffic congestion. Therefore, in addition to considering economic benefits, we will also need to consider environmental benefits in the bike repositioning problem. How to introduce environmental benefits into the objective function would be an interesting topic for future research. Besides, we are also interested in developing efficient and effective algorithms to verify feasibility of a given solution.

Data Availability

All data included in this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors acknowledge the National Natural Science Foundation of China (grant no. 71271220), the Project of Social Science Achievement Review Committee of Hunan Province (grant no. XSP20YBZ165), and the Hengyang Social Science Foundation Project (grant no. 2020B(II)003).

References

[1] J. G. Shi, H. Si, G. Wu, Y. Su, and J. Lan, "Critical factors to achieve dockless bike-sharing sustainability in China: a stakeholder-oriented network perspective," *Sustainability*, vol. 10, no. 6, 2090 pages, 2018.

[2] G. Laporte, F. Meunier, and R. Wolfler Calvo, "Shared mobility systems: an updated survey," *Annals of Operations Research*, vol. 271, no. 1, pp. 105–126, 2018.

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

[4] M. Rainer-Harbach, P. Papazek, B. Hu, and G. R. Raidl, "Balancing bicycle sharing systems: a variable neighborhood search approach," in *Evolutionary Computation in Combinatorial Optimization*, pp. 121–132, Springer, Berlin, Germany, 2013, Lecture Notes in Computer Science, 7832.

[5] L. Di Gaspero, A. Rendl, and T. Urli, "A hybrid ACO + CP for balancing bicycle sharing systems," in *hybrid Metaheuristics*, pp. 198–212, Springer, Berlin, Germany, 2013, Lecture Notes in Computer Science, 7919.

[6] G. R. Raidl, B. Hu, M. Rainer-Harbach, and P. Papazek, "Balancing bicycle sharing systems: improving a VNS by efficiently determining optimal loading operations," in *hybrid Metaheuristics*, pp. 130–143, Springer, Berlin, Germany, 2013, Lecture Notes in Computer Science, 7919.

[7] M. Rainer-Harbach, P. Papazek, G. R. Raidl, B. Hu, and C. Kloimüllner, "PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems," *Journal of Global Optimization*, vol. 63, no. 3, pp. 597–629, 2015.

[8] L. D. Gaspero, A. Rendl, and T. Urli, "Balancing bike sharing systems with constraint programming," *Constraints*, vol. 21, no. 2, pp. 318–348, 2016.

[9] L. Di Gaspero, A. Rendl, and T. Urli, "Constraint-based approaches for balancing bike sharing systems," in *Principles and Practice of Constraint Programming*, pp. 758–773, Springer, Berlin, Germany, 2013, Lecture Notes in Computer Science, 8124.

[10] Y. Liu, W. Y. Szeto, and S. C. Ho, "A static free-floating bike repositioning problem with multiple heterogeneous vehicles, multiple depots, and multiple visits," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 208–242, 2018.

[11] P.-S. You, "A two-phase heuristic approach to the bike repositioning problem," *Applied Mathematical Modelling*, vol. 73, pp. 651–667, 2019.

[12] S. C. Ho and W. Y. Szeto, "A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem," *Transportation Research Part B: Methodological*, vol. 95, pp. 340–363, 2017.

[13] A. Pal and Y. Zhang, "Free-floating bike sharing: solving real-life large-scale static rebalancing problems," *Transportation Research Part C: Emerging Technologies*, vol. 80, pp. 92–116, 2017.

[14] Y. Wang and W. Y. Szeto, "Static green repositioning in bike sharing systems with broken bikes," *Transportation Research Part D: Transport and Environment*, vol. 65, pp. 438–457, 2018.

[15] T. Bulhões, A. Subramanian, G. Erdoğan, and G. Laporte, "The static bike relocation problem with multiple vehicles and visits," *European Journal of Operational Research*, vol. 264, no. 2, pp. 508–523, 2018.

[16] M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani, "The bike sharing rebalancing problem: mathematical formulations and benchmark instances," *OMEGA-the International Journal of Management Science*, vol. 45, pp. 7–19, 2014.

[17] M. Dell'Amico, M. Iori, S. Novellani, and T. Stützle, "A destroy and repair algorithm for the bike sharing rebalancing problem," *Computer & Operations Research*, vol. 71, pp. 149–162, 2016.

[18] R. Alvarez-Valdes, J. M. Belenguer, E. Benavent et al., "Optimizing the level of service quality of a bike-sharing system," *Omega*, vol. 62, pp. 163–175, 2016.

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

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

[21] W. Y. Szeto and C. S. Shui, "Exact loading and unloading strategies for the static multi-vehicle bike repositioning problem," *Transportation Research Part B: Methodological*, vol. 109, pp. 176–211, 2018.

[22] M. Benchimol, P. Benchimol, B. Chappert et al., "Balancing the stations of a self service "bike hire" system," *RAIRO—Operations Research*, vol. 45, no. 1, pp. 37–61, 2011.

[23] B. P. Bruck, F. Cruz, M. Iori, and A. Subramanian, "The static bike sharing rebalancing problem with forbidden temporary operations," *Transportation Science*, vol. 53, no. 3, pp. 882–896, 2019.

[24] D. Chemla, F. Meunier, and R. Wolfler Calvo, "Bike sharing systems: solving the static rebalancing problem," *Discrete Optimization*, vol. 10, no. 2, pp. 120–146, 2013.

[25] F. Cruz, A. Subramanian, B. P. Bruck, and M. Iori, "A heuristic algorithm for a single vehicle static bike sharing rebalancing problem," *Computers & Operations Research*, vol. 79, pp. 19–33, 2017.

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

[27] B. Lahoorpoor, H. Faroqi, A. Sadeghi-Niaraki, and S.-M. Choi, "Spatial cluster-based model for static rebalancing bike sharing problem," *Sustainability*, vol. 11, no. 11, p. 3205, 2019.

[28] S. C. Ho and W. Y. Szeto, "Solving a static repositioning problem in bike-sharing systems using iterated tabu search," *Transportation Research Part E: Logistics and Transportation Review*, vol. 69, pp. 180–198, 2014.

[29] Q. Tang, Z. Fu, and M. Qiu, "A bilevel programming model and algorithm for the static bike repositioning problem," *Journal of Advanced Transportation*, vol. 2019, Article ID 8641492, 19 pages, 2019.

[30] G. Erdoğan, G. Laporte, and R. Wolfler Calvo, "The static bicycle relocation problem with demand intervals," *European Journal of Operational Research*, vol. 238, no. 2, pp. 451–457, 2014.

[31] Y. Li, W. Y. Szeto, J. Long, and C. S. Shui, "A multiple type bike repositioning problem," *Transportation Research Part B: Methodological*, vol. 90, pp. 263–278, 2016.

[32] W. Y. Szeto, Y. Liu, and S. C. Ho, "Chemical reaction optimization for solving a static bike repositioning problem," *Transportation Research Prat D: Transport and Environment*, vol. 47, pp. 104–135, 2016.

[33] H. Si, J.-G. Shi, G. Wu, J. Chen, and X. Zhao, "Mapping the bike sharing research published from 2010 to 2018: a scientometric review," *Journal of Cleaner Production*, vol. 213, pp. 415–427, 2019.

[34] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[35] C. Lei and Y. Ouyang, "Continuous approximation for demand balancing in solving large-scale one-commodity pickup and delivery problems," *Transportation Research Part B: Methodological*, vol. 109, pp. 90–109, 2018.