Transportation Research Part C 125 (2021) 102895

Contents lists available at ScienceDirect

Transportation Research Part C

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

An enhanced artificial bee colony algorithm for the green bike repositioning problem with broken bikes

Yue Wang , W.Y. Szeto *

Department of Civil Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong
The University of Hong Kong Shenzhen Institute of Research and Innovation, Shenzhen, China

ARTICLE INFO

Keywords:
Green bike repositioning problem
Emissions
Broken bikes
Artificial bee colony algorithm

ABSTRACT

The Bike Repositioning Problem (BRP) has raised many researchers' attention in recent years to improve the service quality of Bike Sharing Systems (BSSs). It is mainly about designing the routes and loading instructions for the vehicles to transfer bikes among stations in order to achieve a desirable state. This study tackles a static green BRP that aims to minimize the CO_2 emissions of the repositioning vehicle besides achieving the target inventory level at stations as much as possible within the time budget. Two types of bikes are considered, including usable and broken bikes. The Enhanced Artificial Bee Colony (EABC) algorithm is adopted to generate the vehicle route. Two methods, namely heuristic and exact methods, are proposed and incorporated into the EABC algorithm to compute the loading/unloading quantities at each stop. Computational experiments were conducted on the real-world instances having 10–300 stations. The results indicate that the proposed solution methodology that relies on the heuristic loading method can provide optimal solutions for small instances. For large-scale instances, it can produce better feasible solutions than two benchmark methodologies in the literature.

1. Introduction

In Bike Sharing Systems (BSSs), people can rent shared bikes to satisfy the needs for short-distance travel, such as the connection between home and a subway station, or just a leisure trip. However, due to the existence of asymmetric bike flows, some stations may be filled-in with returned bikes while some others are empty. For example, during the morning peak, the bike stations in residential areas might be empty, whereas those in the business area might be full. In a hilly area, the stations at the bottom can be full whereas those at the top can be empty because downhill trips are usually more than uphill trips. Therefore, the bikes need to be redistributed among all the stations after a certain period, so that a desirable inventory level can be maintained at each station to satisfy upcoming users' demand. This operational problem is referred to as the Bike Repositioning Problem (BRP).

The repositioning operation is mainly about using a dedicated fleet of trucks or trailers to transport bikes among stations in order to reach the expected inventory level at each station (Raviv et al., 2013). There are generally two types of repositioning: static and dynamic. For the static repositioning that most of the current studies focus on (e.g., Cruz et al., 2017; Ho and Szeto, 2017; Liu et al., 2018; Szeto and Shui, 2018), the bike inventory and demand levels at each station are assumed to be relatively stable over time. The dynamic repositioning considers the variations of bike inventory and demand levels over time within a day (see Ghosh et al., 2017; Zhang et al., 2017; Shui and Szeto, 2018; Brinkmann et al., 2019). These two types of repositioning consider different application

* Corresponding author.
E-mail addresses: yuewhku@connect.hku.hk (Y. Wang), ceszeto@hku.hk (W.Y. Szeto).

https://doi.org/10.1016/j.trc.2020.102895
Received 16 June 2019; Received in revised form 20 March 2020; Accepted 24 November 2020
Available online 6 March 2021

scenarios.

In most BRP studies, the objectives mainly focused on two aspects: user-related factors such as the unmet demand or user dissatisfaction, and operator-related factors such as the operating costs or travel distances. It has been pointed out that many cities started to use optimization software to decrease fuel consumption when planning bike repositioning activities (Erdoğan et al., 2015). Therefore, besides satisfying the customer demand, operators of BSSs, especially those government-owned ones, may also want to minimize the emissions or fuel consumption caused by repositioning. Wang and Szeto (2018) and Shui and Szeto (2018) captured this in static and dynamic repositioning problems, respectively. Other than the emission-minimization objective function, Wang and Szeto (2018) also considered broken bikes that are required to be transported back to the depot in addition to repositioning usable bikes among stations. However, their proposed mathematical model cannot be efficiently solved by commercial solvers. Therefore, a heuristic method is required for tackling this type of problem.

In this paper, we propose a heuristic method to solve the static green bike repositioning problem (GBRP) with broken bikes introduced by Wang and Szeto (2018), but consider a single vehicle only, time budget, and a modified objective function. The problem aims to design the route and loading instructions for the repositioning vehicle such that the weighted sum of the absolute deviation from the target inventory level, the penalty caused by broken bikes at stations, and the CO_2 emissions of the repositioning vehicle are minimized. The vehicle starts from the depot, transfers bikes among bike stations to bring the inventory level close to the target value, and collects broken bikes before ending at the depot within the time budget. All the stations and the depot can be visited multiple times. The depot is assumed to have infinite capacity and supply of bikes. The challenge lies in determining the loading/unloading quantities of two types of bikes at a station and the depot in each visit so that the weighted sum of absolute deviations, penalty, and emissions can be minimized. In this study, we adopted the Enhanced Artificial Bee Colony (EABC) algorithm proposed by Szeto et al. (2011) as the backbone of the proposed method to solve large instances with up to 300 stations. An adaptive local search was incorporated to improve solution quality. The loading/unloading quantities were determined by either solving a linear programming model or a heuristic method. To demonstrate the accuracy and efficiency of the proposed method, we compared the results with those from CPLEX on the instances with up to 30 stations. Computational experiments were also conducted using the canonical genetic algorithm and the methodology proposed by Alvarez-Valdes et al. (2016). The results show that the EABC algorithm outperforms the other two algorithms for this type of problem.

The contributions of this paper are as follows:

(1) We propose a first model for the bike repositioning problem considering the emissions of the repositioning vehicle, broken bikes, and the time budget simultaneously.
(2) We develop an EABC algorithm incorporating an adaptive local search procedure to solve the green BRP with broken bikes.
(3) We provide two methods to determine the loading/unloading quantities of two types of bikes at a station and the depot in each visit.

The remainder of the paper is organized as follows. Section 2 reviews the objective functions, types of bikes, and existing methods in the BRPs. Section 3 presents the problem formulation. Section 4 provides the algorithmic structure of the EABC algorithm. In Section 5, the performance of the proposed method to solve real-world instances is demonstrated. Finally, Section 6 concludes this paper and puts forward future works.

2. Literature review

In this section, we first review the factors considered in the objective functions in BRPs. Then we look into the studies that considered broken bikes or multiple types of bikes. After that, we discuss the existing methodologies for solving BRPs.

As aforementioned, most of the current studies on BRPs considered either the user-related factors or/and the operator-related factors in the objective functions. The user-related factors mainly care about the users' demand. For example, the expected number of unsatisfied future demand (Brinkmann et al., 2019), the penalty cost of not satisfying bike or locker demand (Ho and Szeto, 2017), the expected user dissatisfaction (Zhang et al., 2017), and the deviation from the target inventory level (Kloimüllner et al., 2014; Di Gaspero et al., 2016). The operator-related factors mainly capture the operating costs, such as total travel time (Angeloudis et al., 2014; Dell'Amico et al., 2016), route cost (Cruz et al., 2017), loading and unloading operations (Rainer-Harbach et al., 2015; Liu et al., 2018), and so on. In summary, the benefits/costs of bike users and system operators are always considered when planning the repositioning work.

BRPs are closely related to vehicle routing problems as both require determining vehicle routes. In the vehicle routing problem, many studies considered the environmental issue in order to contribute to green logistics (see Lin et al. (2014) for more examples). For the BRP, besides the benefits/costs of bike users and system operators, the environmental impact of repositioning bikes also attracts our attention. Most of the static repositioning works are carried out by fossil-fueled trucks because of their larger capacity and longer cruising range compared to bike-trailers or other repositioning vehicles (Wang and Szeto, 2018). However, emitting pollutants and CO_2 may go against the original intention of developing the BSSs. Modern technology has enabled the reduction of harmful substances in the tail-gases through improving the combustion efficiency or purifying the exhaust except for CO_2 (Kopfer et al., 2014). Some of those pollutants, such as CO and CH, are burnt to CO_2 after further oxidation. CO_2 is the major component of greenhouse gases (about 26%), leading to the greenhouse effect that affects the climate and ecosystem. There are only limited studies considering the environmental issue in the BRP. To the best of our knowledge, only Wang and Szeto (2018) and Shui and Szeto (2018) captured the emission cost of CO_2 in the objective functions of BRPs.

To capture the emission cost of CO_2 in the objective functions of BRPs, a CO_2 emission model is required. Demir et al. (2011) looked into different CO_2 emission models considering factors from three categories: vehicle-related factors such as vehicle type, payload, engine, and fuel; environment-related factors such as the road condition and weather; and traffic condition. The experimental results show that fuel consumption and consequently CO_2 emissions are sensitive to the change in vehicle speed, load, and road gradient. For the static repositioning problem, which usually happens in the night, the vehicle speed can be assumed constant. In addition, the road gradient can be integrated into the asymmetric distance matrix (Wang and Szeto, 2018). If ignoring the insignificant or un-measurable factors, the CO_2 emissions can be expressed as a function of travel distance and vehicle load. Xiao et al. (2012) found a linear relationship between the fuel consumption rate and the vehicle load. Based on this study, Zhang et al. (2014) extended it to a CO_2 emission model as follows:

$$e = CER \cdot \left(\rho_0 + \frac{\rho^* - \rho_0}{Q} q\right) \cdot d \tag{2.1}$$

where e is the amount of CO_2 emissions, CER is the CO_2 emission rate, ρ_0 and ρ^* are the fuel consumption rates when the vehicle is empty and fully-loaded, respectively, Q is the vehicle capacity, d is the travel distance with a load q. This function differs from other travel cost functions in the BRP because it relates to not only travel distance but also the product of vehicle load and travel distance. This property implies that the simple greedy loading heuristic (e.g., Rainer-Harbach et al., 2013) may not work, because loading too many bikes at a stop could aggravate the emissions. Wang and Szeto (2018) also showed that a shorter distance may not necessarily lead to lower emissions. In addition, for the multiple-visit case, the number of bikes loaded onto the vehicles at each stop may affect the emissions without changing the travel distance. Therefore, the strategy to deal with the loading instructions at visited stations under an emission-minimization objective should be different from that under the distance minimization counterpart.

To the best of our knowledge, no existing BRP studies proposed a strategy to deal with the loading instructions at visited stations considering an emission-minimization objective, except for the study of Shui and Szeto (2018). But our study differs from theirs in the following aspects: First, our study considers two types of bikes, whereas Shui and Szeto (2018) only considered usable bikes; second, our study allows for multiple visits to both the depot and stations in a single planning horizon, but Szeto and Shui (2018) required that the vehicle visited each node at most once in each period. Therefore, a method is required for handling loading and unloading quantities at visited stations under an emission-minimization objective.

Besides the aforementioned emission-minimization objective function, we also considered the broken bikes in the BSSs. Broken bikes are almost unavoidable in the BSSs, and imply user dissatisfaction because they cannot be used to satisfy bike demand and additionally occupy the lockers against parking (Kaspi et al., 2017). Therefore, broken bikes need to be collected and transported away for repairs. A report on Seattle's bike share showed that in Quarter 3 of 2019, the percentage of bikes not in good working order reached 18.4% (Seattle Department of Transportation, 2019). This percentage is not small and cannot be ignored. It is therefore important to capture broken bikes in BRPs.

Few studies have proposed methodologies to consider broken bikes in BRPs. Alvarez-Valdes et al. (2016) considered both usable and broken bikes in their study. They first solved a minimum cost flow problem and used the insertion algorithm for handling usable bikes. After that, the broken bikes were inserted back to the route. However, the impact of arc flows on the objective function was not taken into account. Chang et al. (2018) considered a faulty bike-sharing recycling problem in their study, but they did not consider the repositioning of usable bikes simultaneously. Zhang et al. (2018) solved an integer linear programming model for their BRP problem with both usable and broken bikes using a hybrid discrete particle swarm optimization algorithm. The loading/unloading quantity was predetermined as the demand/supply of stations. Wang and Szeto (2018) proposed an MILP to formulate a BRP in which both an emission cost objective function, usable bikes, and broken bikes were considered, but the exact method cannot tackle large instances without incorporating heuristic methods. Therefore, solving the BRP with broken bikes is still under exploration.

Except for broken bikes, other types of bikes have also been considered in the BRPs. Li et al. (2016) studied a situation where multiple types of bikes are available. Those bikes may have different substitution and occupancy strategies. For example, the one-seat bikes can occupy the compartment for two-seat bikes, but cannot satisfy the needs for two-seat bikes. On the contrary, the two-seat bikes can satisfy the needs for one-seat bikes, but cannot use the compartment for one-seat bikes. The difference between their study and ours is that both one-seat and two-seat bikes are usable. A greedy heuristic can be used in their study to compute the loading/unloading quantities because the objective function is not related to the vehicle load. This method, however, cannot be applied to our problem.

To solve the BRPs, exact methods can provide optimal solutions to small-sized BSSs in a reasonable time. Examples include the Dantzig-Wolfe decomposition and the Benders decomposition method (Contardo et al., 2012) and the branch and cut algorithm (Dell'Amico et al., 2014; Erdoğan et al., 2014). Erdoğan et al. (2015) also presented an exact method utilizing the combinatorial Benders' cut. Nevertheless, the exact method is limited by the problem size. The best formulation by Dell'Amico et al. (2014) can only be efficiently solved instances with up to 50 stations. Erdoğan et al. (2015) only solved instances with up to 60 stations. When the number of stations increases, the computational time increases exponentially. As a result, the heuristic method is essential when solving large instances.

Various heuristic methods have been developed to solve the BRPs, such as large neighborhood search (Vogel et al., 2014; Di Gaspero et al., 2016), variable neighborhood search (Rainer-Harbach et al., 2013), tabu search (Ho and Szeto, 2014), artificial bee colony algorithm (Szeto and Shui, 2018), hybrid genetic algorithm (Li et al., 2016), and chemical reaction optimization (Liu et al., 2018). They can provide satisfactory results within a short computational time. Some of the optimization methods are summarized in Table 2-1.

Table 2-1
Summary of optimization methods for BRPs.

Optimization methods		References
Exact method	Dantzig-Wolfe and Benders decomposition	Contardo et al. (2012)
	Branch-and-cut	Dell'Amico et al. (2014), Erdoğan et al. (2014)
Approximation method	9.5 approximation algorithm	Benchimol et al. (2011)
Heuristic method		
Single-solution based	Tabu search	Chemla et al. (2013), Ho and Szeto (2014)
	Variable neighborhood search	Raidl et al. (2013), Rainer-Harbach et al. (2013), Kloimüllner et al. (2014), Rainer-Harbach et al. (2015)
	GRASP	Kloimüllner et al. (2014), Rainer-Harbach et al. (2015)
	Iterated local search	Cruz et al. (2017)
	Large neighborhood search	Vogel et al. (2014), Di Gaspero et al. (2016), Ho and Szeto (2017)
Population-based	Hybrid genetic algorithm	Li et al. (2016)
	Memetic algorithm	Ting and Liao (2013)
	Ant colony optimization	Di Gaspero et al. (2013)
	Chemical reaction optimization	Szeto et al. (2016), Liu et al. (2018)
	Artificial bee colony algorithm	Shui and Szeto (2018), Szeto and Shui (2018)
Hybridization of heuristics	GRASP + Path Relinking	Papazek et al. (2014), Ho and Szeto (2016)

The Artificial Bee Colony (ABC) algorithm, which also belongs to the heuristic method, was first proposed by Karaboga (2005) for solving the unimodal and multi-modal numerical optimization problems. It has been used for solving various optimization problems, including the capacitated vehicle routing problem (e.g., Szeto et al., 2011), the flexible job shop scheduling problem (e.g., Li et al., 2011), the load dispatch problem (e.g., Hemamalini and Simon, 2010), the traveling salesman problem (e.g., Karaboga and Gorkemli, 2011), and so on. The interested reader is referred to the paper by Karaboga et al. (2014). The wide application of the ABC algorithm can be due to several reasons. First, the mechanism of the ABC algorithm is simple, thus straightforward to be applied to various optimization problems. Second, it has a good balance between the diversification and intensification procedures through different types of bees. Third, it has the flexibility to hybridize with other methods. Therefore, the ABC algorithm has already been applied to the BRPs to generate vehicle routes (see Szeto and Shui (2018) for example) and was also adopted in our study.

In summary, the environmental impact of bike repositioning lacks sufficient attention among the literature of BRPs, the majority of which only considered the benefits/costs of bike users and system operators. The existence of broken bikes also brings up our attention. However, existing exact methods are not adequate to solve the BRP considering those properties. Given that the ABC algorithm has achieved great success in many other optimization problems, we see the potentials of improving it in solving the described problem. Therefore, in the next section, we explain the algorithmic structure of our improved version of the ABC algorithm in detail.

3. Problem formulation

In this study, we considered a BSS with one depot and multiple bike stations. A green BRP is formulated from the perspective of public repositioning operator to rebalance the BSS as much as possible, and reduce the CO_2 emissions produced during the repositioning at the same time. Denote S as the set of stations and S_0 as the set of nodes including the stations and the depot. The depot is assumed to have infinite capacity and supply of bikes. It is assumed that we have perfect information about each station $i \in S$, including its capacity c_i, the initial inventory level of usable bikes p_i and broken bikes b_i, and the target inventory level of usable bikes q_i. The travel distances d_{ij} and travel times t_{ij} between all pairs of nodes $i.j \in S_0$ are known. A bike station is defined as a pickup station if $p_i > q_i$, a drop-off station if $p_i < q_i$, or a balanced station if $p_i = q_i$. A repositioning vehicle with limited capacity Q starts from the depot ($i = 0$), collects bikes from pickup stations, deliver bikes to drop-off stations, collects the broken bikes at any stations, and ends at the depot. The total repositioning time, including the travel time of the vehicle and handling time of bikes, should not exceed the time budget T. The loading/unloading time of a bike is denoted by L. The vehicle is allowed to have an initial load when it departs from the depot, or/and return to the depot with redundant usable bikes. The redundant usable bikes and broken bikes are unloaded from the vehicle to the depot. The depot and stations can be visited multiple times during the repositioning work. It is assumed that the vehicle turns off the engine during idling at stations and hence emissions are only produced during traveling and no emissions are produced during idling. The green BRP aims to design the route and loading instructions for the repositioning vehicle such that the weighted sum of the absolute deviation from the target inventory level, the penalty induced by broken bikes, and the CO_2 emissions of the vehicle are minimized.

The parameters in the mathematical model include the following:

A	The maximum number of stops of the vehicle
CER	The CO_2 emission rate (kg/liter)
M	A very large number
Q	The capacity of the vehicle
L	The time for loading or unloading a bike
T	The time budget for completing the repositioning work
b_i	The number of broken bikes at station $i \in S$
d_{ij}	Travel distance from node $i \in S_0$ to node $j \in S_0$
t_{ij}	Travel time from node $i \in S_0$ to node $j \in S_0$
p_i	The initial inventory level of usable bikes at station $i \in S$
q_i	The target inventory level of usable bikes at station $i \in S$
c_i	The capacity of station $i \in S$
ρ^*	The full-load fuel consumption rate of the vehicle (liters/km)
ρ_0	The empty-load fuel consumption rate of the vehicle (liters/km)
θ	The penalty (a weighting factor) for a remaining broken bike at bike stations
w	The weighting factor of the vehicle's total CO_2 emissions

The decision variables include the following:

x_{ia}	$= \begin{cases} 1, & \text{if node } i \text{ is the } a^{th} \text{ stop of the vehicle} \\ 0, & \text{otherwise} \end{cases}$
y^B_{ia}	It indicates the number of broken bikes loaded onto or unloaded from the vehicle at node i at its a^{th} stop. If $y^B_{ia} > 0$, y^B_{ia} broken bikes are loaded onto the vehicle; otherwise, $-y^B_{ia}$ broken bikes are unloaded from the vehicle.
y^G_{ia}	It indicates the number of usable bikes loaded onto or unloaded from the vehicle at node i at its a^{th} stop. If $y^G_{ia} > 0$, y^G_{ia} usable bikes are loaded onto the vehicle; otherwise, $-y^G_{ia}$ usable bikes are unloaded from the vehicle.

The auxiliary variables include the following:

f^B_a	The number of broken bikes on the vehicle when leaving its a^{th} stop (a non-negative variable)
f^G_a	The number of usable bikes on the vehicle when leaving its a^{th} stop (a non-negative variable)
h_a	The time of the vehicle leaving the a^{th} stop
z_a	The emissions produced when the vehicle travels from the $(a-1)^{\text{th}}$ stop to the a^{th} stop

Then the mathematical model of the described problem can be formulated as follows:

$$\text{Minimize } \sum_{i \in S} \left[\left| p_i - \sum\nolimits_{a=1}^{A} y^G_{ia} - q_i \right| + \theta \cdot \left(b_i - \sum\nolimits_{a=1}^{A} y^B_{ia} \right) \right] + w \cdot \sum_{a=2}^{A} z_a \tag{3.1}$$

Subject to:

$$\sum_{i \in S_0} x_{ia} = 1, \quad \forall a = 1, \cdots, A, \tag{3.2}$$

$$x_{01} = 1, \tag{3.3}$$

$$x_{0A} = 1, \tag{3.4}$$

$$-Q \cdot x_{ia} \le y^G_{ia} \le Q \cdot x_{ia}, \quad \forall i \in S_0, a = 1, \cdots, A, \tag{3.5}$$

$$0 \le y^B_{ia} \le b_i \cdot x_{ia}, \quad \forall i \in S, a = 1, \cdots, A, \tag{3.6}$$

$$-Q \cdot x_{0a} \le y^B_{0a} \le 0, \quad \forall a = 1, \cdots, A, \tag{3.7}$$

$$(p_i - q_i) \cdot y^G_{ia} \ge 0, \quad \forall i \in S, a = 1, \cdots, A, \tag{3.8}$$

$$p_i - \sum\nolimits_{a=1}^{A} y^G_{ia} \ge 0, \quad \forall i \in S, \tag{3.9}$$

$$b_i - \sum\nolimits_{a=1}^{A} y_{ia}^{B} \geq 0, \quad \forall i \in S, \tag{3.10}$$

$$p_i - \sum\nolimits_{a=1}^{A'} y_{ia}^{G} + b_i - \sum\nolimits_{a=1}^{A'} y_{ia}^{B} \leq c_i, \quad \forall i \in S, A' = 1, \cdots, A, \tag{3.11}$$

$$f_a^G = f_{a-1}^G + \sum_{i \in S_0} y_{ia}^G, \quad \forall a = 2, \cdots, A, \tag{3.12}$$

$$f_a^B = f_{a-1}^B + \sum_{i \in S_0} y_{ia}^B, \quad \forall a = 2, \cdots, A, \tag{3.13}$$

$$f_1^G = y_{01}^G, \tag{3.14}$$

$$f_1^B = y_{01}^B, \tag{3.15}$$

$$f_A^G = 0, \tag{3.16}$$

$$f_A^B = 0, \tag{3.17}$$

$$f_a^G + f_a^B \leq Q, \quad \forall a = 1, \cdots, A, \tag{3.18}$$

$$h_a - h_{a-1} \geq \sum_{i \in S_0} t_{ij} x_{i,a-1} + L(|\sum_{i \in S_0} y_{ia}^G| + |\sum_{i \in S_0} y_{ia}^B|) - M\left(1 - x_{ja}\right), \quad \forall j \in S_0, a = 2, \cdots, A, \tag{3.19}$$

$$h_1 = L y_{01}^G, \tag{3.20}$$

$$h_A \leq T, \tag{3.21}$$

$$z_a \geq CER \cdot \left[\rho_0 + \frac{\rho^* - \rho_0}{Q} \cdot \left(f_{a-1}^G + f_{a-1}^B\right)\right] \cdot d_{ij} - M \cdot \left[2 - \left(x_{i,a-1} + x_{ja}\right)\right], \quad \forall i, j \in S_0, a = 2, \cdots, A, \tag{3.22}$$

$$x_{ia} \in \{0, 1\}, \quad \forall i \in S_0, a = 1, \cdots, A, \tag{3.23}$$

$$y_{ia}^G, \text{integers}, \quad \forall i \in S_0, a = 1, \cdots, A, \tag{3.24}$$

$$y_{ia}^B, \text{integers}, \quad \forall i \in S_0, a = 1, \cdots, A, \tag{3.25}$$

$$f_a^G \geq 0, \quad \forall a = 1, \cdots, A, \tag{3.26}$$

$$f_a^B \geq 0, \quad \forall a = 1, \cdots, A, \tag{3.27}$$

$$h_a \geq 0, \quad \forall a = 1, \cdots, A, \tag{3.28}$$

$$z_a \geq 0, \quad \forall a = 2, \cdots, A. \tag{3.29}$$

The objective (3.1) is to minimize the weighted sum of the absolute deviation from the target inventory level, the penalty induced by broken bikes, and the CO_2 emissions generated by the vehicle during the repositioning. Constraints (3.2) limit that the vehicle can only visit one station at each stop. Constraints (3.3) and (3.4) ensure that the vehicle starts from and ends at the depot. Constraints (3.5)–(3.7) ensure that no loading/unloading actions happen at those unvisited nodes. In addition, broken bikes can only be collected at stations and delivered to the depot. Constraints (3.8) are monotonicity constraints, i.e., we only load bikes from pickup stations, and unload bikes to drop-off stations. Constraints (3.9)–(3.11) indicate that the final inventory level of each type of bike at each station must be no less than zero, and the inventory level at each station does not exceed the station's capacity at each visit. Constraints (3.12)–(3.17) define the vehicle load after leaving each stop. Constraints (3.18) are vehicle capacity constraints. Constraints (3.19) define the minimum additional time required to serve the next stop. This additional time includes the travel time between the (a-1)$^{\text{th}}$ stop and the a^{th} stop, and the handling time for the bikes loaded/unloaded at the a^{th} stop. Constraint (3.20) defines the handling time required at the first stop, which is the depot. Constraint (3.21) implies that the vehicle must finish its work within the time budget. Constraints (3.22) define the CO_2 emissions produced when the vehicle travels from the (a-1)$^{\text{th}}$ stop to the a^{th} stop. Constraints (3.23)–(3.29) are domain constraints.

The absolute deviation from the target level at station $i \in S$ in the objective function can be linearized by replacing it with an auxiliary variable δ_i and introducing additional constraints (3.30)–(3.32):

$$\delta_i \geq p_i - \sum\nolimits_{a=1}^{A} y_{ia}^{G} - q_i, \quad \forall i \in S, \tag{3.30}$$

$$\delta_i \geq q_i - p_i + \sum_{a=1}^{A} y_{ia}^G, \quad \forall i \in S, \tag{3.31}$$

$$\delta_i \geq 0, \quad \forall i \in S. \tag{3.32}$$

The following valid inequalities can be added to reduce the search space and speed up the running time of this formulation (Raviv et al., 2013):

$$x_{0a} + x_{0,a+1} - x_{0,a+2} \leq 1, \quad \forall a = 1, \cdots, A-2, \tag{3.33}$$

$$x_{ia} + x_{i,a+1} \leq 1, \quad \forall i \in S, a = 2, \cdots, A-1, \tag{3.34}$$

$$h_a - h_{a-1} \geq L\left(\left|\sum_{i\in S_0} y_{ia}^G\right| + \left|\sum_{i\in S_0} y_{ia}^B\right|\right) + \sum_{j\in S_0} \widetilde{T}_j x_{ja}, \quad \forall a = 2, \cdots, A, \tag{3.35}$$

$$h_a - h_{a-1} \geq L\left(\left|\sum_{i\in S_0} y_{ia}^G\right| + \left|\sum_{i\in S_0} y_{ia}^B\right|\right) + \sum_{i\in S_0} \widehat{T}_i x_{i,a-1}, \quad \forall a = 2, \cdots, A, \tag{3.36}$$

where $\widetilde{T}_j = \min_{i\in S_0: i\neq j} t_{ij}$, and $\widehat{T}_i = \min_{j\in S_0: i\neq j} t_{ij}$.

Constraints (3.33) imply that if the vehicle visits the depot at two consecutive stops, it remains there afterward. Constraints (3.34) guarantee that the vehicle does not visit the same station at two consecutive stops. Constraints (3.35) state that the minimum additional time for serving the a^{th} stop is the sum of the handling time for loading/unloading bikes at this stop and the minimum travel time from any other node to the a^{th} stop. Constraints (3.36) replace the latter term in the previous constraints using the minimum travel time from the $(a\text{-}1)^{\text{th}}$ stop to any other node.

Constraints (3.19), (3.35), and (3.36) can be linearized by replacing them with the following constraints (3.37)–(3.48):

$$h_a - h_{a-1} \geq \sum_{i\in S_0} t_{ij} x_{i,a-1} + L\left(\sum_{i\in S_0} y_{ia}^G + \sum_{i\in S_0} y_{ia}^B\right) - M\left(1 - x_{ja}\right), \quad \forall j \in S_0, a = 2, \cdots, A, \tag{3.37}$$

$$h_a - h_{a-1} \geq \sum_{i\in S_0} t_{ij} x_{i,a-1} + L\left(-\sum_{i\in S_0} y_{ia}^G + \sum_{i\in S_0} y_{ia}^B\right) - M\left(1 - x_{ja}\right), \quad \forall j \in S_0, a = 2, \cdots, A, \tag{3.38}$$

$$h_a - h_{a-1} \geq \sum_{i\in S_0} t_{ij} x_{i,a-1} + L\left(\sum_{i\in S_0} y_{ia}^G - \sum_{i\in S_0} y_{ia}^B\right) - M\left(1 - x_{ja}\right), \quad \forall j \in S_0, a = 2, \cdots, A, \tag{3.39}$$

$$h_a - h_{a-1} \geq \sum_{i\in S_0} t_{ij} x_{i,a-1} + L\left(-\sum_{i\in S_0} y_{ia}^G - \sum_{i\in S_0} y_{ia}^B\right) - M\left(1 - x_{ja}\right), \quad \forall j \in S_0, a = 2, \cdots, A, \tag{3.40}$$

$$h_a - h_{a-1} \geq L\left(\sum_{i\in S_0} y_{ia}^G + \sum_{i\in S_0} y_{ia}^B\right) + \sum_{j\in S_0} \widetilde{T}_j x_{ja}, \quad \forall a = 2, \cdots, A, \tag{3.41}$$

$$h_a - h_{a-1} \geq L\left(-\sum_{i\in S_0} y_{ia}^G + \sum_{i\in S_0} y_{ia}^B\right) + \sum_{j\in S_0} \widetilde{T}_j x_{ja}, \quad \forall a = 2, \cdots, A, \tag{3.42}$$

$$h_a - h_{a-1} \geq L\left(\sum_{i\in S_0} y_{ia}^G - \sum_{i\in S_0} y_{ia}^B\right) + \sum_{j\in S_0} \widetilde{T}_j x_{ja}, \quad \forall a = 2, \cdots, A, \tag{3.43}$$

$$h_a - h_{a-1} \geq L\left(-\sum_{i\in S_0} y_{ia}^G - \sum_{i\in S_0} y_{ia}^B\right) + \sum_{j\in S_0} \widetilde{T}_j x_{ja}, \quad \forall a = 2, \cdots, A, \tag{3.44}$$

$$h_a - h_{a-1} \geq L\left(\sum_{i\in S_0} y_{ia}^G + \sum_{i\in S_0} y_{ia}^B\right) + \sum_{i\in S_0} \widehat{T}_i x_{i,a-1}, \quad \forall a = 2, \cdots, A, \tag{3.45}$$

$$h_a - h_{a-1} \geq L\left(-\sum_{i\in S_0} y_{ia}^G + \sum_{i\in S_0} y_{ia}^B\right) + \sum_{i\in S_0} \widehat{T}_i x_{i,a-1}, \quad \forall a = 2, \cdots, A, \tag{3.46}$$

$$h_a - h_{a-1} \geq L\left(\sum_{i\in S_0} y_{ia}^G - \sum_{i\in S_0} y_{ia}^B\right) + \sum_{i\in S_0} \widehat{T}_i x_{i,a-1}, \quad \forall a = 2,\cdots,A, \tag{3.47}$$

$$h_a - h_{a-1} \geq L\left(-\sum_{i\in S_0} y_{ia}^G - \sum_{i\in S_0} y_{ia}^B\right) + \sum_{i\in S_0} \widehat{T}_i x_{i,a-1}, \quad \forall a = 2,\cdots,A. \tag{3.48}$$

4. Methodology

In this section, we first introduce how the EABC algorithm was modified from the original ABC algorithm. Next, we explain the solution representation and how the initial routes were generated. After that, we describe the computation of loading/unloading quantity at which stop, after which the absolute deviation from the target inventory level at stations, the remaining broken bikes at stations, and the vehicle load can be derived accordingly and then the objective value can be calculated. Finally, we describe the neighborhood search in this algorithm.

The ABC algorithm is a meta-heuristic that mimics the behavior of bees when searching for food sources. It is a process of exploiting the neighborhood of a food source and exchanging information among bees. A food source usually represents a solution with a fitness value indicating its relative attractiveness over other food sources. The fitness value can be determined based on the objective function of the optimization problem (function (3.1)). For a minimization problem, the fitness function can be the reciprocal of the objective function.

The initial population is a set of randomly generated food sources (solutions) at the beginning of the search. One employed bee is assigned to each food source. Each of the employed bees searches for a new food source (neighbor solution) around the assigned one. The latter is replaced if the former has better quality (fitness value). Then the employed bees share the information about food quality with other bees after returning to the hive. The onlooker bees in the hive decide which food source to continue exploiting according to the information they received. In the onlooker bee phase, a food source with better quality attracts more bees going there, hence having more opportunities to be exploited. When a food source is exhausted (no better food can be found near it in *limit* (a fixed number) successive iterations, this food source is abandoned and replaced with a new one found by a scout bee. Three types of bees work successively and repeatedly until the termination criterion is satisfied (e.g., the maximum number of iterations is reached). The best solution memorized so far is the final result.

Based on the original ABC algorithm, Szeto et al. (2011) proposed an enhanced version of it, in which two steps were modified. First, in the onlooker bee phase, a new and better food source replaces the one with the most failed trials for the improvement of its quality among all the existing food sources, rather than the old one. Second, in the scout bee phase, the exhausted food source is replaced with a new one found in its neighborhood, instead of a randomly generated one. The main steps of the EABC algorithm are summarized in Algorithm 1.

Algorithm 1. The enhanced artificial bee colony algorithm

Generate a set of initial solutions $F = \{x_1, x_2, \ldots, x_m\}$, where m is the number of solutions
Evaluate all the solutions and derive fitness values $f(x_i), i = 1,\cdots,m$ as the reciprocal of the objective value
Initialize the number of failed trials for finding a better neighbor solution around solution i, i.e., set $l_i = 0, i = 1,\cdots,m$
while Termination condition is not met **do**
  //Employed bee phase:
  for each solution x_i **do**
    Randomly select $\tilde{x} \in N(x_i)$, where $N(\cdot)$ is the set of neighbor solutions
    if $f(\tilde{x}) > f(x_i)$ **then**
      $x_i \leftarrow \tilde{x}, l_i \leftarrow 0$
    else
      $l_i \leftarrow l_i + 1$
    end if
  //Onlooker bee phase:
  Associate each solution x_i with a set $G_i = \varnothing$
  for each onlooker bee **do**
    Select $x_i \in F$ with the probability $p(x_i) = f(x_i)/\sum_{j=1}^{m} f(x_j)$
    Randomly select $\tilde{x} \in N(x_i), G_i \leftarrow G_i \cup \{\tilde{x}\}$
  for each solution x_i **do**
    if $G_i \neq \varnothing$ **then**
      Set $\widehat{x} = \arg max_{\sigma\in G_i} f(\sigma)$
      if $f(\widehat{x}) > f(x_i)$ **then**
        Select $\tilde{x}_j \in F$ with $j = \arg max_{k=1,\cdots,m} l_k$
        if $f(\widehat{x}) > f\left(\tilde{x}_j\right)$ **then**
          $\tilde{x}_j \leftarrow \widehat{x}, l_j \leftarrow 0$
        end if
      else
        $l_i \leftarrow l_i + 1$

(*continued on next page*)

(*continued*)

```
    end if
  end if
//Scout bee phase:
for each solution x_i
  if l_i ≥ limit then
    Randomly select x̃ ∈ N(x_i)
    x_i ← x̃, l_i = 0
  end if
end while
```

4.1. Solution representation and route initialization

A solution to the BRP consists of both a visiting sequence (also referred to as a route) and a loading instruction sequence that indicates the loading instructions at each stop. For easy understanding and handling, each route was encoded directly using the station index and the depot index (i.e., 0). The start and end of a route are the depot (i.e., 0), and there can be other stops at the depot in the route. An example of the encoding method for the route of a single vehicle is demonstrated in Fig. 4-1. The number beside the arc shows the traversing order. Two visits to Station 2 are encoded using the same index "2". After the first visit, the vehicle travels to station 4, but after the second visit, the vehicle goes back to the depot. The vehicle stops over at the depot between two visits to Station 5. Conversely, the encoded sequence can be decoded to the route of the vehicle.

Each loading instruction has two parts: one for usable bikes and the other for broken bikes. For each type of bike, the instruction defines the action (loading, unloading, or no action) to be taken and the quantity involved. A loading action is denoted using a positive integer that equals the number of bikes collected at the corresponding stop. An unloading action is denoted using a negative integer, with the absolute value equal to the number of bikes unloaded at that stop. No action is denoted using a zero.

In the initialization phase, a set of initial routes were first generated. A route was first initialized using the depot as the start. For the current (incomplete) route, we define a set of *reachable stations* for the last stop of this route as the set of stations that allow the vehicle to reach those stations from the last stop, perform at least one loading/unloading action there, and return to the depot within the time budget. We define the *gain* as the penalty reduction if a specific station is visited, and define the *cost* as the time increase including the travel time and loading/unloading time. Then for each reachable station, we have a function $g(\cdot) = \frac{gain}{cost}$ to evaluate the contribution of appending this station to the end of the current route. All those reachable stations were sorted according to the value of $g(\cdot)$ in descending order. Then the restricted candidate list (RCL) consists of top, say 40%, of the stations in the set of reachable stations, and also the depot. The node to be appended was randomly selected from the RCL. This process was repeated until no station was reachable within the time budget. Then the final depot was appended to the end of the route. Other routes were generated in the same manner. The loading instructions for the initial routes were generated by the methods described in Section 4.2.

4.2. Computation of the loading instructions

For the routes generated either in the initialization phase or in the intermediate steps of the algorithm, the loading instructions need to be determined before calculating the objective value of this solution. Section 4.2.1 introduces a network flow model based on the linear programming approach proposed by Rainer-Harbach et al. (2013) (i.e., the exact method). The differences of this model from theirs mainly are as follows: the vehicle can have a load of bikes when it starts from or/and ends at the depot; the depot can be visited multiple times and act as both supply and demand nodes; two types of bikes with different loading/unloading requirements are considered; the time limit is a hard constraint. Section 4.2.2 presents a heuristic method as an alternative to compute the loading/unloading quantities. Three components of this method are described in detail, namely the initial assignment, repair, and quantity adjustment.

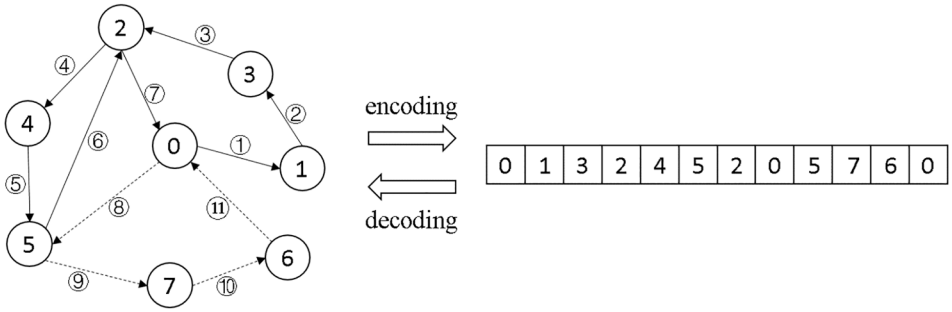

Fig. 4–1. The encoding and decoding for a route of a single vehicle with multiple visits to both stations and the depot.

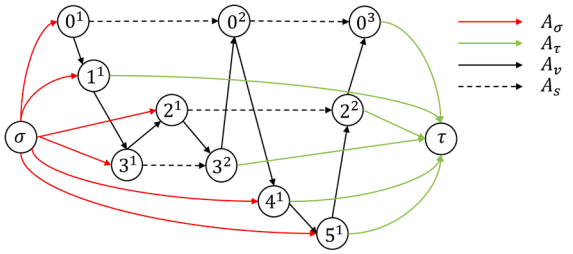

Fig. 4–2. Example of the graph for the visiting sequence 0-1-3-2-3-0-4-5-2-0.

4.2.1. The network flow model for computing the loading/unloading quantities

We define a directed graph $G = (V_G, A_G)$ for a given node visiting sequence. The node set $V_G = V_s \cup V_0 \cup \{\sigma, \tau\}$, where V_s consists of all stops at stations and V_0 includes all stops at the depot. Each stop is represented using copies i^j with j indicating the order of visits to node i. For example, 3^2 means the 2nd visit to node 3. σ and τ are the source and sink respectively. The arc set $A_G = A_\sigma \cup A_\tau \cup A_v \cup A_s$, where (1) A_σ consists of all arcs between the source σ and the first visit to each node i (i.e., i^1) in the visiting sequence, with the capacity p_i for usable bikes and b_i for broken bikes, (2) A_τ consists of all arcs between the last visit to each station i (denoted by i^{last}) in the visiting sequence to the sink τ with capacity c_i, (3) A_v is the set of the arcs between two consecutive stops in the visiting sequence with capacity Q, and (4) A_s is the set of the arcs between two consecutive visits to the same node i with capacity c_i. The flow on an arc belonging to A_σ describes the initial inventory level at each node and the flow on an arc belonging to A_τ correspondingly describes the final inventory level. The flow on an arc belonging to A_v indicates the vehicle load when traversing the corresponding link. The flow on an arc belonging to A_s means the inventory level of bikes at each node between two consecutive visits to it.

We set the initial inventory level of usable bikes at the depot p_0 to be a large value and the capacity c_0 is assumed to be infinite. The initial inventory level of broken bikes at the depot b_0 is set to be 0.

An example of the directed graph for the visiting sequence 0-1-3-2-3-0-4-5-2-0 is presented in Fig. 4-2.

Then we can formulate a linear programming model for the defined network. The decision variables are the arc flows $f^G_{u,v}$ and $f^B_{u,v}$, $\forall (u,v) \in A_G$ for usable bikes and broken bikes, respectively. The auxiliary variables δ^G_i and δ^B_i are used to derive the absolute deviation from the target inventory level of usable and broken bikes, respectively. For clear presentation, we define two constants $\beta_1 = CER \cdot \rho_0$, and $\beta_2 = CER \cdot \frac{\rho^* - \rho_0}{Q}$. Then the linear programming model can be formulated as follows:

$$\text{Minimize} \sum_{i \in S} \left(\delta^G_i + \theta \delta^B_i\right) + w \left[\beta_1 \sum_{(u,v) \in A_v} d_{u,v} + \beta_2 \sum_{(u,v) \in A_v} d_{u,v} \left(f^G_{u,v} + f^B_{u,v}\right)\right] \tag{4.1}$$

Subject to:

$$\sum_{\left(u,i^j\right) \in A_v} f^G_{u,i^j} + \sum_{\left(u,i^j\right) \in A_\sigma \cup A_s} f^G_{u,i^j} = \sum_{\left(i^j,v\right) \in A_v} f^G_{i^j,v} + \sum_{\left(i^j,v\right) \in A_\tau \cup A_s} f^G_{i^j,v}, \quad \forall i^j \in V_s \cup V_0, \tag{4.2}$$

$$\sum_{\left(u,i^j\right) \in A_v} f^B_{u,i^j} + \sum_{\left(u,i^j\right) \in A_\sigma \cup A_s} f^B_{u,i^j} = \sum_{\left(i^j,v\right) \in A_v} f^B_{i^j,v} + \sum_{\left(i^j,v\right) \in A_\tau \cup A_s} f^B_{i^j,v}, \quad \forall i^j \in V_s \cup V_0, \tag{4.3}$$

$$f^G_{\sigma,i^1} = p_i, \quad \forall i \in S_0 : \left(\sigma, i^1\right) \in A_\sigma, \tag{4.4}$$

$$f^B_{\sigma,i^1} = b_i, \quad \forall i \in S_0 : \left(\sigma, i^1\right) \in A_\sigma, \tag{4.5}$$

$$f^G_{i^{\text{last}},\tau} - q_i \le \delta^G_i, \quad \forall i \in S : \left(i^{\text{last}}, \tau\right) \in A_\tau, \tag{4.6}$$

$$q_i - f^G_{i^{\text{last}},\tau} \le \delta^G_i, \quad \forall i \in S : \left(i^{\text{last}}, \tau\right) \in A_\tau, \tag{4.7}$$

$$\delta^B_i = f^B_{i^{\text{last}},\tau}, \quad \forall i \in S : \left(i^{\text{last}}, \tau\right) \in A_\tau, \tag{4.8}$$

$$f^G_{i^{\text{last}},\tau} \le c_i, \quad \forall i \in S : \left(i^{\text{last}}, \tau\right) \in A_\tau, \tag{4.9}$$

$$f^B_{i^{\text{last}},\tau} \le b_i, \quad \forall i \in S : \left(i^{\text{last}}, \tau\right) \in A_\tau, \tag{4.10}$$

$$f^G_{i^{\text{last}},\tau} + f^B_{i^{\text{last}},\tau} \le c_i, \quad \forall i \in S : \left(i^{\text{last}}, \tau\right) \in A_\tau, \tag{4.11}$$

$$f^G_{u^j,i^j} \le Q, \quad \forall \left(u^j, i^j\right) \in A_v, \tag{4.12}$$

$$f^{B}_{u^{i},i^{j}} \le Q, \quad \forall (u^{i}, i^{j}) \in A_v, \tag{4.13}$$

$$f^{G}_{u^{i},i^{j}} + f^{B}_{u^{i},i^{j}} \le Q, \quad \forall (u^{i}, i^{j}) \in A_v, \tag{4.14}$$

$$f^{G}_{i^{i},i^{j}} \le c_i, \quad \forall i \in S, (i^{i}, i^{j}) \in A_s, \tag{4.15}$$

$$f^{B}_{i^{i},i^{j}} \le b_i, \quad \forall i \in S, (i^{i}, i^{j}) \in A_s, \tag{4.16}$$

$$f^{G}_{i^{i},i^{j}} + f^{B}_{i^{i},i^{j}} \le c_i, \quad \forall i \in S, (i^{i}, i^{j}) \in A_s, \tag{4.17}$$

$$(p_i - q_i)\left(f^{G}_{i^{i},v} - f^{G}_{u,i^{j}}\right) \ge 0, \quad \forall i^{j} \in V_s, (u, i^{j}) \in A_v, (i^{j}, v) \in A_v, \tag{4.18}$$

$$f^{B}_{i^{i},v} - f^{B}_{u,i^{j}} \ge 0, \quad \forall i^{j} \in V_s, (u, i^{j}) \in A_v, (i^{j}, v) \in A_v, \tag{4.19}$$

$$f^{B}_{0^{i},i^{j}} = 0, \quad \forall (0^{i}, i^{j}) \in A_v, \tag{4.20}$$

$$L \sum_{i^{j} \in V_s \cup V_0} \left(\left| \sum_{(u,i^{j}) \in A_v} f^{G}_{u,i^{j}} - \sum_{(i^{i},v) \in A_v} f^{G}_{i^{i},v} \right| + \left| \sum_{(u,i^{j}) \in A_v} f^{B}_{u,i^{j}} - \sum_{(i^{i},v) \in A_v} f^{B}_{i^{i},v} \right| \right) + \sum_{(u,v) \in A_v} t_{u,v} \le T, \tag{4.21}$$

$$\delta^{G}_{i} \ge 0, \quad \forall i \in S, \tag{4.22}$$

$$f^{G}_{u,v} \ge 0, \quad \forall (u,v) \in A_G, \tag{4.23}$$

$$f^{B}_{u,v} \ge 0, \quad \forall (u,v) \in A_G. \tag{4.24}$$

The objective function (4.1) comes from function (3.1) except that the routing variables are given here. Constraints (4.2) and (4.3) are flow conservation constraints on the node set V_s. The flows on arcs $(\sigma, i^{j}) \in A_\sigma$ are set to the initial inventories of two types of bikes as stated in constraints (4.4) and (4.5). Constraints (4.6)–(4.8) define the absolute deviation of the inventory level of each type of bike from the target level. Constraints (4.9)–(4.17) are link capacity constraints. Constraints (4.18) and (4.19) are the monotonicity constraints on the inventory level of two types of bikes at stations. Constraints (4.20) limit that the vehicle should not carry broken bikes when departing from the depot. Constraint (4.21) is the time limit constraint. Constraints (4.22)–(4.24) are domain constraints. It is noted that δ^{B}_{i} is not a decision variable and can be substituted with $f^{B}_{i^{\text{last}},\tau}$ (equation (4.8)).

Constraints (4.21) can be linearized by introducing auxiliary variables $y^{G,+}_{i_j}, y^{G,-}_{i_j}, y^{B,+}_{i_j}, y^{B,-}_{i_j}, \forall i^{j} \in V_s \cup V_0$. Then constraints (4.21) can be replaced with constraints (4.25)–(4.31):

$$L \sum_{i^{j} \in V_s \cup V_0} \left(y^{G,+}_{i_j} + y^{G,-}_{i_j} + y^{B,+}_{i_j} + y^{B,-}_{i_j} \right) + \sum_{(u,v) \in A_v} t_{u,v} \le T, \tag{4.25}$$

$$\left(\sum_{(u,i^{j}) \in A_v} f^{G}_{u,i^{j}} - \sum_{(i^{i},v) \in A_v} f^{G}_{i^{i},v} \right) + y^{G,+}_{i_j} - y^{G,-}_{i_j} = 0, \quad \forall i^{j} \in V_s \cup V_0, \tag{4.26}$$

$$\left(\sum_{(u,i^{j}) \in A_v} f^{B}_{u,i^{j}} - \sum_{(i^{i},v) \in A_v} f^{B}_{i^{i},v} \right) + y^{B,+}_{i_j} - y^{B,-}_{i_j} = 0, \quad \forall i^{j} \in V_s \cup V_0, \tag{4.27}$$

$$0 \le y^{G,+}_{i_j} \le Q, \quad \forall i^{j} \in V_s \cup V_0, \tag{4.28}$$

$$0 \le y^{G,-}_{i_j} \le Q, \quad \forall i^{j} \in V_s \cup V_0, \tag{4.29}$$

$$0 \le y^{B,+}_{i_j} \le Q, \quad \forall i^{j} \in V_s \cup V_0, \tag{4.30}$$

$$0 \le y^{B,-}_{i_j} \le Q, \quad \forall i^{j} \in V_s \cup V_0. \tag{4.31}$$

4.2.2. A heuristic method for the loading/unloading quantity determination

The heuristic method intends to give feasible loading instructions of a route quickly. Given a newly generated route, the loading/unloading quantities at existing stops were first determined using the initial assignment step described in Section 4.2.2.1. If there were unbalanced stations after this assignment and the time budget allowed, those stations were appended to the end of the current route one by one, and the loading/unloading quantity was determined immediately after appending each of them. This step is called route extension and described in Section 4.2.2.2. Section 4.2.2.3 introduces a tailored step for the emission component of the objective function to reduce the vehicle load without affecting the penalty component.

4.2.2.1. Initial assignment. The first component of the objective function is the absolute deviation between the final and the target inventory levels at stations, and also the penalty caused by the broken bikes remained at stations. Therefore, this step aims to compute the maximum quantity that the vehicle can handle at each stop to satisfy the requirement at each station as much as possible.

The nodes are tackled in the order that they are visited. We first consider the stops at stations (i.e., not the depot). For the a-th stop of a route at station i, we have the remaining vehicle capacity rc_a, the vehicle load of usable bikes lg_a, the remaining time $\overline{T}_a$ for handling the bikes (already excluding the travel time for returning to the depot), and the remaining supply/demand at this station – rg_i for usable bikes and rb_i for broken bikes.

For a pickup station, the loading quantity of broken bikes at Stop a is first determined by

$$br_a = \min\left\{rc_a, \left\lfloor \overline{T}_a \Big/ 2L \right\rfloor, rb_i\right\} \tag{4.32}$$

Then the loading quantity of usable bikes is determined by

$$pd_a = \min\left\{rc_a - br_i, \left\lfloor \left(\overline{T}_a - 2L \cdot br_a\right) \Big/ 2L \right\rfloor, rg_i \right\} \tag{4.33}$$

Note that $2L$ is used here because we need to consider the unloading time of a bike before collecting it.

The resources left for the next stop (i.e., Stop $a+1$) at Station j are updated as follows:

$$rc_{a+1} = rc_a - br_a - pd_a, \tag{4.34}$$

$$lg_{a+1} = lg_a + pd_a, \text{ and} \tag{4.35}$$

$$\overline{T}_{a+1} = \overline{T}_a - 2L \cdot (br_a + pd_a) - t_{ij}. \tag{4.36}$$

Then the demand/supply of Station i is updated as follows:

$$rg_i \rightarrow rg_i - pd_a, \tag{4.37}$$

$$rb_i \rightarrow rb_i - br_a. \tag{4.38}$$

For a drop-off station, we unloading usable bikes first to make room for collecting broken bikes. The unloading quantity is determined as

$$pd_a = \min\{lg_a, rg_i\} \tag{4.39}$$

We do not consider the unloading time here because it has been included when collecting those bikes, see equation (4.36).

The remaining vehicle capacity at this stop is updated by

$$rc_a \rightarrow rc_a + pd_a. \tag{4.40}$$

If the quantity is bounded by the vehicle load lg_a, we can check whether the vehicle can collect more or drop fewer usable bikes when it departs from the last depot. We assume that the original loading instruction at the last depot is fd ($fd > 0$ for loading fd bikes, and $fd < 0$ for unloading $-fd$ bikes), and the minimum remaining vehicle capacity from the last depot to the current stop is vc. Then the increased loading amount or reduced unloading amount is determined by

$$ep = \min\left\{vc, rg_i - pd_a, \left\lfloor \overline{T}_a \Big/ 2L \right\rfloor - \min\{fd, 0\} \right\} \tag{4.41}$$

If $fd + ep > 0$, i.e., the final action at the last depot is loading, then the remaining time budget for the stops after Stop a should be updated by

$$\overline{T}_a = \overline{T}_a - 2L \cdot (\min\{fd, 0\} + ep). \tag{4.42}$$

Then the loading quantity of broken bikes is determined using equation (4.32). The resources left for the next stop (i.e., Stop $a+1$) at Station j can be updated as follows:

$$rc_{a+1} = rc_a - br_a, \tag{4.43}$$

$$lg_{a+1} = lg_a - pd_a, \text{ and} \tag{4.44}$$

$$\overline{T}_{a+1} = \overline{T}_a - 2L \cdot br_a - t_{ij}. \tag{4.45}$$

The information of Station i is updated as

$$rg_i \rightarrow rg_i + pd_a + ep \text{ and} \tag{4.46}$$

$$rb_i \rightarrow rb_i - br_a. \tag{4.47}$$

For a balanced station with neither demand or supply of usable bikes, the loading quantity of broken bikes can be determined using equation (4.32), and other information can be updated using equations (4.34)–(4.38) with $pd_a = 0$.

For those intermediate stops at the depot, all the bikes (i.e., both usable and broken bikes) on the vehicle are unloaded. Then for the next stop after the depot (i.e., Stop $a+1$) at Station j, the remaining vehicle capacity rc_{a+1} is updated as the full vehicle capacity, the load of usable bikes lg_{a+1} is updated as zero, and the remaining time $\overline{T}_{a+1}$ is updated as $\overline{T}_a - t_{ij}$.

If no station is reachable from the current stop within the time budget, then a depot is appended to the end of the current route and the initial assignment stops. Appending this depot is always feasible because, for every previously appended station, the time for returning to the depot and unloading all the bikes on the vehicle has been considered in advance.

4.2.2.2. Route extension. For a newly generated route (e.g., in the neighborhood search introduced in Section 4.3), it may happen that after computing the loading instructions for existing stops in the route, the remaining time still allows the vehicle to visit additional stops. In this case, the route can be extended by appending unbalanced stations. Candidate stations are considered in descending order of the additional time caused by appending them (visiting this station and returning to the depot). The computation of loading/unloading quantities follows the same rule as in Section 4.2.2.1.

We define the "current inventory level" as the present inventory level at a station after any step is performed. After the initial assignment and/or the route extension, the existing route has the following characteristics:

(1) The pickup stations have the current inventory level no less than the target inventory level (i.e., a non-negative deviation),
(2) The drop-off stations have the current inventory level no larger than the target inventory level (i.e., a non-positive deviation), and a negative deviation implies that the unloading quantity at this stop is bounded by the vehicle load (see equation (4.39), otherwise the deviation should be zero).

These two characteristics will support the argument in the following section.

4.2.2.3. Quantity adjustment. The emission function adopted in this study is related to both the travel distance and vehicle payload. For a given route, the travel distance is fixed and the initial assignment has assigned each stop an initial loading/unloading quantity. Then after previously described two steps, the quantity adjustment step here aims to reduce the vehicle load along the route without affecting the total absolute deviation reduction achieved in the previous steps.

We look into pairs of Stops m and n ($m < n$, and both at stations). The principle is to reduce the vehicle load between these two stops by loading fewer bikes or unloading more bikes at the former stop (m) and correspondingly loading more bikes or unloading fewer bikes at the latter stop (n). After the quantity adjustments at these two stops, the inventory level at Stop m increases and that at Stop n decreases.

We have the following two inferences based on the adjustment principle and the characteristics of the current inventory level of stations mentioned at the end of the last section:

(1) Stop m should have its current inventory level no less than the target inventory level (a non-negative deviation)
   (a) If Stop m is a drop-off station and its current inventory level is less than the target inventory level (has a negative deviation), the inventory level at this station cannot increase because the unloading quantity at this stop is bounded by the vehicle load (the second characteristic mentioned at the end of the last section). Therefore, Stop m cannot be a drop-off station with a negative deviation.
   (b) If Stop m is a pickup station, according to the first characteristic mentioned at the end of the last section, i.e., a pickup station does not have a negative deviation, Stop m cannot be a pickup station with a negative deviation.

We can conclude that Stop m cannot have a negative deviation.

(2) If Stop n is at a different station from Stop m, then Stop n should have its current inventory level larger than the target inventory level (a positive deviation)

If Stop n has a negative deviation or zero deviation, and its inventory level further decreases after the adjustment, then its absolute deviation increases. Since the inventory level at Stop m (a stop with a non-negative deviation) increases, its deviation also increases. Then the adjustments to the two stops lead to an increase in the total absolute deviation of all stations. Therefore, we can conclude that Stop n cannot have a non-positive deviation.

Remark: if Stop n is at the same station as Stop m, then Stop n has the same requirement as Stop m (i.e., a non-negative deviation).

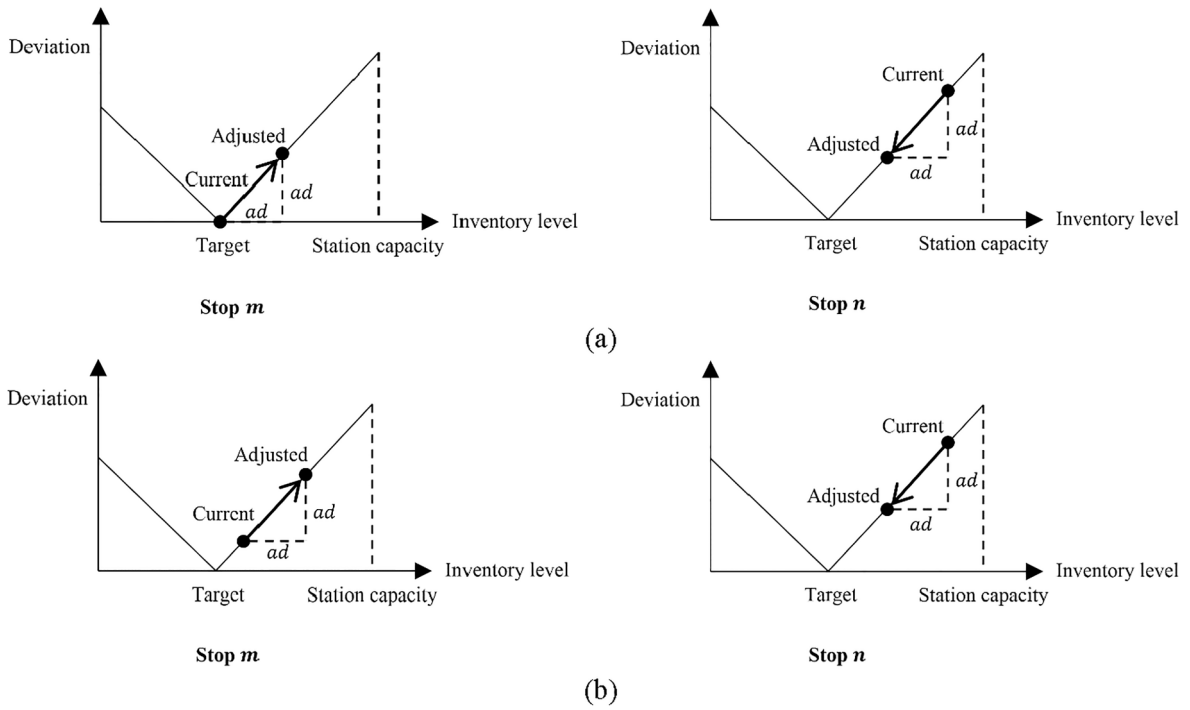

Fig. 4–3. The changes in the inventory level at a pair of different stations in Type I quantity adjustment: (a) Stop m has a zero deviation and Stop n has a positive deviation, (b) both stops have positive deviations.

With these inferences, we summarized two types of adjustments for the two stops:

Type I: Stop m and Stop n are two visits to different stations, and the station at Stop m has a non-negative deviation (i.e., the current inventory level at this station is not less than the target inventory level), whereas the station at Stop n has a positive deviation (i.e., the current inventory level is larger than the target inventory level),
Type II: Stop m and Stop n are two visits to the same station having a non-negative deviation from the target inventory level.

For Type I adjustment, we use Fig. 4-3 to show the changes in inventory levels at stops m and n at different stations. The absolute deviation is a piecewise linear function of the inventory level, which gives a zero deviation when the inventory level is equal to "Target". The "Current" point corresponds to the current inventory level and the absolute deviation of the corresponding station before the adjustment. For Stop m, the current inventory level can be either equal to the target inventory level (shown in Fig. 4-3(a)), or larger than the target inventory level (shown in Fig. 4-3(b)). For Stop n, the current inventory level should be strictly larger than the target inventory level. The "Adjusted" point corresponds to the inventory level and the absolute deviation after the adjustment. The arrows on the solid line indicate the directions of the change in the inventory level and the absolute deviation. As mentioned earlier, the inventory level at Stop m increases and that at Stop n decreases, and the adjusted quantities at two stations are identical (denoted as ad). We can see from Fig. 4-3 that the increasing absolute deviation at Stop m can be offset by the decreasing absolute deviation at Stop n.

The increasing/decreasing amount at two stops ad can be determined as follows. We denote that the minimum vehicle load of usable bikes between Stop m and Stop n before the adjustment is g_{min}, the original loading/unloading quantity is pd_m at Stop m and pd_n at Stop n, the remaining capacity of the station at the m-th stop is sc_m, the deviation at Stop n is Dev_n, and the remaining time budget is $\widehat{T}$.

Since Stop m should have a non-negative deviation and Stop n should have a positive deviation, we can have the following two combinations of Stop m and Stop n:

(a) Stop m is a pickup station and Stop n is another pickup station: the adjusted amount is $ad = \min\{g_{min}, pd_m, Dev_n\}$, i.e., the vehicle picks up ad fewer bikes at Stop m and picks up ad more bikes at Stop n.

(b) Stop m is a drop-off station and Stop n is a pickup station: the adjusted amount is $ad = \min\{g_{min}, sc_m, Dev_n, \left\lfloor \widehat{T}/2L \right\rfloor\}$, i.e., the vehicle drops off ad more bikes at Stop m and picks up ad more bikes at Stop n. The remaining time budget is updated as $\widehat{T} - 2L \cdot ad$.

Note that case (b) could make a drop-off station with zero deviation have a positive deviation after this adjustment (recall that after

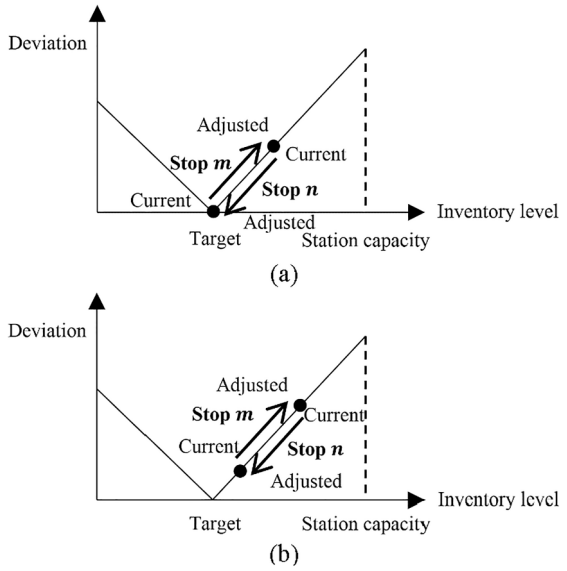

Fig. 4–4. The changes in the inventory level at a pair of stops at the same station in Type II quantity adjustment: (a) the station has a zero deviation, (b) the station has a positive deviation.

the initial assignment and/or the route extension, the drop-off stations only have non-positive deviations). That is to say, during the quantity adjustment step, pickup stations still have non-negative deviations, whereas drop-off stations do not have limits on their deviations. Therefore, two additional combinations are possible with Stop n in the previous two combinations being a drop-off station:

(c) Stop m is a pickup station and Stop n is a drop-off station: the adjusted amount is $ad = \min\{g_{min}, pd_m, pd_n, Dev_n\}$, i.e., the vehicle picks up ad fewer bikes at Stop m and drops off ad fewer bikes at Stop n. The remaining time budget is updated as $\widehat{T} + 2L \cdot ad$.
(d) Stop m is a drop-off station and Stop n is another drop-off station: the adjusted amount is $ad = \min\{g_{min}, sc_m, pd_n, Dev_n\}$, i.e., the vehicle drops off ad more bikes at Stop m and drops off ad fewer bikes at Stop n.

For Type II adjustment, Stop m and Stop n are two visits to the same station. The minimum vehicle load of usable bikes between Stop m and Stop n before the adjustment is denoted as g_{min}. We have the following two scenarios:

(a) If this station is a pickup station and the loading quantity at Stop m is pd_m, then the adjusted amount is $ad = \min\{g_{min}, pd_m\}$, i.e., the vehicle picks up ad fewer bikes at Stop m and picks up ad more bikes at Stop n.
(b) If this station is a drop-off station, the remaining station capacity after the m-th stop is sc_m, and the unloading quantity at Stop n is pd_n, then the adjusted amount is $ad = \min\{g_{min}, sc_m, pd_n\}$. i.e., the vehicle drops off ad more bikes at Stop m and drops off ad fewer bikes at Stop n.

We have the changes in the inventory levels as shown in Fig. 4-4. The increasing inventory level at Stop m can be offset by the decreasing inventory level at Stop n. Therefore, the inventory level of this station stays unchanged after the adjustment, but the vehicle load between Stop m and Stop n is reduced.

After both Type I and Type II adjustments, the total absolute deviation of the system stays unchanged, but the vehicle load between Stop m and Stop n is reduced by ad. This process repeats until all combinations of two stops at stations are checked.

In summary, the heuristic method for the computation of loading instructions consists of three steps: first, the quantity that can satisfy the requirement of existing stops the most is determined based on available resources, such as the vehicle capacity, carried usable bikes and so on; after that, the route can be extended if time allows; finally, the initial quantities are adjusted to reduce the vehicle load without changing the total absolute deviations of the system already achieved in the previous two steps.

4.3. Neighborhood search

The exploitation ability of the EABC algorithm is mainly driven by the employed bee phase and the onlooker bee phase. To enhance the search in the neighborhood, we used an adaptive local search in these two phases. This search involves using a set of different neighborhood operators to change the adjacency or/and the order of the stops in a route to generate a new route. Then, the loading instructions were determined using either method introduced in Section 4.2 and the objective value was calculated. These operators were applied in random order. Once an operator generated an improved solution, it repeated until no improvement was achieved. Note that for the scout bee phase in the EABC algorithm, instead of using a randomly generated solution, a neighbor solution of the old one

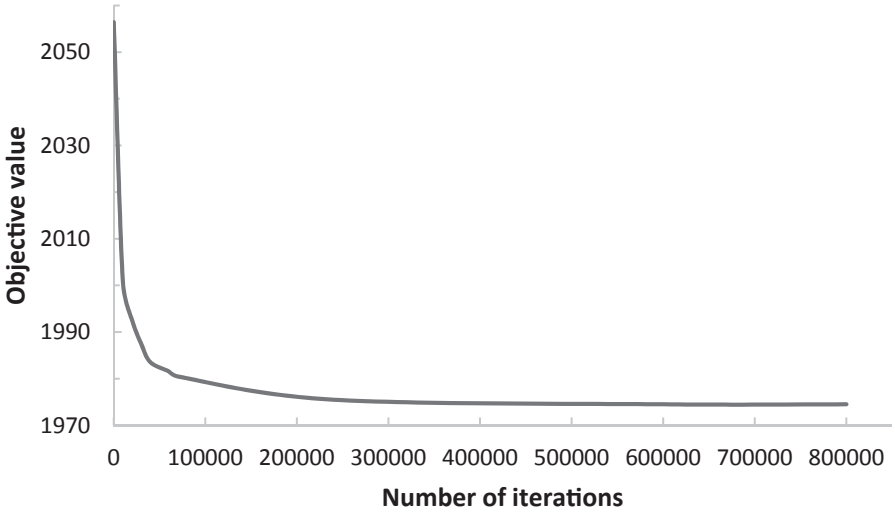

Fig. 5–1. Converging process of the objective value over the number of iterations.

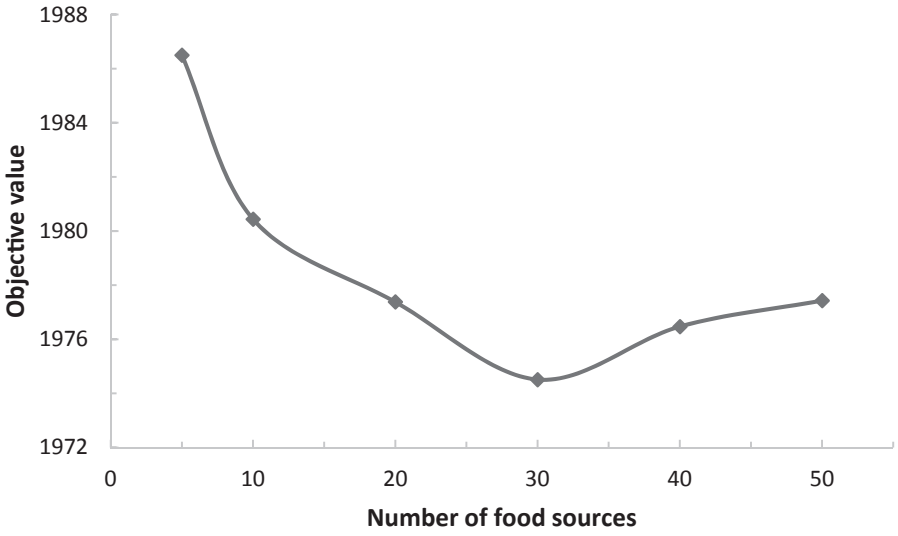

Fig. 5–2. Effect of the number of food sources.

was adopted to replace the solution without any improvement in the past few iterations. The scout bee used the same set of neighborhood operators as in the employed and onlooker bee phases.

There are six neighborhood search operators, namely single node relocation, single node swap, the relocation of a subsequence, the swap of two subsequences, node removal, and node insertion. For more details of the first four operators, the reader is referred to the study of Szeto et al. (2011).

5. Computational experiments

In this section, we present the experiments using the benchmark instances created by Rainer-Harbach et al. (2015) based on Citybike Vienna. For each instance, 10% of the bikes were randomly selected as broken bikes. The name of each instance was formatted as [S][Number of stations]_[Index of the instance]. The proposed method was coded with C++ in Visual Studio 2010 and executed on the high-performance computer with two 10-core Intel Xeon E5-2600 v3 (Haswell) processors and 96 GB physical memory. The parameters in the emission function follow the settings in the study of Wang and Szeto (2018). The penalty for a remaining broken bike at bike stations θ was set as 2 and the weighting factor of emissions was set as 0.1. The capacity of the repositioning vehicle (Q) was assumed to be 20 and the loading/unloading time for one bike was assumed to be 30 s. To reduce the number of parameters, the number of employed bees was set to equal the number of onlooker bees. Our preliminary experiments found that the enhanced ABC

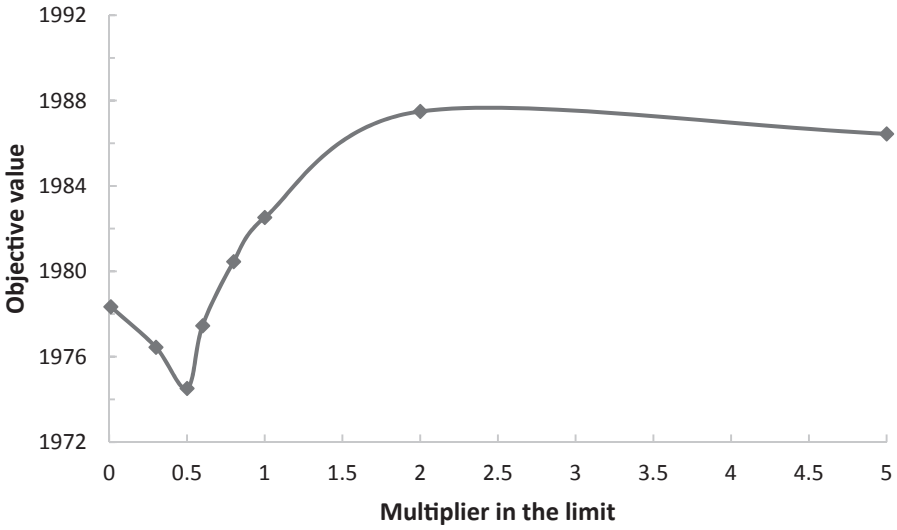

Fig. 5–3. Effect of the multiplier in the function of parameter *limit*.

algorithm performed better than the original version. Therefore, we implemented the former one (the EABC algorithm) in the following experiments. For every instance in the experiments, the program was run 20 times.

The remaining part of this section is organized as follows. Section 5.1 presents the results of parameter calibration. Section 5.2 discusses the effectiveness of the heuristic method and the network flow approach introduced in Section 4.2.2. Section 5.3 shows the efficiency of the proposed method by comparing the solution quality and computational time with CPLEX. Finally, in Section 5.4, we compare the results obtained by the proposed method with those by the benchmark method in Alvarez-Valdes et al. (2016) and also the canonical genetic algorithm (GA).

5.1. Parameter tuning

In the proposed algorithm, there are mainly three parameters to be tuned, i.e., the maximum number of iterations for termination, the number of food sources (also equal to the number of employed bees and the number of onlooker bees), and the value of *limit* that triggers the scout bee phase. We took the instances with 300 stations to tune those parameters in the stated order.

5.1.1. Termination condition

After running the algorithm pretty long, we find that the algorithm can converge before 300,000 iterations. The converging process is shown in Fig. 5-1. The 300,000 iterations were normalized by the network size (i.e., 300 stations). Therefore, 1000 times of the network size was used as the maximum number of iterations for the following experiments. For smaller networks, using this number in this termination condition is also sufficient for the algorithm to converge.

5.1.2. Number of food sources

With the other parameters fixed, experiments were conducted on the cases in which the numbers of food sources were 5, 10, 20, 30, 40, and 50. The trend of the objective value over the number of food sources is shown in Fig. 5-2. It can be seen that setting the number of food sources as 30 can generate the best objective value. Therefore, the number of food sources was set as 30 for the following experiments.

5.1.3. The multiplier in the limit

Following Simon (2013), we set the parameter *limit* based on the formula $m^*N^*|S|$, where N is the total number of employed and onlooker bees, and $|S|$ is the network size. The parameter m is a multiplier, which was set as 0.5 by Simon (2013). We tuned the value of m. Different values were tried from 0.01 to 5 and the trend of objective value is presented in Fig. 5-3. It can be found that $m = 0.5$ gives the best objective value. Therefore, in the following experiments, the value of *limit* was set as $0.5 \cdot (30+30) \cdot |S| = 30|S|$.

In summary, the algorithm converges when the number of iterations reaches $1000|S|$. The best numbers of employed bees and onlooker bees are both 30. If a food source has not been improved for $30|S|$ times, the scout bee phase should be triggered.

5.2. The effectiveness of using the heuristic method for the loading instruction determination

In Section 4.2 we introduced two approaches to obtain the loading/unloading quantities: solving a network flow model (the exact method) and a heuristic method. The objective of this experiment is to compare the results obtained from using either of them under the framework of the EABC algorithm to demonstrate the effectiveness of the latter.

Table 5-1
The results from using the exact method and the heuristic method for the loading instruction determination.

|S|	CPU/s	Instance	Exact loading		Heuristic loading		Improvement of mean value	p-value
			Mean	Best	Mean	Best		
60	96	S60_1	117.25	113.26	97.50	96.02	16.84%	9.73E−21
		S60_2	134.33	128.29	108.42	107.07	19.29%	1.25E−23
		S60_3	115.88	113.16	99.06	96.97	14.51%	3.23E−28
		S60_4	133.51	129.17	108.70	106.15	18.58%	1.48E−21
		S60_5	129.26	123.95	109.45	107.04	15.33%	2.55E−23
120	200	S120_1	518.63	513.78	489.04	485.59	5.70%	3.29E−24
		S120_2	583.35	576.18	555.50	550.94	4.77%	2.08E−26
		S120_3	517.19	510.87	488.90	484.62	5.47%	1.18E−23
		S120_4	616.52	610.65	588.17	584.32	4.60%	1.20E−25
		S120_5	607.04	600.63	577.12	574.38	4.93%	2.17E−22
180	310	S180_1	1044.89	1037.93	1015.55	1009.65	2.81%	1.01E−22
		S180_2	991.05	981.96	953.93	951.56	3.75%	2.51E−22
		S180_3	1058.77	1051.87	1024.64	1019.60	3.22%	1.72E−28
		S180_4	982.92	976.67	954.49	951.40	2.89%	3.51E−21
		S180_5	1022.28	1008.82	987.87	982.41	3.37%	6.82E−19
240	425	S240_1	1578.52	1568.56	1543.94	1541.17	2.19%	6.44E−25
		S240_2	1449.83	1443.87	1418.25	1414.64	2.18%	2.78E−22
		S240_3	1470.81	1461.82	1433.81	1429.34	2.52%	5.49E−25
		S240_4	1454.23	1446.95	1417.10	1413.5	2.55%	6.30E−29
		S240_5	1467.56	1461.00	1435.55	1432.5	2.18%	6.86E−22
300	590	S300_1	1925.42	1916.07	1888.71	1885.65	1.91%	9.44E−20
		S300_2	1983.46	1968.49	1958.40	1954.45	1.26%	6.52E−16
		S300_3	2084.50	2069.71	2052.72	2049.39	1.52%	5.46E−19
		S300_4	2040.64	2030.81	2006.53	2000.42	1.67%	5.86E−26
		S300_5	2007.71	1999.71	1979.60	1974.42	1.40%	3.09E−25

Table 5-2
Comparison of the results of CPLEX and the EABC algorithm.

Instance	CPLEX			The EABC algorithm			Improvement
	Best bound	Best integer	CPU/s	Mean	Best	CPU/s	
S10_1	1.27	1.27	415	1.27	1.27	7	0
S10_2	1.19	1.19	4134	1.19	1.19	7	0
S10_3	1.20	1.20	66	1.20	1.20	8	0
S10_4	1.22	1.22	219	1.22	1.22	8	0
S10_5	1.05	1.05	430	1.05	1.05	8	0
S20_1	0.07	2.30	7200	2.26	2.26	23	1.74%
S20_2	0.07	2.41	7200	2.32	2.32	23	3.73%
S20_3	0.07	2.51	7200	2.44	2.44	24	2.79%
S20_4	1.07	3.51	7200	2.41	2.41	24	31.34%
S20_5	0.08	2.40	7200	2.30	2.30	23	4.17%
S30_1	0.06	3.85	7200	3.55	3.55	46	7.79%
S30_2	0.07	3.94	7200	3.66	3.66	45	7.11%
S30_3	0.08	4.20	7200	3.78	3.78	44	10.00%
S30_4	0.07	3.30	7200	3.13	3.13	45	5.15%
S30_5	0.07	4.06	7200	3.82	3.82	46	5.91%

The tested instances were grouped according to their network size $|S|$, which varied from 60 to 300. Each group had five different instances. The computational time was controlled the same for a specific group to run the EABC algorithm with these two different approaches. The time budget for the repositioning work was set as $|T|$=6h. We ran the program of the EABC algorithm 20 times with the two approaches, and recorded the average and best objective values of 20 runs for each instance. We calculated the improvement on the average objective value by using the heuristic method compared with that by using the exact method, and also carried out the *t*-test to ensure that the difference in the mean objective value is statistically significant. The results are presented in Table 5-1. It can be concluded that, within the same computational time, using the proposed heuristic method to obtain the loading/unloading quantities can generate statistically better solutions than solving a network flow model.

5.3. Comparison of the results of CPLEX and the EABC algorithm

To show the efficiency of the proposed EABC algorithm together with the heuristic method for the loading/unloading quantity

Table 5-3
Comparison of the results of the sequential method, GA, and the EABC algorithm.

|S|	CPU/s	Instance	Sequential method		GA		EABC algorithm		EABC vs Sequential		EABC vs GA	
			Mean	Best	Mean	Best	Mean	Best	Improvement	p-value	Improvement	p-value
60	96	S60_1	188.72	184.50	104.82	98.21	97.50	96.02	48.33%	7.85E−39	6.98%	7.01E−09
		S60_2	204.87	197.43	118.01	111.04	108.42	107.07	47.08%	9.31E−37	8.12%	2.45E−08
		S60_3	202.39	198.53	103.67	100.94	99.06	96.97	51.05%	1.16E−47	4.44%	1.59E−09
		S60_4	214.43	210.21	116.56	112.08	108.70	106.15	49.31%	3.14E−46	6.74%	1.48E−11
		S60_5	202.10	198.47	116.49	111.86	109.45	107.04	45.84%	2.28E−51	6.04%	2.90E−09
120	200	S120_1	635.14	626.75	503.50	490.57	489.04	485.59	23.00%	1.15E−38	2.87%	5.98E−09
		S120_2	689.50	683.80	571.76	564.01	555.50	550.94	19.43%	7.27E−55	2.84%	8.50E−13
		S120_3	645.08	638.86	503.82	497.70	488.90	484.62	24.21%	1.51E−46	2.96%	7.81E−16
		S120_4	723.77	715.39	600.94	593.56	588.17	584.32	18.74%	2.54E−35	2.13%	3.59E−13
		S120_5	715.18	703.27	594.38	581.51	577.12	574.38	19.30%	7.06E−35	2.90%	3.00E−11
180	310	S180_1	1168.76	1160.89	1035.03	1024.84	1015.55	1009.65	13.11%	2.90E−39	1.88%	7.00E−13
		S180_2	1109.86	1095.7	978.28	967.88	953.93	951.56	14.05%	7.09E−35	2.49%	4.12E−13
		S180_3	1176.14	1166.47	1047.19	1037.97	1024.64	1019.60	12.88%	1.31E−53	2.15%	2.02E−15
		S180_4	1117.59	1109.62	969.54	953.50	954.49	951.40	14.59%	3.01E−46	1.55%	6.99E−10
		S180_5	1142.73	1130.55	1012.35	999.64	987.87	982.41	13.55%	8.37E−36	2.42%	3.94E−11
240	425	S240_1	1710.88	1701.41	1570.87	1558.44	1543.94	1541.17	9.76%	3.39E−40	1.71%	1.06E−12
		S240_2	1574.90	1561.54	1441.84	1427.72	1418.25	1414.64	9.95%	1.52E−35	1.64%	1.11E−10
		S240_3	1596.50	1590.56	1466.23	1453.69	1433.81	1429.34	10.19%	3.16E−55	2.21%	1.34E−16
		S240_4	1574.84	1564.55	1444.55	1435.88	1417.10	1413.5	10.02%	2.16E−42	1.90%	5.45E−17
		S240_5	1593.04	1587.46	1460.87	1453.96	1435.55	1432.5	9.89%	6.60E−53	1.73%	6.98E−21
300	590	S300_1	2037.20	2027.61	1917.17	1904.84	1888.71	1885.65	7.29%	4.67E−37	1.48%	9.40E−16
		S300_2	2132.52	2127.28	1980.17	1970.59	1958.40	1954.45	8.16%	4.32E−55	1.10%	3.45E−14
		S300_3	2212.97	2206.41	2082.63	2067.65	2052.72	2049.39	7.24%	3.97E−56	1.44%	1.87E−16
		S300_4	2171.41	2167.63	2035.75	2025.7	2006.53	2000.42	7.59%	2.59E−57	1.44%	8.34E−19
		S300_5	2134.36	2121.6	2002.03	1994.73	1979.60	1974.42	7.25%	4.11E−34	1.12%	2.25E−16

determination, we compared the results of this solution approach with those obtained by ILOG CPLEX 12.6 invoked by a C++ program. The results of CPLEX and the EABC algorithm are summarized in Table 5-2.

We can find that for instances having 10 stations, CPLEX can converge to the optimal solution within the commonly used time limit of 7200 s. The EABC algorithm can also find the optimal solutions in all 20 runs for every instance using much shorter computational time. For instances having 20 and 30 stations, CPLEX cannot converge in a time limit of 7200 s and only feasible integer solutions are provided. However, the EABC algorithm can provide better feasible solutions within a much shorter time than CPLEX.

5.4. Comparisons of the results of the EABC algorithm with benchmark methodologies

This section presents the comparisons of the EABC algorithm (with the heuristic method presented in Section 4.2.2 for the loading instruction determination) with two benchmark methodologies to solve the proposed problem. The first method was developed by Alvarez-Valdes et al. (2016). It is a sequential approach (we call it the "sequential method" in the following context), where a minimum cost flow problem is first solved to pair the nodes in the network, i.e., the flows of bikes between two nodes, then a construction phase (an insertion algorithm) and an improvement phase are incorporated into a multi-start framework. The second method is the genetic algorithm (GA). GA is a meta-heuristic in the class of evolutionary computational algorithms, which mimics the evolution process of creatures in nature on the basis of the survival of the fittest. There are mainly two operators for creating the offspring, namely crossover and mutation. The crossover operator recombines parts of two selected solutions (parents) to create a new solution (offspring). The mutation operator changes some parts of a solution. For the GA, the simple ordered crossover (OX) (see Prins (2004) for instance) was used. The mutation step was designed to be the same as the scout bee phase in the EABC algorithm. The probability that a selected solution undergoes the OX was tuned as 0.7, and the probability that it undergoes mutation was tuned as 0.9.

In this section, the performance of these two methods together with the proposed EABC algorithm are compared under the same computational time. We looked into 25 instances with the network size varying from 60 to 300. For each size, five different networks were selected. The time budget for the repositioning work was set as 6 h. Each algorithm was run 20 times for every instance and we recorded the average and best objective values in 20 runs for each instance.

Table 5-3 shows the results of the sequential method, GA, and the EABC algorithm. We use the improvement of the average objective value produced by the EABC algorithm against the other two methods in percentage, and also the p-values of the *t*-test to show the performance of the proposed method. We can see that for all 25 instances, the EABC algorithm outperforms both the sequential method and the GA, and that all the p-values are far smaller than 0.01. We can, therefore, conclude that the improvement of the EABC algorithm on the objective value is statistically significant.

6. Conclusions

In this study, we used the enhanced artificial bee colony algorithm incorporated with an adaptive local search to solve the green bike repositioning problem with broken bikes. The objective is to minimize the weighted sum of the absolute deviation from the target inventory level, the penalty caused by broken bikes at stations, and the CO_2 emissions of the repositioning vehicle are minimized. The EABC algorithm was used to generate the vehicle route and a local search using a set of neighborhood operators was incorporated to improve solution quality. Two methods were proposed to obtain the loading/unloading quantities of two types of bikes at each visited station along a route, including a linear programming approach and a heuristic method.

To show the accuracy and efficiency of the proposed methodology, computational experiments were conducted on real-world instances with the size varying from 10 to 300 stations. The results show that the proposed method can produce optimal solutions for instances with 10 stations. For all the tested instances with 20 and 30 stations, the proposed method that relies on the heuristic loading method can provide better feasible solutions within a much shorter time than CPLEX. For larger instances with 60 to 300 stations, the proposed method outperforms the methodology by Alvarez-Valdes et al. (2016) and the canonical genetic algorithm.

Although the proposed method only focuses on the single vehicle case, it serves as the cornerstone for the multiple-vehicle case. One option to handle the multiple-vehicle case is to divide the stations into several groups and assign one vehicle to each. Another alternative is to design the routes of all the vehicles simultaneously. It may need more complicated techniques to alter the sequences and synchronize the inventory level of stations across multiple routes. This is left for future studies.

CRediT authorship contribution statement

Yue Wang: Conceptualization, Methodology, Software, Investigation, Visualization, Writing - original draft, Writing - review & editing. **W.Y. Szeto:** Conceptualization, Writing - review & editing, Supervision, Funding acquisition, Project administration.

Acknowledgments

This research was jointly supported by a grant from the National Natural Science Foundation of China (No. 71771194) and the Research Grants Council of the Hong Kong Special Administrative Region of China (17200119). The computational experiments were

performed using research computing facilities offered by Information Technology Services, the University of Hong Kong. We are grateful to the two reviewers for their constructive comments. We would also like to thank Dr. Chin-Sum Shui from National Chiao Tung University for providing the data on broken bikes used in this study.

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.
Angeloudis, P., Hu, J., Bell, M.G., 2014. A strategic repositioning algorithm for bicycle-sharing schemes. Transportmetrica A: Trans. Sci. 10 (8), 759–774.
Benchimol, M., Benchimol, P., Chappert, B., De La Taille, A., Laroche, F., Meunier, F., Robinet, L., 2011. Balancing the stations of a self service "Bike hire" system. RAIRO – Oper. Res. 45 (1), 37–61.
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.
Chang, S., Song, R., He, S., Qiu, G., 2018. Innovative bike-sharing in China: Solving faulty bike-sharing recycling problem. J. Adv. Transport. 4941029.
Chemla, D., Meunier, F., Wolfler Calvo, R., 2013. Bike sharing systems: Solving the static rebalancing problem. Discrete Optimiz. 10 (2), 120–146.
Contardo, C., Morency, C., Rousseau, L.-M., 2012. Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, Montreal, Canada: CIRRELT.
Cruz, F., Subramanian, A., Bruck, B.P., Iori, M., 2017. A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. Comput. Oper. Res. 79, 19–33.
Dell'Amico, M., Hadjicostantinou, E., Iori, M., Novellani, S., 2014. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. Omega 45, 7–19.
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.
Demir, E., Bektaş, T., Laporte, G., 2011. A comparative analysis of several vehicle emission models for road freight transportation. Transport. Res. Part D: Trans. Environ. 16 (5), 347–357.
Di Gaspero, L., Rendl, A., Urli, T., 2013. A hybrid ACO+ CP for balancing bicycle sharing systems. In: International Workshop on Hybrid Metaheuristics, 2013. Springer Berlin Heidelberg, 198–212.
Di Gaspero, L., Rendl, A., Urli, T., 2016. Balancing bike sharing systems with constraint programming. Constraints 21 (2), 318–348.
Erdoğan, G., Battarra, M., Wolfler Calvo, R., 2015. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. Eur. J. Oper. Res. 245 (3), 667–679.
Erdoğan, G., Laporte, G., Wolfler Calvo, R., 2014. The static bicycle relocation problem with demand intervals. Eur. J. Oper. Res. 238 (2), 451–457.
Ghosh, S., Varakantham, P., Adulyasak, Y., Jaillet, P., 2017. Dynamic repositioning to reduce lost demand in bike sharing systems. J. Artificial Intell. Res. 58, 387–430.
Hemamalini, S., Simon, S.P., 2010. Artificial bee colony algorithm for economic load dispatch problem with non-smooth cost functions. Electr. Power Compon. Syst. 38 (7), 786–803.
Ho, S.C., Szeto, W.Y., 2014. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. Transport. Res. Part E: Logist. Transport. Rev. 69, 180–198.
Ho, S.C., Szeto, W.Y., 2016. GRASP with path relinking for the selective pickup and delivery problem. Expert Syst. Appl. 51, 14–25.
Ho, S.C., Szeto, W.Y., 2017. A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem. Transport. Res. Part B: Methodol. 95, 340–363.
Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department.
Karaboga, D., Gorkemli, B., 2011. A combinatorial artificial bee colony algorithm for traveling salesman problem. In: 2011 International Symposium on Innovations in Intelligent Systems and Applications, 2011. IEEE, 50–53.
Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N., 2014. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. 42 (1), 21–57.
Kaspi, M., Raviv, T., Tzur, M., 2017. Bike-sharing systems: user dissatisfaction in the presence of unusable bicycles. IISE Trans. 49 (2), 144–158.
Kloimüllner, C., Papazek, P., Hu, B., Raidl, G.R., 2014. Balancing bicycle sharing systems: an approach for the dynamic case. In: European Conference on Evolutionary Computation in Combinatorial Optimization, 2014. Springer, 73–84.
Kopfer, H.W., Schönberger, J., Kopfer, H., 2014. Reducing greenhouse gas emissions of a heterogeneous vehicle fleet. Flexible Services Manuf. J. 26 (1–2), 221–248.
Li, J.-Q., Xie, S.-X., Pan, Q.-K., Wang, S., 2011. A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. Int. J. Comput. Commun. Control 6 (2), 286–296.
Li, Y., Szeto, W.Y., Long, J., Shui, C.S., 2016. A multiple type bike repositioning problem. Transp. Res. Part B 90, 263–278.
Lin, C., Choy, K.L., Ho, G.T.S., Chung, S.H., Lam, H.Y., 2014. Survey of green vehicle routing problem: past and future trends. Expert Syst. Appl. 41 (4), 1118–1138.
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. Transport. Res. Part C: Emerg. Technol. 92, 208–242.
Papazek, P., Kloimüllner, C., Hu, B., Raidl, G.R., 2014. Balancing bicycle sharing systems: an analysis of path relinking and recombination within a GRASP hybrid. In: International Conference on Parallel Problem Solving from Nature, 2014. Springer, Berlin, Heidelberg, 792–801.
Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. 31 (12), 1985–2002.
Raidl, G.R., Hu, B., Rainer-Harbach, M., Papazek, P., 2013. Balancing bicycle sharing systems: Improving a VNS by efficiently determining optimal loading operations. In: International Workshop on Hybrid Metaheuristics, 2013. Springer, Berlin, Heidelberg, 130–143.
Rainer-Harbach, M., Papazek, P., Hu, B., Raidl, G.R., 2013. Balancing bicycle sharing systems: A variable neighborhood search approach. In: European Conference on Evolutionary Computation in Combinatorial Optimization, 2013. Springer, Berlin, Heidelberg, 121–132.
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. Global Optim. 63 (3), 597–629.
Raviv, T., Tzur, M., Forma, I.A., 2013. Static repositioning in a bike-sharing system: Models and solution approaches. EURO J. Transport. Logist. 2 (3), 187–229.
Seattle Department of Transportation, 2019. Retrieved from https://www.seattle.gov/Documents/Departments/SDOT/BikeProgram/2019Q3_BikeShare_Summary_Report.pdf.
Shui, C.S., Szeto, W.Y., 2018. Dynamic green bike repositioning problem – A hybrid rolling horizon artificial bee colony algorithm approach. Transport. Res. Part D: Trans. Environ. 60, 119–136.
Simon, D., 2013. Evolutionary Optimization Algorithms. John Wiley & Sons.
Szeto, W.Y., Liu, Y., Ho, S.C., 2016. Chemical reaction optimization for solving a static bike repositioning problem. Transport. Res. Part D: Trans. Environ. 47, 104–135.
Szeto, W.Y., Shui, C.S., 2018. Exact loading and unloading strategies for the static multi-vehicle bike repositioning problem. Transport. Res. Part B: Methodol. 109, 176–211.
Szeto, W.Y., Wu, Y., Ho, S.C., 2011. An artificial bee colony algorithm for the capacitated vehicle routing problem. Eur. J. Oper. Res. 215 (1), 126–135.
Ting, C.-K., Liao, X.-L., 2013. The selective pickup and delivery problem: formulation and a memetic algorithm. Int. J. Prod. Econ. 141 (1), 199–211.
Vogel, P., Neumann Saavedra, B.A., Mattfeld, D.C., 2014. A hybrid metaheuristic to solve the resource allocation problem in bike sharing systems. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8457, pp. 16-29.
Wang, Y., Szeto, W.Y., 2018. Static green repositioning in bike sharing systems with broken bikes. Transport. Res. Part D: Trans. Environ. 65, 438–457.

Xiao, Y., Zhao, Q., Kaku, I., Xu, Y., 2012. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. Comput. Oper. Res. 39 (7), 1419–1431.
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. Transport. Res. Part B: Methodol. 103, 188–207.
Zhang, S., Lee, C.K.M., Choy, K.L., Ho, W., Ip, W.H., 2014. Design and development of a hybrid artificial bee colony algorithm for the environmental vehicle routing problem. Transport. Res. Part D: Trans. Environ. 31, 85–99.
Zhang, S., Xiang, G., Huang, Z., 2018. Bike-sharing static rebalancing by considering the collection of bicycles in need of repair. J. Adv. Transport. 8086378.