Solving a static repositioning problem in bike-sharing systems using iterated tabu search — Source link

Sin C. Ho, Wai Yuen Szeto

Institutions: Aarhus University, University of Hong Kong

Published on: 01 Sep 2014 - Transportation Research Part E-logistics and Transportation Review (Pergamon)

Topics: Tabu search and Heuristic (computer science)

Solving a static repositioning problem in bike-sharing systems using iterated tabu search

Sin C. Ho[a] and W. Y. Szeto[b]

[a]Department of Economics and Business
Aarhus University, Denmark
Email: sinch@asb.dk

[b]Department of Civil Engineering
The University of Hong Kong, Hong Kong
Email:ceszeto@hku.hk

Highlights

- We study the static bike repositioning problem
- We modify the problem to improve its realism and reduce the solution space
- We solve the problem by iterated tabu search with specific operators
- We obtain high quality solutions efficiently

Solving a static repositioning problem in bike-sharing systems using iterated tabu search

Abstract

In this paper, we study the static bike repositioning problem where the problem consists of selecting a subset of stations to visit, sequencing them, and determining the pick-up/drop-off quantities (associated with each of the visited stations) under the various operational constraints. The objective is to minimize the total penalties incurred at all the stations. We present an iterated tabu search heuristic to solve the described problem. Experimental results show that this simple heuristic can generate high quality solutions using small computing times.

Keywords: Bike-sharing; Static repositioning; Tabu search

1 Introduction

Bike sharing is very popular in many places nowadays. In April 2013, there were 535 schemes worldwide, in 49 countries (Larsen, 2013). It is therefore important to research bike sharing. Currently, bike sharing research topics include bike network design (dell'Olio et al., 2011; Lin and Yang, 2011; Martinez et al., 2012; Romero et al., 2012; Sayarshad et al., 2012; Lin et al., 2013; Lu, 2013), bike demand analysis (Froehlich et al., 2008; Borgnat et al., 2009; Kaltenbrunner et al., 2010; Borgnat et al., 2011; Vogel et al., 2011; Lathia et al., 2012), bike network flow analysis (Shaheen et al., 2011; Kitthamkesorn et al., 2013; Shu et al., 2013), bike service level analysis (Nair et al., 2013; Raviv and Kolka, 2013), and many others. As we can see, the bike sharing research has only received attention recently.

One fundamental problem of bike-sharing is that the numbers of bikes required at some stations are not enough to satisfy the bike user demand. Hence, in practice, trucks are deployed to transport bikes from surplus stations or depots to deficit stations to meet the demand. This problem is called a bike repositioning problem, which determines the optimal vehicle (truck) routes and the number of bikes loaded and unloaded at each station to meet the objective, such as satisfying bike user demand, subject to various constraints including routing, vehicle, station, and operational constraints. Very often, it is not necessary for the trucks to visit all bike stations. This means that the problem involves selecting and determining the sequence of the visited stations. This problem is more complicated than the classical vehicle routing problem (VRP) (Ho and Gendreau, 2006; Szeto et al., 2011) and the classical traveling salesman problem (TSP) (Gendreau et al., 1992) because the repositioning problem further requires determining the pick-up and drop-off quantities. These extra variables imply that the solution space is larger than that of the classical VRP and TSP. The repositioning problem was also referred to as a bike sharing rebalancing problem (BRP) (Dell'Amico et al., 2014) and belongs to the class of many-to-many pickup and delivery problems (Parragh et al., 2008). When the BRP only considers only one vehicle, the problem becomes one-commodity pickup-and-delivery TSP (1-PDTSP) introduced by Hernández-Pérez and Salazar-González (2004a). When the BRP considers multiple vehicles and a maximum duration constraint for each route, the problem belongs to one-commodity pickup-and-delivery VRP introduced by Shi et al. (2009).

Although bike sharing is very popular and the repositioning problem is a practical one, the literature on the problem is rather sparse (around 15 papers) compared with those of the classical VRP and TSP. In the literature, various bike repositioning problems with different constraints and objective functions have been studied. The problems can be broadly classified into two types, namely static and dynamic repositioning problems. The static repositioning problem considers the night time repositioning operation in which the numbers of bikes required by and presence in each station are fixed and known before repositioning takes place. The dynamic repositioning problem considers the case in which the number of bikes required by and presence in each station are changing over time. This problem is always found in the daytime repositioning operation or the 24-hour bike-sharing operation. To the best of our knowledge, very few (Nair and Miller-Hooks, 2011; Caggiani and Ottomanelli, 2012; Contardo et al., 2012) studied the dynamic problem and most of the existing studies are related to the static bike repositioning problem.

In the repositioning problem, the objectives considered include minimizing the total unmet demand or total user dissatisfaction (Contardo et al., 2012), minimizing the sum of relocation and lost user costs (Caggiani and Ottomanelli, 2012), minimizing the total travel cost (Benchimol et al., 2011; Chemla et al., 2013), minimizing the weighted sum of the deviation from the targeted number of bikes in each station, the numbers of loading and unloading quantities, and the total travel time on all routes (Papazek et al., 2013; Di Gaspero et al., 2013), minimizing the maximum tour length (Schuijbroek et al., 2013), and minimizing the sum of travel and handling costs (Erdoğan et al., 2013). The choice of objectives should be determined by the application of bike-sharing operations. Very often, the concern of the operator governs the choice of the objective. If the bike sharing system is operated by the government, the societal benefit measure, such as user satisfaction, the maximum deviation, and the penalty cost, should be included in the objective function. However, if the bike sharing system is operated by a private operator, the travel distance or travel time is the key concern and the cost minimizing objective is considered when determining the truck route. Nevertheless, we can observe that some objectives are more general than the others. For example, minimizing the total penalty cost is more general than minimizing the total user dissatisfaction and minimizing the sum of the deviation from the targeted number of bikes in each station, because we can choose a penalty function that assigns a value of zero to the level equal to or greater than the demand level and a very large number to other levels to replicate the effect of minimizing the total user dissatisfaction, and we can select a penalty function that assigns a value to a level equal to the absolute difference between that level and the target level to replicate the effect of minimizing the sum of deviations.

Various operational constraints are also considered in the repositioning problem. For example, despite the maximum time for each repositioning activity, Benchimol et al. (2011) proposed a routing constraint that requires each node being visited exactly once by the vehicle as in the traveling salesman problem. In terms of demand-related constraints, Benchimol et al. (2011) and Chemla et al. (2013) set the perfect balance requirement as a hard constraint but relaxed the time constraint. Lin and Chou (2012) considered road conditions, traffic regulations, and geographical factors in the constraint. Nair and Miller-Hooks (2011) introduced a probabilistic level-of-service constraint such that the repositioning activity must satisfy a certain proportion of the most nearest future demand but ignored the routing constraint. The operational constraints included in the problem should be related to the application. They also determine the complexity of the problem.

Most of the preceding bike sharing studies do not predefine a station to be 1) a pick-up one, 2) a drop-off one, or 3) a neither pick-up nor drop-off station. In fact, a station can be easily classified into one of these three types. We only need to handle the demand at drop-off stations. It is not necessary for a truck to visit all pick-up stations to satisfy the demand of each drop-off station or minimize the total penalty cost. Hence, the problem can be refined so that a more efficient solution method can be proposed. Based on this approach, Ting and Liao (2013) modeled the bike repositioning problem as the selective pick-up and drop-off problem. The input of the bike repositioning problem includes the sets of pick-up and drop-off stations. The truck must visit all the drop-off stations to satisfy the demand but it is unnecessary for the truck to visit all the pick-up stations. However, they considered the total travel time of the vehicle in the objective function and the number of bikes required by each drop-off station is explicitly given. In reality, not all the demand can be satisfied because the supply of bikes from depots and pick-up stations may be insufficient or the operational period is too short to reposition bikes to some of the drop-off stations. In these two cases, it is unwise to ensure the truck to visit all drop-off stations and satisfy all the demand. Moreover, the penalty cost of not satisfying one unit of demand at a station may also vary from one to another. In an area with high (low) bike station density or many (few) transport mode alternatives, the penalty cost may be low (high) because the users can (cannot) easily walk to other nearby bike stations or take other transport modes. It is more reasonable to consider such penalty cost than just user demand for the bike service provided by the government.

To address this issue, we propose a refined problem to consider such penalty cost. We use the concept of the penalty function proposed by Raviv et al. (2013) and consider a fixed planning horizon. The objective is to minimize the total penalty cost. The constraints in Raviv et al.

(2013) are modified to consider the sets of pick-up and drop-off stations individually and to reduce the solution space. As pointed out by Ting and Liao (2013), the original problem is already $\mathcal{NP}$-hard. The refined problem with a more complicated objective function is also $\mathcal{NP}$-hard. It is intractable to use exact methods such as those shown in Table 1 to solve large, realistic repositioning problems. Moreover, because the existing heuristics or approximation methods shown in Table 1 are tailored for solving their bike repositioning problems that are different from ours (in terms of objective function or constraints), their methods cannot be directly applied to solve our problem. Hence, we develop a heuristic that makes use of the problem's properties to solve our problem.

Table 1: A summary on the existing solution methods for bike repositioning problems

Algorithm	**Reference**
Exact algorithms:	
Dantzig-Wolfe and Benders decomposition	Contardo et al. (2012)
Benders decomposition	Erdoğan et al. (2013)
Approximation method:	
9.5-approximation algorithm	Benchimol et al. (2011)
Heuristics:	
Variable neighborhood search	Papazek et al. (2013)
	Rainer-Harbach et al. (2013)
Cluster-first route-second	Schuijbroek et al. (2013)
Ant colony and constraint programming	Di Gaspero et al. (2013)
PILOT/GRASP	Papazek et al. (2013)
Hybrid algorithm:	
Branch-and-cut method with tabu search	Chemla et al. (2013)

As the proposed problem involves a routing problem, we considered to enhance an existing heuristic for routing problems to solve our proposed problem. Because tabu search is well-known to be very efficient to solve routing problems compared with other non-hybrid methods and tabu search has rarely been used to solve the static repositioning problem as shown in Table 1, we choose tabu search as the backbone of our solution method. However, the repositioning problem involves pick-up and drop-off quantity variables in addition to routing variables. Hence, we could not directly apply tabu search for routing problems to solve our proposed problem and some new operators must be added to tabu search to handle the extra variables. For this purpose, we develop specific operators to ensure solution feasibility. In order to accelerate the search when evaluating a neighbor solution, several ideas are incorporated into the heuristic in order to avoid re-computing all the pick-up/drop-off quantities. To improve the solution quality, the tabu search procedure is embedded into an iterative framework, where several intensification/diversification mechanisms are applied to the solution obtained from tabu search. To show the efficiency and accuracy of our method, we set up different test scenarios and compare the results obtained from IBM Ilog CPLEX.

The remainder of the paper is organized as follows. Section 2 presents the formulation. Section 3 depicts the solution method. Section 4 describes the test cases and discusses the results. Section 5 gives the conclusion.

2 Mathematical formulation

In this section, we develop the formulation based on Raviv et al. (2013), but modify their formulation, consider single vehicle only, and explicitly define pick-up and drop-off stations similar to the studies by Caggiani and Ottomanelli (2012) and Ting and Liao (2013). However, unlike their studies, pick-up and drop-off stations are deduced from their penalty functions. A station is defined as a drop-off station if the initial number of bikes at the station is smaller than the optimal number of bikes at that station (i.e., the level at which the penalty cost is minimum). On the other hand, if the initial number of bikes at a station is larger than the optimal number of bikes at that station, then the station is defined as a pick-up station.

The formulation of Raviv et al. (2013) was modified by performing the following sequentially: 1.) Split each constraint into a pair of constraints, one for pick-up stations and one for drop-off stations. 2.) Simplify each resultant constraint according to the definition of the associated station. 3.) Refine the meanings of loading and unloading variables according to the definition of stations. 4.) Simplify the whole formulation for the single vehicle case by eliminating redundant notations such as the notations' subscript for vehicles and the notation for the set of vehicles, and 5.) Simplify the objective function to only consider the sum of the penalty cost of each station and define a specific form of the penalty function for the objective function. Note that if we removed step 4 in the preceding procedure, we could actually have a multi-vehicle formulation for the static bike repositioning problem.

The modified formulation assumes that the depot is both a pick-up and drop-off node and has sufficient bikes and capacity. The formulation also assumes that each station can only be visited by the vehicle at most once as in the literature (e.g., Raviv et al., 2013) because this assumption can make the solution space smaller, leading to the development of an efficient heuristic for solving large-scale bike repositioning problems much easier. Moreover, the formulation allows the vehicle leaving the depot with some bikes loaded on board and returning to the depot non-empty. Furthermore, the formulation allows the vehicle visiting the depot multiple times if necessary. To ease the description of the mathematical formulation of the static bike repositioning problem, we start by introducing the notations.

$\mathcal{N}$	Set of stations, indexed by $i = 1, \dots, \lvert\mathcal{N}\rvert$
$\mathcal{N}_0$	Set of nodes, including the stations and the depot (denoted by $i = 0$)
c_i	Number of lockers installed at station $i \in \mathcal{N}_0$, referred to as the station's capacity
$f_i(s_i)$	A convex penalty function for station $i \in \mathcal{N}$; the function is defined over the integers $s_i = 0, \dots, c_i$
s_i^0	Number of bikes at node i before the repositioning operation starts
s_i^I	$\arg\min_{s_i} f_i(s_i)$, i.e., optimal number of bikes at node i
$\mathcal{P}$	Set of pick-up stations, i.e., $\{i \in \mathcal{N} \mid s_i^0 > s_i^I\}$
$\mathcal{D}$	Set of drop-off stations, i.e., $\{i \in \mathcal{N} \mid s_i^0 < s_i^I\}$
k	Vehicle capacity
t_{ij}	Traveling time from station i to station j
T	Repositioning time, i.e., time allotted to the repositioning operation
L	Time required to remove a bike from a station and load it onto the vehicle
U	Time required to unload a bike from the vehicle and hook it to a locker in a station
a_i,b_i	Parameters associated with the penalty function for station i
M	A very large number

Decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if the vehicle travels directly from node } i \text{ to node } j; \\ 0, & \text{otherwise.} \end{cases}$$

y_{ij}	Number of bikes carried on the vehicle when it travels directly from node i to node j. y_{ij} is zero if $x_{ij} = 0$.
s_i	Inventory level at station i at the end of the repositioning operation
y_i^D	Number of bikes dropped off at node $i \in \mathcal{D} \cup \{0\}$ by the vehicle
y_i^P	Number of bikes picked up at node $i \in \mathcal{P} \cup \{0\}$ by the vehicle
q_i	Auxiliary variable associated with node i used for the sub-tour elimination constraints

The problem can be stated mathematically as:

$$\min \sum_{i\in\mathcal{N}} f_i(s_i) = \sum_{i\in\mathcal{N}} (a_i(s_i - s_i^I)^2 + b_i) \tag{1}$$

$$\text{s. t. } s_i = s_i^0 - y_i^P \qquad \forall i \in \mathcal{P} \tag{2}$$

$$s_i = s_i^0 + y_i^D \qquad \forall i \in \mathcal{D} \tag{3}$$

$$s_0 = s_0^0 - y_0^P + y_0^D \tag{4}$$

$$y_i^P = \sum_{j\in\mathcal{N}_0, j\neq i} y_{ij} - \sum_{j\in\mathcal{N}_0, j\neq i} y_{ji} \qquad \forall i \in \mathcal{P} \tag{5}$$

$$y_i^D = \sum_{j\in\mathcal{N}_0, j\neq i} y_{ji} - \sum_{j\in\mathcal{N}_0, j\neq i} y_{ij} \qquad \forall i \in \mathcal{D} \tag{6}$$

$$y_0^P = \sum_{j\in\mathcal{N}} y_{0j} \tag{7}$$

$$y_0^D = \sum_{j\in\mathcal{N}} y_{j0} \tag{8}$$

$$y_{ij} \le k x_{ij} \qquad \forall i, j \in \mathcal{N}_0, i \neq j \tag{9}$$

$$\sum_{j\in\mathcal{N}} x_{0j} \ge 1 \tag{10}$$

$$\sum_{j\in\mathcal{N}_0, j\neq i} x_{ji} - \sum_{j\in\mathcal{N}_0, j\neq i} x_{ij} = 0 \qquad \forall i \in \mathcal{N}_0 \tag{11}$$

$$\sum_{j\in\mathcal{N}_0, j\neq i} x_{ij} \le 1 \qquad \forall i \in \mathcal{N} \tag{12}$$

$$y_i^P \le s_i^0 \qquad \forall i \in \mathcal{P} \tag{13}$$

$$y_i^D \le c_i - s_i^0 \qquad \forall i \in \mathcal{D} \tag{14}$$

$$\sum_{i\in\mathcal{P}\cup\{0\}} y_i^P - \sum_{i\in\mathcal{D}\cup\{0\}} y_i^D = 0 \tag{15}$$

$$\sum_{i\in\mathcal{P}\cup\{0\}} L y_i^P + \sum_{i\in\mathcal{D}\cup\{0\}} U y_i^D + \sum_{i,j\in\mathcal{N}_0, i\neq j} t_{ij} x_{ij} \le T \tag{16}$$

$$q_j \ge q_i + 1 - M(1 - x_{ij}) \qquad \forall i \in \mathcal{N}_0, j \in \mathcal{N}, i \neq j \tag{17}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{N}_0, i \neq j \tag{18}$$

$$y_i^P \ge 0, \text{integer} \qquad \forall i \in \mathcal{P} \cup \{0\} \tag{19}$$

$$y_i^D \ge 0, \text{integer} \qquad \forall i \in \mathcal{D} \cup \{0\} \tag{20}$$

$$y_{ij} \ge 0 \qquad \forall i, j \in \mathcal{N}_0, i \neq j \tag{21}$$

$$s_i \ge 0 \qquad \forall i \in \mathcal{N}_0 \tag{22}$$

$$q_i \ge 0 \qquad \forall i \in \mathcal{N}_0 \tag{23}$$

The objective function (1) is defined as the sum of the penalty cost incurred at each station, which differs from that of Raviv et al. (2013) that used the weighted sum of the total penalty and operational costs. The penalty cost at each station is calculated by the associated penalty function. Each penalty function is defined over the integers from zero to the maximum station capacity because these integers are the possible number of bikes at that station. Because static repositioning occurs during the night, the numbers of bikes at each station before and after the repositioning operation are usually different. The penalty cost of a station is defined by the number of bikes at that station after the operation. Raviv et al. (2013) suggested that the penalty function or cost represents the expected number of shortages for bicycles or lockers during the next working day. They argued that the penalty function has been captured the effect of stochastic bicycle demand and can be estimated using the method proposed by Raviv and Kolka (2013).

Constraints (2)-(4) define the final inventory level at each node at the end of repositioning. The inventory level of the pick-up station at the end of repositioning equals the initial level minus the pick-up quantity whereas the inventory level of the drop-off station at the end of repositioning equals the initial level plus the delivery quantity. Constraints (5) state that the number of bikes obtained from a pick-up station equals the difference in the number of bikes on the vehicle between after and before visiting the station. Constraints (6) state that the number of bikes delivered to a drop-off station equals the difference in the number of bikes on the vehicle between before and after visiting the station. Constraints (7) and (8) define the pick-up and drop-off quantities at the depot, respectively. Constraints (9) are the vehicle capacity constraints, requiring the number of bikes on the vehicle not greater than its capacity. Constraint (10) ensures that the vehicle must leave the depot at least once. Constraints (11) make sure that if the vehicle visits a station, it must leave that station. Constraints (12) ensure that the vehicle can visit a station at most once. Constraints (13) and (14) require that the pick-up and drop-off quantities at each pick-up and drop-off station are not larger than the number of bikes available at the pick-up station and the remaining capacity of the drop-off station, respectively. Constraints (15) make sure that all the picked-up bikes are delivered eventually. Constraints (16) limit the total operating time, including loading, unloading, and travel times, is not greater than the total time available for repositioning. Constraints (17) are the sub-tour elimination constraints (see Miller et al. (1960)). Finally, constraints (18) define x_{ij} to be a binary variable. Constraints (19) and (20) restrict pick-up and drop-off quantities to be nonnegative integers. Constraints (21)-(23) ensure that the number of bikes on the vehicle, the inventory level at each station, and the auxiliary variables are non-negative.

As (1) is not linear, it needs to be linearized in order to be solved by CPLEX. Raviv et al. (2013) show how (1) can be linearized. They also show how their mathematical formulation can be strengthened. Based on their idea, we add the following constraints:

$$y_i^P \leq \min\{s_i^0, k\} \sum_{j \in \mathcal{N}_0, i \neq j} x_{ij} \qquad \forall i \in \mathcal{P} \tag{24}$$

$$y_i^D \leq \min\{c_i - s_i^0, k\} \sum_{j \in \mathcal{N}_0, i \neq j} x_{ij} \qquad \forall i \in \mathcal{D} \tag{25}$$

$$y_i^P \geq \sum_{j \in \mathcal{N}_0, i \neq j} x_{ij} \qquad \forall i \in \mathcal{P} \tag{26}$$

$$y_i^D \geq \sum_{j \in \mathcal{N}_0, i \neq j} x_{ij} \qquad \forall i \in \mathcal{D} \tag{27}$$

Constraints (24)-(25) further tighten the solution space for loading and unloading quantities at each of the pick-up and drop-off stations, respectively, by including the vehicle's capacity, and conditioning the quantities only for cases in which the vehicle visits the corresponding station.

Constraints (26)-(27) tighten the solution space by ensuring that a vehicle enters a station must have a loading and an unloading bike activity, respectively. These constraints are valid when the distance matrix satisfies the triangle inequality because then it is always possible to skip a station without loading and unloading activities.

3 Solution methodology

In our proposed solution, a solution x consists of two parts: 1) a routing sequence $(i_0, i_1, i_2, \ldots, i_n)$ where $i_0 = i_n = 0$ (i.e., the depot), $i_h \in \mathcal{N}, h = 1, 2, \ldots, n-1$, the subscript h is used to define the order of stations being visited, and 2) the decision variable $y^P_{i_h}$ or $y^D_{i_h}$ that describes how many bikes to be picked up or dropped off at each station i_h, respectively. Except for the depot, only one of the decision variables is defined for each i_h, as the station has been classified into either a pick-up or drop-off station before constructing an initial solution. The evaluation function is defined as $z(x) = \sum_{i \in \mathcal{N}} f_i(s_i)$, where s_i is defined by equations (2) and (3). Every neighbor solution x is evaluated based on the evaluation function $z(x)$.

The proposed method consists of the following: the initial solution construction procedure, new solution generation method (based on the insertion, removal and exchange moves) incorporated into a tabu search procedure, and intensification and diversification procedures to further improve and diversify the search.

3.1 Initial solution construction

The initial solution to the repositioning problem is obtained by a simple construction heuristic depicted below:

Step 0 Sort the pick-up and the drop-off stations individually in a descending order based on $|f_i(s^0_i) - f_i(s^I_i)|$, $\forall i \in \mathcal{N}$. Set $l = 1$. Initialize the route as (i_0, i_n).

Step 1 Choose the l-th pick-up station from the list of ordered pick-up stations. Let g be the station number of the station placed in the lth place, and τ be the remaining time of the repositioning operation if station g is appended at the end of the route (i.e., before i_n). If $\tau > 0$, then set $y^P_g = \min\{k - y_{i_{n-1} i_n}, s^0_g - s^I_g, \lfloor \tau/(U + L) \rfloor\}$. If $y^P_g > 0$, then station g is added to the end of the route. Remove station g from the ordered list. Set $l = l + 1$. Step 1 is repeated until no more pick-up stations can be added to the route without violating the constraints. Set $l = 1$.

Step 2 Repeat Step 1 for drop-off stations, replace y^P_g with y^D_g, and the min function with $\min\{y_{i_{n-1} i_n}, s^I_g - s^0_g\}$.

Step 3 Repeat Steps 1 and 2 until no more stations can be appended at the end of the route without violating the constraints.

3.2 Tabu search

Tabu search is a well known metaheuristic proposed by Fred Glover in 1986 (Glover, 1986). Many difficult combinatorial optimization problems (e.g., vehicle routing problems) have been efficiently solved by tabu search. Tabu search is based on the local search principle: At each iteration, the best solution in the neighborhood is chosen. To avoid getting stuck in a local optimum and to prevent cycling of solutions, solutions possessing some attributes of previously-visited-solutions are banned from being chosen for a certain number of iterations (i.e., those attributes are recorded in a tabu list). Sometimes tabu lists may be too powerful as they restrain excellent solutions to be chosen. Aspiration criteria are very useful to overcome this obstacle.

The tabu search heuristic runs γ iterations. At each iteration, a feasible solution that minimizes $z(\bar{x})$ is chosen from the three neighborhoods: $\mathcal{R}(x)$, $\mathcal{I}(x)$ and $\mathcal{E}(x)$. The selected solution is either a non-tabu solution or a solution that satisfies the aspiration criterion. If the objective function value evaluated at the selected solution is less than that evaluated at the current best z_1^*, we update the current best solution z_1^*. The reverse move is set tabu for a certain number of iterations. Two tabu lists are maintained: $TABU_1(i)$ denotes when station i can be removed from the solution and $TABU_2(i)$ states when station i can be inserted into the solution again. The selected solution is set to be the current solution. The algorithmic description of the heuristic is given in Algorithm 1.

Algorithm 1 `TabuSearch`(x_0)

```
Require: Initial solution $x_0$
 1: Set $\breve{x}^* = x_0$ and $z_1^* = z(x_0)$.
 2: Set $TABU_1(i) = -1$ and $TABU_2(i) = -1 \ \forall \ i \in \mathcal{N}$.
 3: Set $x = x_0$.
 4: for $c = 1, \ldots, \gamma$ do
 5:     Select a solution $\bar{x} \in \mathcal{R}(x) \cup \mathcal{I}(x) \cup \mathcal{E}(x)$ that minimizes $z(\bar{x})$, and the solution is either a
        non-tabu solution or a solution that satisfies the aspiration criterion (i.e., $z(\bar{x}) < z_1^*$).
 6:     if $\bar{x} \neq \emptyset$ then
 7:         If $z(\bar{x}) < z_1^*$, set $\breve{x}^* = \bar{x}$ and $z_1^* = z(\bar{x})$.
 8:         Set the move from $\bar{x}$ to $x$ tabu for $\theta_1$ and/or $\theta_2$ iterations.
 9:         Set $x = \bar{x}$.
10:     end if
11: end for
12: return $\breve{x}^*$.
```

3.3 The removal move

The neighborhood $\mathcal{R}(x)$ is a set of feasible neighbor solutions obtained by applying the removal moves. Each of the removal moves removes station $i_h \in W(x) \setminus \{0\}$, where $W(x)$ is the set of stations used to define the route sequence in the solution x. Sometimes when a station is removed from the solution, the resultant solution may no longer remain feasible as the values of the variables $y_{i_d}^P$ and $y_{i_d}^D$ where $d > h$ may no longer satisfy the vehicle capacity constraint and/or the non-negativity constraint. Instead of recomputing the pick-up or drop-off quantity at each station for the resultant solution, we only modify the quantity $y_{i_0}^P$, $y_{i_{h-1}}^P$, $y_{i_{h-1}}^D$, $y_{i_{h+1}}^P$ or $y_{i_{h+1}}^D$ in the resultant solution to generate a feasible neighbor solution efficiently. With each i_h that gets removed, a maximum of three feasible neighbor solutions can be generated. These three potential neighbor solutions have the same routing sequence. The only difference between the first potential neighbor solution and solution x is that they have different values of $y_{i_0}^P$. The difference between the second (third) potential neighbor solution and x is that they have different values of $y_{i_{h-1}}^P$ or $y_{i_{h-1}}^D$ ($y_{i_{h+1}}^P$ or $y_{i_{h+1}}^D$). The determination of the values of $y_{i_0}^P$, $y_{i_{h-1}}^P$, $y_{i_{h-1}}^D$, $y_{i_{h+1}}^P$ or $y_{i_{h+1}}^D$ is described in Sections 3.3.1 and 3.3.2.

3.3.1 Station i_h is a pick-up station

When i_h is a pick-up station, we consider three possible ways in adjusting the pick-up/drop-off quantities in the resultant solution:

Adjusting $y^P_{i_0}$ Denote r as the minimum residual capacity of the vehicle before visiting station i_h. Then, r is obtained by $\min_{l=0,1,\ldots,h-1}\{k - y_{i_l i_{l+1}}\}$. It is possible to adjust the quantity $y^P_{i_0}$ in the resultant solution to obtain a feasible neighbor solution if the following three conditions are satisfied by the current solution simultaneously: $r \geq y^P_{i_h}$, $y^P_{i_0} + y^P_{i_h} \leq k$, and $s^0_{i_0} - y^P_{i_0} - y^P_{i_h} \geq 0$. The three conditions ensure that $y^P_{i_h}$ extra bikes can be loaded to the vehicle at the depot while satisfying the vehicle capacity constraint. If the three conditions are satisfied, then the pick-up quantity at the depot in the neighbor solution is set to be $y^P_{i_0} + y^P_{i_h}$, and the quantities associated with other stations remain unchanged.

Adjusting $y^P_{i_{h-1}}$ or $y^D_{i_{h-1}}$ If i_{h-1} is a pick-up station, then it is possible to adjust $y^P_{i_{h-1}}$ in the resultant solution to form a feasible neighbor solution if the current solution satisfies $s^0_{i_{h-1}} - y^P_{i_{h-1}} - y^P_{i_h} \geq 0$. This condition ensures that node i_{h-1} can provide $y^P_{i_h}$ extra bikes for pick-up. The pick-up quantity at that node in the resultant solution is set to be $y^P_{i_{h-1}} + y^P_{i_h}$ to obtain a feasible neighbor solution. If i_{h-1} is a drop-off station, then the neighbor solution is feasible if $s^0_{i_{h-1}} + y^D_{i_{h-1}} - y^P_{i_h} \leq c_{i_{h-1}}$ (i.e., drop off fewer bikes at i_{h-1}) and $y^D_{i_{h-1}} > y^P_{i_h}$. If it is feasible, then set $y^D_{i_{h-1}} = y^D_{i_{h-1}} - y^P_{i_h}$. If in the latter case, $y^D_{i_{h-1}} = y^P_{i_h}$, then i_{h-1} is also eliminated from the route and that is not ideal.

Adjusting $y^P_{i_{h+1}}$ or $y^D_{i_{h+1}}$ Same as above, except replace $h-1$ with $h+1$.

3.3.2 Station i_h is a drop-off station

We also consider three possible ways in adjusting the pick-up/drop-off quantities in the resultant solution when i_h is a drop-off station. They are depicted below.

Adjusting $y^P_{i_0}$ Let u be the minimum vehicle load before visiting station i_h. Then, u is computed as $\min_{l=0,1,\ldots,h-1}\{y_{i_l i_{l+1}}\}$. Then, we may adjust $y^P_{i_0}$ in the resultant solution to obtain a feasible neighbor solution if the current solution x satisfies the following: $u - y^D_{i_h} \geq 0$ and $y^P_{i_0} - y^D_{i_h} \geq 0$. It is because the vehicle can be loaded with fewer bikes at the depot while satisfying the constraints if the two conditions are satisfied. In this case, $y^P_{i_0}$ is set to be $y^P_{i_0} - y^D_{i_h}$, and the other quantities remain unchanged.

Adjusting $y^D_{i_{h-1}}$ or $y^P_{i_{h-1}}$ If i_{h-1} is a drop-off station, then the resultant solution is feasible if the current solution satisfies $s^0_{i_{h-1}} + y^D_{i_{h-1}} + y^D_{i_h} \leq c_{i_{h-1}}$. In this case, more bikes will be dropped off at i_{h-1} and $y^D_{i_{h-1}}$ is set to be $y^D_{i_{h-1}} + y^D_{i_h}$. If i_{h-1} is a pick-up station, then the resultant solution is feasible if $s^0_{i_{h-1}} - y^P_{i_{h-1}} + y^D_{i_h} \geq 0$ and $y^P_{i_{h-1}} > y^D_{i_h}$. If it is feasible, then set $y^P_{i_{h-1}} = y^P_{i_{h-1}} - y^D_{i_h}$. We do not allow $y^P_{i_{h-1}}$ equal to $y^D_{i_h}$ to avoid i_{h-1} being eliminated from the route.

Adjusting $y^D_{i_{h+1}}$ or $y^P_{i_{h+1}}$ Same as above, except replace $h-1$ with $h+1$.

3.4 The insertion move

The neighborhood $\mathcal{I}(x)$ consists of all feasible neighbor solutions obtained by applying the insertion moves. Each of the insertion moves involves two steps. The first step is to insert a station $i \in (\mathcal{P} \cup \mathcal{D}) \setminus W(x)$ in the routing sequence in the current solution x. The station can be added between every pair of nodes i_{h-1} and i_h in the sequence. The station can also be added after the depot but the next visiting node must also be the depot. The second step is to set the quantity at each station in the resultant solution, including the quantity at the inserted station. The insertion move is feasible if the travel time on the resultant route does not violate constraint (16) and the new quantities computed satisfy constraints (5)-(8), (13)-(15), (19)-(20). To save computation time, instead of recomputing the quantities at each station in the resultant solution, we set some of the quantities equal to the corresponding quantities in the current solution, compute y_i^P or y_i^D, and only adjust the quantity y_0^P, $y_{i_{h-1}}^P$, $y_{i_{h-1}}^D$, $y_{i_h}^P$ or $y_{i_h}^D$ to obtain a feasible neighbor solution. The rules used for modifying the quantities are described in Sections 3.4.1 and 3.4.2.

3.4.1 Station i is a pick-up station

We consider four possible ways in adjusting the quantities in the resultant solution after inserting pick-up station i:

No need to adjust other stations' quantities Let τ be the remaining time left if station i is inserted between i_{h-1} and i_h, and κ be the maximum vehicle load after visiting station i_{h-1}. Then, $\kappa = \max_{l=h-1,h,\ldots,n-1}\{y_{i_l i_{l+1}}\}$ and $y_i^P = \min\{\lfloor \tau/(U+L) \rfloor, s_i^0, k-\kappa\}$, which depends on the remaining time left, the number of bikes available at the station, and the spare vehicle capacity. If $y_i^P > 0$, then the insertion of i between i_{h-1} and i_h is feasible without altering any of the other stations' pick-up/drop-off quantities.

Adjusting $y_{i_0}^P$ This case is possible if the original pick-up quantity $y_{i_0}^P > 0$ as we aim to reduce the pick-up quantity at the depot and the reduced number is compensated by the pick-up quantity at station i, i.e., $y_i^P = \min\{s_i^0, u\}$, where $u = \min_{l=0,\ldots,h-1}\{y_{i_l i_{l+1}}\}$. Including u in the formula is because we cannot reduce too many bikes from i_0; otherwise, the load on the vehicle traveling between some stations would be negative. If $y_i^P > 0$, then station i can be inserted between i_{h-1} and i_h with depot's quantity reduced by y_i^P (i.e., $y_{i_0}^P = y_{i_0}^P - y_i^P$).

Adjusting $y_{i_{h-1}}^P$ or $y_{i_{h-1}}^D$ There are two mutually exclusive cases:

1. If i_{h-1} is a pick-up station: It is possible to pick-up fewer bikes at i_{h-1} and let the vehicle pick-up the rest from i. The quantity y_i^P in the resultant solution is determined from

$$\underset{v=0,1,\ldots,\min\{s_i^0, y_{i_{h-1}}^P\}}{\arg\min} \quad f_{i_{h-1}}(s_{i_{h-1}}^0 - y_{i_{h-1}}^P + v) + f_i(s_i^0 - v).$$

   The insertion is feasible if $y_i^P > 0$ and $y_{i_{h-1}}^P > y_i^P$ as this move aims to pick-up y_i^P fewer bikes from i_{h-1}. In this case, $y_{i_{h-1}}^P = y_{i_{h-1}}^P - y_i^P$.

2. If i_{h-1} is a drop-off station: For every additional bike the vehicle drops off at station i_{h-1}, a bike is picked up from station i. The quantity y_i^P equals $\min\{y_{i_{h-1}i_h}, c_{i_{h-1}} - s_{i_{h-1}}^0 - y_{i_{h-1}}^D, s_i^0, \lfloor \tau/(U+L) \rfloor\}$. The insertion is feasible if $y_i^P > 0$ because this move can drop off y_i^P more bikes at i_{h-1}, In this case, $y_{i_{h-1}}^D = y_{i_{h-1}}^D + y_i^P$.

Adjusting $y^P_{i_h}$ or $y^D_{i_h}$ There are two mutually exclusive cases:

1. If i_h is a pick-up station: Same as above, except replace $h-1$ with h.
2. If i_h is a drop-off station: For every bike the vehicle picks up from station i, a bike is dropped off at station i_h. The quantity y^P_i is $\min\{k-y_{i_{h-1}i_h}, c_{i_h}-s^0_{i_h}-y^D_{i_h}, s^0_i, \lfloor\tau/(U+L)\rfloor\}$. The insertion is feasible if $y^P_i > 0$ so that the vehicle can drop off y^P_i more bikes at i_h. In this case, $y^D_{i_h} = y^D_{i_h} + y^P_i$.

3.4.2 Station i is a drop-off station

We consider five possible ways in adjusting the quantities in the resultant solution after inserting drop-off station i:

No need to adjust other stations' quantities Denote ν as the minimum vehicle load after visiting station i_{h-1}. Then, $\nu = \min_{l=h-1,h,\dots,n-1}\{y_{i_l i_{l+1}}\}$ and $y^D_i = \min\{c_i - s^0_i, \nu\}$. If $y^D_i > 0$, then the insertion of i between i_{h-1} and i_h is feasible without altering any of the other stations' pick-up/drop-off quantities.

Adjusting $y^P_{i_0}$ This case is possible if $y^P_{i_0} < k$ as it is only possible to drop off at i if we increase the pick-up quantity at the depot. The quantity $y^D_i = \min\{\lfloor\tau/(U+L)\rfloor, c_i - s^0_i, r, s^0_{i_0} - y^P_{i_0}\}$, where r is defined in Section 3.3.1. If $y^D_i > 0$, then it is feasible to insert i between i_{h-1} and i_h, where y^D_i more bikes are picked up from the depot (i.e., $y^P_{i_0} = y^P_{i_0} + y^D_i$).

Adjusting $y^D_{i_{h-1}}$ or $y^P_{i_{h-1}}$ There are two mutually exclusive cases:

1. If i_{h-1} is a drop-off station: It is possible to drop off fewer at i_{h-1} and drop off the rest at i. The quantity y^D_i is determined from

$$\underset{\upsilon=0,1,\dots,\min\{c_i-s^0_i, y^D_{i_{h-1}}\}}{\arg\min} \quad f_{i_{h-1}}(s^0_{i_{h-1}} + y^D_{i_{h-1}} - \upsilon) + f_i(s^0_i + \upsilon).$$

   The insertion is feasible if $y^D_i > 0$ and $y^D_{i_{h-1}} > y^D_i$ as the vehicle can drop off y^D_i fewer bikes at i_{h-1} (i.e., $y^D_{i_{h-1}} = y^D_{i_{h-1}} - y^D_i$).
2. If i_{h-1} is a pick-up station: For every additional bike the vehicle picks up from station i_{h-1}, a bike is dropped off at station i. The quantity y^D_i is $\min\{k - y_{i_{h-1}i_h}, s^0_{i_{h-1}} - y^P_{i_{h-1}}, c_i - s^0_i, \lfloor\tau/(U+L)\rfloor\}$. The insertion is feasible if $y^D_i > 0$ because the vehicle can pick up y^D_i more bikes at i_{h-1}. In this case, $y^P_{i_{h-1}} = y^P_{i_{h-1}} + y^D_i$.

Adjusting $y^D_{i_h}$ or $y^P_{i_h}$ There are two mutually exclusive cases:

1. If i_h is a drop-off station: Same as above, except replace $h-1$ with h.
2. If i_h is a pick-up station: For every bike the vehicle drops off at station i, a bike is picked up at station i_h. The quantity y^D_i is $\min\{y_{i_{h-1}i_h}, c_i - s^0_i, s^0_{i_h} - y^P_{i_h}, \lfloor\tau/(U+L)\rfloor\}$. The insertion is feasible if $y^D_i > 0$ as the vehicle can pick up y^D_i more bikes at i_h. In this case, $y^P_{i_h} = y^P_{i_h} + y^D_i$.

Adjusting pick-up quantities at the depot due to creating a new trip When a new trip is created by inserting both i and the depot after the depot in the route sequence (i.e., $0-i-0$), the quantity, y^D_i, is set to be $\min\{c_i - s^0_i, k, s^0_{i_0} - y^P_{i_0}, \lfloor\tau/(U+L)\rfloor\}$. This is also the pick-up quantity from the depot.

Note that the term ”trip” of a vehicle means the vehicle leaves the depot, visits one or more stations, and then return to the depot. The vehicle may make multiple trips on a single route and hence the vehicle can visit the depot multiple times if necessary.

3.5 The exchange move

The neighborhood $\mathcal{E}(x)$ consists of all feasible neighbor solutions obtained by applying the exchange moves. Each of the exchange moves works as follows: It exchanges station $i_h \in W(x) \cap \mathcal{P}$ with station $i \in \mathcal{P} \setminus W(x)$ in position h in the routing sequence in the current solution x and set the pick-up quantity at i to be that at i_h. This exchange is feasible if $s_i^0 - y_{i_h}^P \geq 0$ and the resultant traveling time does not violate constraint (16). Similarly, this move also exchanges station $i_h \in W(x) \cap \mathcal{D}$ with station $i \in \mathcal{D} \setminus W(x)$ and set the drop-off quantity at i equal to that at i_h. The exchange is feasible if $s_i^0 + y_{i_h}^D \leq c_i$ and constraint (16) is not violated.

3.6 Tabu tenures and aspiration criterion

Tabu tenures are set according to the planning horizon T and the number of stations $|\mathcal{N}|$. We utilize two tabu lists; one for the removal of stations, $TABU_1(\cdot)$, and one for the insertion of stations, $TABU_2(\cdot)$. The first tabu tenure θ_1 is set to equal $[(T/3600)\log_{10}|\mathcal{N}|]$, where T is given in seconds and $[\cdot]$ is the nearest integer function. The second tabu tenure θ_2 equals $2\theta_1$ for short planning horizon instances, and $\theta_2 = [1.5\theta_1]$ for the instances of long planning horizon.

Insertion moves are set tabu as follows: $TABU_1(i) = c + \theta_1$ if the solution $\bar{x}$ is selected and obtained by inserting station i into the solution x at iteration c in Algorithm 1. Hence, the reverse move is declared forbidden for θ_1 iterations. Removal moves are set tabu in a similar way: $TABU_2(i_h) = c + \theta_2$ if $\bar{x}$ is selected and obtained by removing station i_h from x at iteration c. Thus, inserting i_h back to the solution is declared forbidden for θ_2 iterations. Tabu exchange moves are set as follows: $TABU_1(i) = c + \theta_1$ and $TABU_2(i_h) = c + \theta_2$ if $\bar{x}$ is selected and obtained by removing station i_h from x and inserting i into x at iteration c.

When evaluating the removal neighborhood $\mathcal{R}(x)$ and the insertion neighborhood $\mathcal{I}(x)$, a neighbor solution is considered tabu if $TABU_1(i_h) \geq c$ and $TABU_2(i) \geq c$, respectively. For the exchange neighborhood $\mathcal{E}(x)$, a neighbor solution is considered tabu if both $TABU_1(i_h) \geq c$ and $TABU_2(i) \geq c$.

Tabu lists can sometimes be too powerful as excellent solutions are not allowed to be chosen. A remedy is to apply an aspiration criterion in order to revoke the tabu status. The aspiration criterion used in the heuristic is the most widely used criterion. If $z(x) < z(x^*)$, where x^* is the best solution encountered so far, then the solution x can be chosen even though the solution is obtained by a forbidden move as it is obvious that x has never been encountered before.

3.7 Intensification and diversification procedures

After tabu search, the following intensification and diversification procedures are applied to the best solution from tabu search.

Adjusting the quantities Sometimes a solution can be improved by re-adjusting the assigned quantities among the stations (in a heuristically way). In this procedure, we have four cases. In the first two cases, we look at a pair of stations i_m and i_h (of the same type) at a time, and check whether one/several units of the assigned quantity can be shifted from one station to the other. In the third case, we check whether the drop-off quantity of the station just before visiting the depot can be increased by dropping off more bikes at that station and dropping off fewer bikes at the depot. Finally, in the fourth case, we check whether the pick-up quantity of the station just after visiting the depot can be increased (decreased) by picking up more (fewer) bikes at that station and picking up fewer (more) bikes at the depot.

1. If stations i_m and i_h are pick-up stations and $m < h$: The number of bikes picked up at these two stations can be modified by either the forward or backward adjustment method. The forward (backward) adjustment procedure requires that some of the bikes, which are originally obtained from i_m (i_h), can be picked up at i_h (i_m) instead. For the forward adjustment method, we compute the adjustment $\rho = \arg\min_{\nu=0,1,\ldots,\min\{y^P_{i_m}, s^0_{i_h}-y^P_{i_h}\}} f_{i_m}(s^0_{i_m} - y^P_{i_m} + \nu) + f_{i_h}(s^0_{i_h} - y^P_{i_h} - \nu)$ subject to $y_{i_l i_{l+1}} - \nu \geq 0, l = m+1, \ldots, h-1$. Hence, $y^P_{i_m} = y^P_{i_m} - \rho$ and $y^P_{i_h} = y^P_{i_h} + \rho$. The backward adjustment is computed as $\rho = \arg\min_{\nu=0,1,\ldots,\min\{y^P_{i_h}, s^0_{i_m}-y^P_{i_m}, k-y_{i_m i_{m+1}}\}} f_{i_m}(s^0_{i_m} - y^P_{i_m} - \nu) + f_{i_h}(s^0_{i_h} - y^P_{i_h} + \nu)$ subject to $y_{i_l i_{l+1}} + \nu \leq k, l = m+1, \ldots, h-1$. Hence, $y^P_{i_m} = y^P_{i_m} + \rho$ and $y^P_{i_h} = y^P_{i_h} - \rho$.
2. If stations i_m and i_h are drop-off stations and $m < h$: The forward adjustment is computed as $\rho = \arg\min_{\nu=0,1,\ldots,\min\{y^D_{i_m}, c_{i_h}-s^0_{i_h}-y^D_{i_h}\}} f_{i_m}(s^0_{i_m} + y^D_{i_m} - \nu) + f_{i_h}(s^0_{i_h} + y^D_{i_h} + \nu)$ subject to $y_{i_l i_{l+1}} + \nu \leq k, l = m+1, \ldots, h-1$. Hence, $y^D_{i_m} = y^D_{i_m} - \rho$ and $y^D_{i_h} = y^D_{i_h} + \rho$. For the backward adjustment, we compute
$$\rho = \arg\min_{\nu=0,1,\ldots,\min\{y_{i_m i_{m+1}}, y^D_{i_h}, c_{i_m}-s^0_{i_m}-y^D_{i_m}\}} f_{i_m}(s^0_{i_m} + y^D_{i_m} + \nu) + f_{i_h}(s^0_{i_h} + y^D_{i_h} - \nu)$$
subject to $y_{i_l i_{l+1}} - \nu \geq 0, l = m+1, \ldots, h-1$. Hence, $y^D_{i_m} = y^D_{i_m} + \rho$ and $y^D_{i_h} = y^D_{i_h} - \rho$.
3. If i_h is a drop-off station, $i_{h+1} = 0$, and $y_{i_h i_{h+1}} > 0$: We compute
$$\rho = \arg\min_{\nu=0,1,\ldots,\min\{c_{i_h}-s^0_{i_h}-y^D_{i_h}, y_{i_h i_{h+1}}\}} f_{i_h}(s^0_{i_h} + y^D_{i_h} + \nu).$$
If $\rho > 0$, then the drop-off quantity at i_h is increased by ρ (i.e., $y^D_{i_h} = y^D_{i_h} + \rho$).
4. If i_m is a pick-up station, $i_{m-1} = 0$, and $y_{i_{m-1} i_m} > 0$: We calculate the adjustment $\rho = \arg\min_{\nu=0,1,\ldots,\min\{s^0_{i_m}-y^P_{i_m}, y_{i_{m-1} i_m}\}} f_{i_m}(s^0_{i_m} - y^P_{i_m} - \nu)$. If $\rho > 0$, then $y^P_{i_m} = y^P_{i_m} + \rho$ and $y^P_{i_{m-1}} = y^P_{i_{m-1}} - \rho$. Otherwise, $\rho = \arg\min_{\nu=0,1,\ldots,\min\{k-y_{i_{m-1} i_m}, y^P_{i_m}\}} f_{i_m}(s^0_{i_m} - y^P_{i_m} + \nu)$. If $\rho > 0$, then $y^P_{i_m} = y^P_{i_m} - \rho$ and $y^P_{i_{m-1}} = y^P_{i_{m-1}} + \rho$.

Note that by adjusting the pick-up/drop-off quantities, one or several quantities may be reduced to zero. In case they equal zero, the corresponding stations are removed from the routing sequence.

2-opt This local search procedure removes two edges from the solution and adds two new edges back so it remains a tour (Lin, 1965). The objective is about reducing the tour-length. The procedure disregards the pick-up or drop-quantities and employs the best-improvement (BI) strategy.

Intra-relocation This local search procedure is also about reducing the tour-length (and disregarding the pick-up/drop-off quantities) by relocating a station from its position in the route sequence and reinserting it into some other position until no improvements of the tour-length can be achieved.

Removing a station This procedure removes station i from the solution x. It chooses the station with the least reduction of the value of objective function (1). The selection criterion is: $\arg\min_{i \in W(x)} \{f_i(s_i^0) - f_i(s_i^0 - r) : r = \begin{cases} y_i^P, & \text{if } i \in \mathcal{P}; \\ -y_i^D, & \text{if } i \in \mathcal{D}. \end{cases} \}$.

Inserting a station This is the same as described in Section 3.4 where the best solution is selected from the neighborhood, except that tabu list is not used in this procedure.

Algorithm 2 Perturbation($\hat{x}$,z^*)

```
Require: Tabu search solution $\hat{x}$ and the best overall solution value $z^*$
 1: $\breve{z} = z(\hat{x})$
 2: if $z(\hat{x}) < z^*$ then
 3:   $\hat{x}$ = AdjustQuantities($\hat{x}$)
 4:   $\hat{x}$ = 2-opt($\hat{x}$)
 5:   $\hat{x}$ = IntraRelocation($\hat{x}$)
 6:   $\hat{x}$ = Repairing($\hat{x}$)
 7:   $\hat{x}$ = AdjustQuantities($\hat{x}$)
 8:   for $a = 1$ to $5$ do
 9:     $\hat{x}$ = InsertStation($\hat{x}$)
10:     $\hat{x}$ = AdjustQuantities($\hat{x}$)
11:   end for
12:   $\hat{x}$ = RemoveStation($\hat{x}$)
13:   $\hat{x}$ = Repairing($\hat{x}$)
14:   $\hat{x}$ = AdjustQuantities($\hat{x}$)
15: else
16:   $\hat{x}$ = AdjustQuantities($\hat{x}$)
17:   for $a = 1$ to $\phi$ do
18:     $\hat{x}$ = RemoveStation($\hat{x}$)
19:   end for
20:   $\hat{x}$ = 2-opt($\hat{x}$)
21:   $\hat{x}$ = IntraRelocation($\hat{x}$)
22:   $\hat{x}$ = Repairing($\hat{x}$)
23:   $\hat{x}$ = AdjustQuantities($\hat{x}$)
24:   for $a = 1$ to $5$ do
25:     $\hat{x}$ = InsertStation($\hat{x}$)
26:     $\hat{x}$ = AdjustQuantities($\hat{x}$)
27:   end for
28: end if
29: return the best solution $\breve{x}^*$ and the final solution $\hat{x}$
```

After applying the 2-opt, the intra-relocation, and the removal-of-station procedures, the solution is usually infeasible in terms of the pick-up/drop-off quantities. The solution needs to be feasible before the AdjustQuantities procedure can be applied. Hence, the solution will be repaired by initializing all the pick-up/drop-off quantities at stations with zero and then assigning the pick-up/drop-off quantity at each station with a minimal number of bikes so that the solution becomes feasible. Then, one pick-up station and one drop-off station are selected and their pick-up and drop-off quantities are increased by at most two to maximize the reduction of the objective value. The problem of determining this pair of stations (p, d) and the increase in the pick-up/drop-off quantity r is formulated as follows.

$$(p,d) = \underset{i \in W(x)\cap\mathcal{P}, j \in W(x)\cap\mathcal{D}}{\arg\max} \{f_i(s_i^0 - y_i^P) - f_i(s_i^0 - y_i^P - r) + f_j(s_j^0 + y_j^D) - f_j(s_j^0 + y_j^D + r) :$$
$$r = \min\{s_i^0 - y_i^P, c_j - s_j^0 - y_j^D, \lfloor \tau/(U+L) \rfloor, 2\}\}$$

subject to the capacity and loading constraints. Afterwards, the pick-up and drop-off quantities are updated ($y_p^P = y_p^P + r$ and $y_d^D = y_d^D + r$) and the procedure is repeated until τ does not permit to further increase the pick-up/drop-off quantity of any stations or the objective value could not be reduced.

The perturbation procedure is given in Algorithm 2 and is divided into two parts; the first part is devoted to when an overall best solution has been attained in the preceding tabu search, and the other part is for when an overall best solution has not been obtained. There are some differences between the two parts. First, the above mentioned procedures are applied in a slightly different order. In the first part, the removal of a station is executed at the end of Algorithm 2, while in the second part it is executed in the beginning. This is because in the second part tabu search has not been able to generate a better solution than the current best solution x^*. Hence, the algorithm removes ϕ stations in the beginning in order to diversify the search in the hope of moving into some other more promising regions of the solution space. Second, the strategy used in 2-opt of the first part is the BI strategy. In the second part, the BI strategy is also used but with some exceptions. If the BI strategy results in relocating station $i_1 \in \mathcal{P}$ somewhere else in the route and station $i_2 \in \mathcal{D}$, then this change is not implemented. Rather, the next best choice is implemented instead. This is because the objective value is usually deteriorated if the first visited pick-up station is relocated, where most likely its pick-up quantity is picked up at the depot instead (which does not contribute anything to the objective function value).

After the route length is shortened by 2-opt and IntraRelocation, there is usually room for inserting one or more stations into the route. Our preliminary experiments found that inserting a maximum of five stations into the route is a good choice.

If any of the solutions improves $\breve{z}$, then the solution is saved as $\breve{x}^*$ and $\breve{z}$ is updated. The best solution $\breve{x}^*$ and the final solution $\hat{x}$ are returned from the Perturbation procedure.

3.8 The overall algorithm

To further improve the solutions, the tabu search procedure is embedded into an iterative framework (see Algorithm 3), where several intensification/diversification mechanisms (see Section 3.7) are applied to the solution obtained from tabu search. Algorithm 3 is run until no improvement can be achieved or it has run for at least β iterations. Preliminary computational experiments have shown that when $K = 1$ and $w < \beta$ (with $\beta \geq 5$) is left out, the algorithm may sometimes terminate after only two or three iterations which results in inferior solutions. When the algorithm runs for at least five iterations, the solutions are significantly better.

The algorithm continues with the final solution $\breve{x}$ from the Perturbation procedure rather than from the best solution x'. Computational experiments have also shown that restarting the search from $\breve{x}$ is more beneficial from a diversification point of view. Otherwise, it may be difficult to leave valleys in the search space (especially the deep ones) as only small changes are made in each iteration of the heuristic.

Algorithm 3 Iterated Tabu Search

```
1: $x_0$ = InitialSolution()
2: Set $x^* = x_0$, $z^* = z(x^*)$, $iterWI = 0$ and $w = 0$.
3: while $iterWI < K$ or $w < \beta$ do
4:     $\hat{x}$ = TabuSearch($x_0$)
5:     $(x', \breve{x})$ = Perturbation($\hat{x}$,$z^*$)
6:     if $z(x') < z^*$ then
7:         Set $x^* = x'$, $z^* = z(x^*)$ and $iterWI = 0$.
8:     else
9:         Set $iterWI = iterWI + 1$.
10:    end if
11:    Set $w = w + 1$.
12:    Set $x_0 = \breve{x}$.
13: end while
14: return $x^*$
```

4 Computational experiments

The heuristic was coded in C++ and all computational experiments were carried out on a Dell notebook with an Intel Core i5-2520M CPU @ 2.5 GHz. Three sets of instances were used to conduct the experiments in this study. The sets include:

Set 1 It contains the Paris instances (up to 100 stations) used in the study by Raviv et al. (2013). They showed the results for 30 and 60 stations, and up to two vehicles. However, in this paper, instances of varying numbers of stations (30 to 100 stations) were used. The vehicle capacity is fixed to 20. There is a total of 12 instances in this set.

Set 2 This set contains instances of 100-400 stations, with a step size of 25. The smaller instances are subsets of the larger ones. The locations of the different stations were taken from the benchmark instances for the 1-PDTSP studied by Hernández-Pérez and Salazar-González (2004b). As in Raviv et al. (2013), the following setting was used. *a*) Manhattan distances (see the definition in `http://en.wiktionary.org/wiki/Manhattan_distance`) were used as they are more realistic than Euclidean distances. *b*) Two repositioning times (in seconds) were considered: $T = 9000$ and $T = 18000$. *c*) The time for picking up or dropping off one bike is 60 seconds. *d*) Other input data (i.e., s_i^0, c_i, and $f_i(\cdot)$) were taken from their instances. As their data do not cover all 400 stations, the original data were replicated when necessary. The vehicle capacity in each instance varied from 10 to 40, with a step size of 10. This set contains 104 instances.

Set 3 This set contains instances ranging from 200 to 400 stations. The locations of the stations were also taken from the benchmark instances for the 1-PDTSP. The vehicle capacity is 20. Repositioning times and loading/unloading times are the same as above. Other input data were randomly generated. This set contains a total of 30 instances.

The first set of instances was included in the experiments because our problem is closely related to the problem studied by Raviv et al. (2013). Because of the same reason, their solution method (i.e. CPLEX) was used for comparison. The instances used by Raviv et al. (2013) contain 30-100 stations and we believe that the sizes of the instances do not justify the usage of heuristics. Hence, the experiments were also conducted on larger instances, i.e., the second and third sets of instances with 100-400 stations.

The results from the heuristic were obtained by setting $K = 1$, $\beta = 5$, and $\gamma = 200$ after some preliminary experiments. The parameter ϕ was set to $[0.5\eta]$ for the instances of short planning horizon (i.e., $T = 9000$), and $[0.3\eta]$ for the instances of long planning horizon (i.e., $T = 18000$), where η is the number of stations visited by the vehicle.

The results obtained from the heuristic were compared to those obtained from CPLEX 12.4 with the default settings and a maximum running time of 2 hours. For each instance, CPLEX returned a lower bound, an upper bound (z^{UB}), and a solution x^* (if feasibility was achieved within those 2 hours). In principle, if CPLEX terminates before the 2-hour limit, it implies that x^* is the optimal solution and $z(x^*) = z^{UB}$. However, this may not always be true regarding the experiments conducted in this study as it has been verified that for some instances, this holds: $z(x^*) < z^{UB}$. The discrepancy between $z(x^*)$ and z^{UB} may be due to some internal rounding errors in CPLEX. Hence, $z(x^*)$ instead of z^{UB} is reported for all instances in which optimality has been proven by CPLEX.

Tables 2 and 3 present the results from the experiments conducted on the first set of instances in which optimality has been proven by CPLEX for most of them. Opt denotes the *true* optimal objective value of the solution x^*, i.e., $z(x^*)$, and Heuristic denotes the result from the heuristic. Gap denotes the deviation (in %) of the results from Opt. CPU is the running time of CPLEX or the heuristic (in seconds). $\psi = \sum_{i\in\mathcal{N}} y_i^D$ and $\varphi = \sum_{i\in\mathcal{N}} y_i^P$ denote the total number of bikes dropped off and picked up at all of the visited stations, respectively. Done is the percentage of job done, and is the percentage reduction in the expected shortages. This is computed using the formula stated in Raviv et al. (2013). Finally, $\varsigma = \sum_{i,j\in\mathcal{N}_0, i\neq j} t_{ij}x_{ij}$ is the total travel time in seconds.

Table 2: Results of Raviv's instances, $\beta = 5$

$\lvert\mathcal{N}_0\rvert$	T	Opt	Gap	CPU	Heuristic	CPU	η	ψ	φ	Done	ς
30	9000	214.16	0.570	22.75	215.38	0.225	8	51	42	40.26	2854
	18000	167.07	0.431	523.75	167.79	0.245	18	95	75	66.63	6488
45	9000	335.05	0.546	790.75	336.88	0.305	10	45	45	30.60	3552
	18000	273.09	1.058	757.04	275.98	0.375	18	100	80	55.49	5992
60	9000	463.45	-0.047	7200.00	*463.23*	0.230	9	46	45	22.50	3468
	18000	382.52	1.406	638.10	387.90	0.450	20	100	81	45.27	5950
75	9000	568.57	0.026	1437.20	586.72	0.465	8	47	47	21.01	3270
	18000	486.15	0.084	1275.20	486.56	0.455	19	97	96	42.54	6350
90	9000	692.00	0.062	323.09	692.43	0.745	11	49	49	18.86	3042
	18000	609.48	0.000	1999.15	**609.48**	0.815	20	97	97	37.40	6334
100	9000	782.69	0.055	760.00	783.12	0.760	11	49	49	17.07	3042
	18000	700.17	0.734	2654.00	705.31	0.785	20	93	93	32.80	6814
Avg			0.410	1531.748		0.488					

From Tables 2 and 3, we observe that CPLEX found the optimal solution to almost all of the instances from Set 1 within the 2-hour limit. The results obtained by our heuristic are 0.41% from the optimal values on average and mean computing time is 0.488 second. Our heuristic found the optimal solution for one instance (the optimal objective value is bolded in Tables 2 and 3), and improved the bound on another instance (which gives a negative gap value in Tables 2 and 3). When letting the heuristic run a little longer (e.g., $\beta = 30$), the average deviation is then reduced to 0.187% with a slightly increase in computing time (on average 2 seconds more). Compared with the average running time of CPLEX (i.e., 1531.748 seconds), our heuristic could

Table 3: Results of Raviv's instances, $\beta = 30$

$\|\mathcal{N}_0\|$	T	**Opt**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
30	9000	214.16	0.570	22.75	215.38	0.857	8	51	42	40.26	2854
	18000	167.07	0.305	523.75	167.58	1.279	17	96	76	66.75	6390
45	9000	335.05	0.546	790.75	336.88	1.403	10	45	45	30.60	3552
	18000	273.09	0.549	757.04	274.59	1.653	19	100	80	56.06	5986
60	9000	463.45	-0.222	7200.00	*462.42*	1.450	9	47	47	22.75	3296
	18000	382.52	0.251	638.10	383.48	2.542	17	103	85	46.61	5564
75	9000	568.57	0.026	1437.20	586.72	1.840	8	47	47	21.01	3270
	18000	486.15	0.084	1275.20	486.56	3.213	19	97	96	42.54	6350
90	9000	692.00	0.062	323.09	692.43	3.432	11	49	49	18.86	3042
	18000	609.48	0.000	1999.15	**609.48**	4.180	20	97	97	37.40	6334
100	9000	782.69	0.055	760.00	783.12	3.962	11	49	49	17.07	3042
	18000	700.17	0.013	2654.00	700.26	4.586	20	97	97	33.82	6334
Avg			0.187	1531.748		2.533					

get a very good upper bound efficiently.

Tables 2 and 3 also show that when the operation duration is longer, the number of stations visited is larger. Moreover, we observe that the number of stations visited is much less than the number of stations in the network, because the vehicle has a limited capacity and the repositioning time is not long enough to allow all stations to be visited.

Because our problem is different from Raviv et al. (2013) in the sense that we explicit define pick-up and drop-off stations and do not consider the operational cost in the objective function, their results may differ from ours shown in Tables 2 and 3. In principle, without rounding errors due to CPLEX and the weight of the total operational cost relative to the total penalty cost equal to zero, the optimal shortage of their problem obtained could be better than that of ours, because our solution set is smaller as a result of predetermining pick-up and drop-off stations. Meanwhile, as in their results, the total number of bikes dropped off at all of the visited stations can be larger than that picked up at these stations. This implies that some bikes that are dropped off at some visited stations are actually picked up from the depot.

Tables 4-11 present the results from the computational experiments conducted on the second set of instances. LB and UB stand for the lower and upper bounds obtained from CPLEX, respectively. CPLEX could not find the optimal solution to any of the 104 instances from the second set. Feasibility was achieved for less than half of the instances within the 2-hour limit. The UBs are usually not too good compared to the results obtained from our heuristic which produces solutions with better quality in more than half of the cases (where feasibility was achieved by CPLEX) using much less computing time. The overall improvement of the results obtained by our heuristic over the UBs from CPLEX (for the 45 instances where feasibility was achieved) is 0.456%.

As can be seen in these tables, the heuristic is very fast and produces solutions with high quality. For the short planning horizon instances, the average gap (from the LBs) is 0.687%, and for the longer horizon instances, the average gap is 1.261%. The overall average deviation is 0.974%. According to the summary table 12, it seems that the heuristic is better in planning vehicle routes and determining the pick-up/drop-off quantities for the instances of short planning horizon. The results can be further improved by running the heuristic longer (e.g., $\beta = 30$, Table 13) where

Table 4: Results of the second set of instances, $T = 9000$ and $k = 10$

$\lvert\mathcal{N}_0\rvert$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
100	765.689	0.843	772.91	-0.092	7200	772.20	0.759	14	53	53	19.29	2610
125	1021.255	0.643	1034.20	-0.613	7200	1027.86	1.444	15	55	55	15.15	2400
150	1247.729	0.545	1257.66	-0.246	7200	1254.57	1.438	18	55	55	12.66	2374
175	1406.761	0.711	1430.66	-0.967	7200	1416.83	1.986	16	58	58	11.65	2016
200	1629.402	0.697	-	-	7200	1640.84	2.334	17	57	57	10.51	2136
225	1885.456	0.624	1948.84	-2.645	7200	1897.30	2.989	19	56	56	9.16	2276
250	2106.665	0.817	-	-	7200	2124.02	3.281	17	57	57	8.11	2150
275	2270.393	0.685	-	-	7200	2286.06	3.764	16	57	57	7.84	2144
300	2493.339	0.785	-	-	7200	2513.06	3.703	18	55	55	7.20	2382
325	2752.142	0.920	-	-	7200	2777.69	4.195	16	52	52	6.15	2750
350	2974.340	0.732	-	-	7200	2996.27	3.577	16	55	55	5.97	2394
375	3138.892	0.705	-	-	7200	3161.19	6.195	18	56	56	5.74	2264
400	3375.275	0.659	-	-	7200	3397.68	8.186	15	55	55	5.38	2390
Avg		0.721					3.373					

Table 5: Results of the second set of instances, $T = 18000$ and $k = 10$

$\lvert\mathcal{N}_0\rvert$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
100	677.236	1.582	688.07	0.007	7200	688.12	1.250	30	106	106	36.28	5268
125	922.087	1.974	932.67	0.857	7200	940.66	1.675	32	108	107	28.27	5014
150	1142.561	1.113	1158.77	-0.289	7200	1155.42	2.213	33	111	111	25.03	4662
175	1300.503	1.108	-	-	7200	1315.08	3.330	31	112	112	23.26	4558
200	1520.752	1.006	1542.33	-0.397	7200	1536.21	2.737	36	113	113	20.99	4422
225	1773.529	1.241	1833.10	-2.034	7200	1795.81	3.917	32	112	112	17.91	4538
250	1995.193	1.061	2134.16	-5.509	7200	2016.59	6.995	37	112	112	16.39	4542
275	2153.474	1.129	-	-	7200	2178.06	5.947	35	116	116	15.71	4076
300	2376.437	1.408	-	-	7200	2410.37	5.899	31	108	108	14.08	5034
325	2631.309	1.210	-	-	7200	2663.54	9.975	37	111	111	13.02	4652
350	2852.320	1.334	-	-	7200	2890.89	6.619	31	109	109	11.83	4884
375	3014.210	1.397	-	-	7200	3056.93	7.876	32	109	109	11.31	4910
400	3249.340	1.156	-	-	7200	3287.34	10.052	32	111	111	10.91	4672
Avg		1.286					5.268					

the overall deviation is reduced to 0.87%.

As expected, the runtime of the heuristic increases with the size of the network and repositioning time because the solution space is larger. However, the runtime does not decrease with vehicle capacity as shown in Table 12. The runtime decreases with vehicle capacity but increases with it at the end. It may be because an increase in vehicle capacity reduces the chance of having infeasible solutions and this is beneficial only if vehicle capacity is small. However, when vehicle capacity is larger, the feasible solution space is also larger. Hence, a further increase in vehicle capacity leads to a longer computation time. It may be further explained by the fact that the heuristic does not terminate after a fixed number of iterations, but rather it terminates after K consecutive iterations without improvement. Hence, the number of iterations needed by the

Table 6: Results of the second set of instances, $T = 9000$ and $k = 20$

$\vert\mathcal{N}_0\vert$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
100	757.675	0.845	762.19	0.255	7200	764.13	0.476	10	56	56	20.91	2268
125	1015.481	0.708	1020.08	0.259	7200	1022.72	0.968	15	58	58	15.93	2036
150	1241.588	0.581	1251.86	-0.240	7200	1248.85	1.249	12	56	56	13.37	2262
175	1398.796	0.780	1415.29	-0.389	7200	1409.79	1.557	12	58	58	12.45	1994
200	1620.869	0.853	1651.72	-1.024	7200	1634.81	1.791	9	57	57	11.11	2150
225	1878.321	0.748	1908.03	-0.816	7200	1892.47	2.865	11	58	58	9.58	2032
250	2103.136	0.587	-	-	7200	2115.55	2.708	14	59	59	8.77	1912
275	2266.508	0.630	-	-	7200	2280.87	2.652	14	59	59	8.22	1902
300	2488.187	0.710	-	-	7200	2505.99	3.948	15	60	60	7.67	1772
325	2745.796	0.636	-	-	7200	2763.36	3.315	14	60	60	7.01	1800
350	2970.251	0.744	-	-	7200	2992.52	2.871	12	56	56	6.18	2274
375	3134.805	0.481	-	-	7200	3149.95	4.851	11	61	61	6.34	1678
400	3370.077	0.687	-	-	7200	3393.38	3.642	11	56	56	5.59	2264
Avg		0.691					2.530					

Table 7: Results of the second set of instances, the second set of instances, $T = 18000$ and $k = 20$

$\vert\mathcal{N}_0\vert$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
100	657.489	1.501	670.73	-0.480	7200	667.51	0.961	23	114	114	40.44	4300
125	905.537	2.006	919.31	0.518	7200	924.07	1.273	24	111	111	30.76	4680
150	1128.336	1.286	1144.23	-0.104	7200	1143.04	2.797	29	114	114	26.58	4294
175	1285.336	1.198	1307.70	-0.518	7200	1300.92	1.570	27	115	115	24.88	4198
200	1505.106	1.187	1527.85	-0.306	7200	1523.18	2.877	22	119	119	22.30	3682
225	1759.620	1.102	1845.61	-3.597	7200	1779.22	2.443	25	117	117	19.34	3952
250	1979.037	1.047	-	-	7200	1999.98	3.872	24	121	121	17.67	3474
275	2143.107	1.140	-	-	7200	2167.81	4.134	24	118	118	16.46	3840
300	2363.469	1.251	-	-	7200	2393.41	6.962	27	116	116	15.22	4078
325	2618.046	1.003	-	-	7200	2644.56	6.417	28	121	121	14.17	3468
350	2840.673	0.991	-	-	7200	2869.10	4.612	28	120	120	13.05	3598
375	3003.741	0.807	-	-	7200	3028.17	6.594	24	124	124	12.85	3120
400	3237.359	0.734	-	-	7200	3261.30	5.554	26	126	126	12.21	2852
Avg		1.173					3.851					

heuristic may vary from instance to instance which results in different computing times.

Table 14 shows how *imbalanced* each instance of the second set is. Ideal denotes the objective value to an instance at its *balanced* state, while Initial denotes the objective value to an instance before repositioning takes place. Every instance is highly imbalanced initially, and according to Tables 4-11, only a fraction of the imbalance can be covered. In particular, the larger the network is, the smaller fraction of the imbalance can be covered. This is because under the restrictions in vehicle capacity and operating hours, the vehicle can only visit very limited stations to reduce their shortages. Hence, even if the network is larger, the number of stations visited and the total number of bikes picked up are about the same. Hence, the fraction of the imbalance to

Table 8: Results of the second set of instances, $T = 9000$ and $k = 30$

$\|\mathcal{N}_0\|$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
100	755.698	0.384	757.55	0.140	7200	758.61	0.671	13	60	60	22.02	1788
125	1012.685	0.641	1017.36	0.183	7200	1019.22	0.919	11	60	60	16.45	1790
150	1239.010	0.493	1244.57	0.047	7200	1245.15	1.167	13	59	59	13.83	1904
175	1397.390	0.854	1409.70	-0.020	7200	1409.42	1.275	12	59	59	12.49	1894
200	1621.292	0.930	1636.25	0.016	7200	1636.51	1.925	12	57	57	10.94	2156
225	1876.554	0.741	1897.06	-0.342	7200	1890.57	1.873	12	58	58	9.74	2030
250	2102.415	0.675	2158.91	-1.955	7200	2116.71	2.176	13	59	59	8.68	1878
275	2263.941	0.775	-	-	7200	2281.62	2.006	13	59	59	8.17	1916
300	2486.033	0.631	-	-	7200	2501.83	5.239	12	63	63	7.95	1424
325	2743.667	0.709	-	-	7200	2763.25	3.111	12	60	60	7.02	1798
350	2968.320	0.581	-	-	7200	2985.68	4.230	10	60	60	6.56	1790
375	3133.110	0.599	-	-	7200	3151.99	4.301	13	60	60	6.23	1796
400	3368.221	0.449	-	-	7200	3383.40	4.334	13	62	62	6.09	1506
Avg		0.651					2.256					

Table 9: Results of the second set of instances, $T = 18000$ and $k = 30$

$\|\mathcal{N}_0\|$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
100	650.735	1.964	662.06	0.258	7200	663.77	0.767	23	113	113	41.20	4418
125	901.479	1.824	916.74	0.163	7200	918.23	1.544	20	116	116	31.64	4072
150	1126.147	1.286	1137.96	0.251	7200	1140.82	1.683	24	118	118	26.85	3836
175	1281.509	1.420	1305.91	-0.455	7200	1299.97	1.922	24	114	114	24.98	4316
200	1499.720	1.266	-	-	7200	1518.95	1.875	24	120	120	22.72	3600
225	1754.574	1.366	1787.79	-0.498	7200	1778.88	2.978	25	116	116	19.37	4078
250	1978.997	0.900	-	-	7200	1996.96	5.150	27	121	121	17.91	3478
275	2137.996	1.233	-	-	7200	2164.69	4.580	28	119	119	16.69	3694
300	2358.116	0.986	-	-	7200	2381.61	4.837	25	124	124	16.01	3118
325	2613.583	0.881	-	-	7200	2636.80	5.280	27	125	125	14.63	2996
350	2835.486	0.955	-	-	7200	2862.84	6.257	27	124	124	13.39	3110
375	2998.819	0.869	-	-	7200	3025.11	6.015	28	125	125	13.01	2988
400	3234.315	0.754	-	-	7200	3258.89	4.778	26	127	127	12.33	2752
Avg		1.208					3.667					

be covered is smaller for larger networks. Tables 4-11 also show that when the operating hours (or the vehicle capacity) is reduced, the fraction of the imbalance to be covered is decreased because the number of stations visited (or the number of bikes picked up or dropped off at visited stations) is reduced in general.

Regarding the third set of instances, CPLEX could not find the optimal solution to any of the 30 instances. Feasibility was achieved for four of the instances within the 2-hour limit of CPU time (see Tables 15-16). Our heuristic managed to find better bounds than those provided by CPLEX. The average gaps (from LBs) are 2.927% and 3.878% for short planning horizon instances and long planning horizon instances, respectively. These gaps are further reduced to 2.668% and 3.43% when the heuristic runs a little longer (e.g., $\beta = 30$).

Table 10: Results of the second set of instances, $T = 9000$ and $k = 40$

$\lvert\mathcal{N}_0\rvert$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
100	755.229	0.668	755.88	0.586	7200	760.31	0.713	11	57	57	21.68	2148
125	1011.117	0.944	1019.15	0.157	7200	1020.75	0.918	13	60	60	16.22	1796
150	1237.474	0.619	1244.79	0.031	7200	1245.18	1.654	12	60	60	13.83	1798
175	1394.505	0.772	1423.98	-1.308	7200	1405.35	2.015	14	60	60	12.96	1794
200	1619.833	0.722	-	-	7200	1631.62	2.000	10	59	59	11.43	1898
225	1876.497	0.621	-	-	7200	1888.23	2.583	10	60	60	9.94	1786
250	2099.115	1.099	-	-	7200	2122.44	3.114	17	55	55	8.24	2370
275	2263.230	0.602	-	-	7200	2276.93	2.357	12	60	60	8.51	1794
300	2486.304	0.627	-	-	7200	2502.00	3.136	11	61	61	7.94	1662
325	2743.351	0.689	-	-	7200	2762.38	2.603	12	62	62	7.07	1542
350	2968.040	0.594	-	-	7200	2985.77	5.541	11	60	60	6.55	1798
375	3131.912	0.595	-	-	7200	3150.66	4.780	13	60	60	6.30	1792
400	3368.217	0.377	-	-	7200	3380.95	7.955	13	64	64	6.22	1302
Avg		0.687					3.028					

Table 11: Results of the second set of instances, $T = 18000$ and $k = 40$

$\lvert\mathcal{N}_0\rvert$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
100	648.455	2.237	661.68	0.243	7200	663.29	0.987	24	115	115	41.30	4188
125	899.603	2.602	913.45	1.116	7200	923.64	1.267	25	114	114	30.83	4320
150	1124.078	1.434	1138.24	0.192	7200	1140.43	1.776	27	115	115	26.90	4172
175	1279.517	1.533	1296.87	0.198	7200	1299.44	2.003	27	118	118	25.05	3816
200	1499.273	1.241	1521.71	-0.237	7200	1518.11	1.923	29	125	125	22.81	2998
225	1753.076	1.279	1792.08	-0.909	7200	1775.79	3.521	25	119	119	19.64	3698
250	1977.671	1.418	-	-	7200	2006.12	4.657	25	116	116	17.20	4064
275	2136.965	1.348	-	-	7200	2166.17	4.098	27	116	116	16.58	4074
300	2358.169	1.023	-	-	7200	2382.55	5.316	25	125	125	15.94	2994
325	2612.708	0.885	-	-	7200	2636.03	7.554	26	126	126	14.68	2864
350	2835.190	1.150	-	-	7200	2868.17	7.901	23	123	123	13.10	3238
375	2998.582	0.996	-	-	7200	3028.76	5.721	27	123	123	12.82	3224
400	3233.586	0.772	-	-	7200	3258.75	7.149	24	125	125	12.34	2980
Avg		1.378					4.144					

5 Conclusion

In this paper, we present an iterated tabu search heuristic for the static bike repositioning problem. The implemented heuristic makes use of specialized neighborhood structures and perturbation to obtain diversification in the search. Experiments were conducted on 156 instances. Computational results show that this simple iterated tabu search heuristic produces high quality solutions using very short computing times.

The contributions of this paper to the literature include the following. 1) This paper proposes a modification to the existing bike reposition problem to improve its realism and to reduce the

Table 12: Summary of the results of the second set of instances, $\beta = 5$

T	k	Avg gap	Avg CPU
9000	10	0.721	3.373
9000	20	0.691	2.530
9000	30	0.651	2.556
9000	40	0.687	3.028
18000	10	1.286	5.268
18000	20	1.173	3.851
18000	30	1.208	3.667
18000	40	1.378	4.144

Table 13: Summary of the results of the second set of instances, $\beta = 30$

T	k	Avg gap	Avg CPU
9000	10	0.672	16.219
9000	20	0.660	13.505
9000	30	0.584	12.645
9000	40	0.644	12.443
18000	10	1.068	22.716
18000	20	1.079	16.487
18000	30	1.128	15.469
18000	40	1.125	15.865

Table 14: Initial and optimal objective values of the second set of instances

$\|\mathcal{N}_0\|$	Ideal	Initial
100	372.95	867.63
125	463.60	1128.64
150	554.70	1355.99
175	642.69	1518.88
200	748.06	1745.64
225	843.91	2003.53
250	931.94	2229.28
275	1020.99	2393.74
300	1128.29	2620.45
325	1219.47	2879.75
350	1306.56	3103.54
375	1397.85	3268.56
400	1509.45	3505.00

solution space. 2) This paper develops an efficient and novel heuristic to obtain high quality solutions to solve the proposed problem. 3) The specific operators developed can also be incorporated in other heuristics to solve the same repositioning problem or other repositioning problems with different objective functions.

In the future, we extend our method to solve the static repositioning problem with multiple

Table 15: Results of the third set of instances, $T = 9000$ and $k = 20$

$\|\mathcal{N}_0\|$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
200	4705.242	2.590	5027.60	-3.924	7200	4830.33	1.996	19	46	46	24.18	3476
200	4627.208	5.379	-	-	7200	4890.28	1.861	15	45	45	24.17	3570
200	4569.509	1.599	-	-	7200	4643.78	2.152	18	52	52	25.70	2758
200	4060.484	3.006	-	-	7200	4186.33	2.075	22	49	49	26.78	3082
200	4644.023	3.182	-	-	7200	4796.63	2.028	17	48	48	25.15	3202
300	6708.367	3.167	-	-	7200	6927.74	2.520	21	47	47	17.58	3338
300	7689.757	2.926	-	-	7200	7921.54	3.820	19	51	51	17.63	2864
300	7059.826	3.455	-	-	7200	7312.46	3.731	19	53	53	19.65	2636
300	7896.429	2.899	-	-	7200	8132.14	3.626	16	47	47	17.89	3348
300	7810.430	3.312	-	-	7200	8077.99	3.548	18	46	46	17.68	3464
400	9867.310	2.469	-	-	7200	10117.10	5.263	16	51	51	16.36	2870
400	9464.528	2.616	-	-	7200	9718.80	4.450	17	54	54	17.01	2504
400	10450.917	2.491	-	-	7200	10717.90	5.785	24	48	48	15.50	3228
400	11064.190	1.938	-	-	7200	11282.90	6.584	22	50	50	14.45	2992
400	9294.085	2.876	-	-	7200	9569.30	5.460	15	49	49	16.88	3114
Avg		2.927					3.660					

Table 16: Results of the third set of instances, $T = 18000$ and $k = 20$

$\|\mathcal{N}_0\|$	**LB**	**Gap**	**UB**	**Gap**	**CPU**	**Heuristic**	**CPU**	η	ψ	φ	**Done**	ς
200	3878.686	2.793	4501.27	-11.355	7200	3990.14	2.418	40	101	101	43.65	5830
200	3755.538	4.299	-	-	7200	3924.25	2.992	36	100	100	46.02	5994
200	3806.262	3.289	3943.67	-0.202	7200	3935.71	2.964	41	108	108	42.74	5034
200	3346.358	4.498	-	-	7200	3503.97	3.166	47	98	98	45.79	6192
200	3784.330	4.204	4401.57	-10.250	7200	3950.41	4.617	37	104	104	44.70	5516
300	5789.674	4.088	-	-	7200	6036.42	5.532	45	103	103	33.57	5636
300	6641.535	4.118	-	-	7200	6926.79	5.327	43	109	109	32.30	4920
300	6001.242	4.527	-	-	7200	6285.83	5.092	39	104	104	36.22	5510
300	6754.097	3.647	-	-	7200	7009.76	3.342	42	110	110	33.78	4792
300	6657.507	4.434	-	-	7200	6966.41	5.121	39	102	102	33.61	5740
400	8773.665	3.187	-	-	7200	9062.52	7.726	43	104	104	28.92	5504
400	8336.122	4.224	-	-	7200	8703.79	11.255	42	104	104	30.11	5520
400	9237.014	3.412	-	-	7200	9563.27	8.909	40	101	101	28.70	5862
400	9806.238	4.129	-	-	7200	10228.60	8.175	53	98	98	25.64	6228
400	8144.318	3.324	-	-	7200	8424.37	5.840	39	109	109	31.75	4918
Avg		3.878					5.498					

vehicles. We will also develop an efficient heuristic that allows a station to be visited more than once.

Acknowledgment

This work was partially supported by NordForsk under grant 25900 and by a grant from National Natural Science Foundation of China (71271183). This support is gratefully acknowledged. The authors are also grateful to the three reviewers for their constructive comments.

References

Benchimol, M., Benchimol, P., Chappert, B., de la Taille, A., Laroche, F., Meunier, F. and Robinet, L. (2011). Balancing the stations of a self service "bike hire" system, *RAIRO-Operations Research* **45**(1): 37–61.

Borgnat, P., Abry, P., Flandrin, P., Robardet, C., Rouquier, J.-B. and Fleury, E. (2011). Shared bicycles in a city: A signal processing and data analysis perspective, *Advances in Complex Systems* **14**(3): 415–438.

Borgnat, P., Abry, P., Flandrin, P. and Rouquier, J.-B. (2009). Studying Lyon's Vélo'v: A statistical cyclic model, *Proceedings of the European Conference on Complex Systems (ECCS)*, Complex System Society.

Caggiani, L. and Ottomanelli, M. (2012). A modular soft computing based method for vehicles repositioning in bike-sharing systems, *Procedia - Social and Behavioral Sciences* **54**: 675–684.

Chemla, D., Meunier, F. and Wolfler Calvo, R. (2013). Bike sharing systems: Solving the static rebalancing problem, *Discrete Optimization* **10**(2): 120–146.

Contardo, C., Morency, C. and Rousseau, L.-M. (2012). Balancing a dynamic public bike-sharing system, *Technical Report CIRRELT-2012-09*, Montréal.

Dell'Amico, M., Hadjicostantinou, E., Iori, M. and Novellani, S. (2014). The bike sharing rebalancing problem: Mathematical formulations and benchmark instances, *Omega* **45**: 7–19.

dell'Olio, L., Ibeas, A. and Moura, J. L. (2011). Implementing bike-sharing systems, *Proceedings of the ICE - Municipal Engineer* **164**(2): 89–101.

Di Gaspero, L., Rendl, A. and Urli, T. (2013). A hybrid ACO+CP for balancing bicycle sharing systems, *in* M. J. Blesa Aguilera, C. Blum, P. Festa, A. Roli and M. Sampels (eds), *Hybrid Metaheuristics 8th International Workshop*, Vol. 7919 of *Lecture Notes of Computer Science*, Springer, pp. 198–212.

Erdoğan, G., Laporte, G. and Wolfler Calvo, R. (2013). The one commodity pickup and delivery traveling salesman problem with demand intervals, *Technical Report CIRRELT-2013-46*, Montréal.

Froehlich, J., Neumann, J. and Oliver, N. (2008). Measuring the pulse of the city through shared bicycle programs, *Proceedings of International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems* pp. 16–20.

Gendreau, M., Hertz, A. and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem, *Operations Research* **40**(6): 1086–1094.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence, *Computers & Operations Research* **13**(5): 533–549.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004a). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery, *Discrete Applied Mathematics* **145**(1): 126–139.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004b). Heuristics for the one-commodity pickup-and-delivery traveling salesman problem, *Transportation Science* **38**(2): 245–255.

Ho, S. C. and Gendreau, M. (2006). Path relinking for the vehicle routing problem, *Journal of Heuristics* **12**(1-2): 55–72.

Kaltenbrunner, A., Meza, R., Grivolla, J., Codina, J. and Banchs, R. (2010). Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system, *Pervasive and Mobile Computing* **6**(4): 455 – 466.

Kitthamkesorn, S., Chen, A., Xu, X. and Ryu, S. (2013). Modeling mode and route similarities in network equilibrium problem with go-green modes, *Networks and Spatial Economics* . In press.

Larsen, J. (2013). Bike-sharing programs hit the streets in over 500 cities worldwide, `http://www.earth-policy.org/plan_b_updates/2013/update112`.

Lathia, N., Ahmed, S. and Capra, L. (2012). Measuring the impact of opening the London shared bicycle scheme to casual users, *Transportation Research Part C* **22**: 88 – 102.

Lin, J. H. and Chou, T. C. (2012). A geo-aware and VRP-based public bicycle redistribution system, *International Journal of Vehicular Technology* **2012**: 14 pages.

Lin, J.-R. and Yang, T.-H. (2011). Strategic design of public bicycle sharing systems with service level constraints, *Transportation Research Part E* **47**(2): 284–294.

Lin, J.-R., Yang, T.-H. and Chang, Y.-C. (2013). A hub location inventory model for bicycle sharing system design: Formulation and solution, *Computers & Industrial Engineering* **65**(1): 77–86.

Lin, S. (1965). Computer solutions of the traveling salesman problem, *Bell System Technical Journal* **44**: 2245–2269.

Lu, C.-C. (2013). Robust multi-period fleet allocation models for bike-sharing systems, *Networks and Spatial Economics* . In press.

Martinez, L. M., Caetano, L., Eiró, T. and Cruz, F. (2012). An optimisation algorithm to establish the location of stations of a mixed fleet biking system: an application to the city of Lisbon, *Procedia - Social and Behavioral Sciences* **54**: 513–524.

Miller, C. E., Tucker, A. W. and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems, *Journal of the ACM* **7**(4): 326–329.

Nair, R. and Miller-Hooks, E. (2011). Fleet management for vehicle sharing operations, *Transportation Science* **45**(4): 524–540.

Nair, R., Miller-Hooks, E., Hampshire, R. C. and Bušić, A. (2013). Large-scale vehicle sharing systems: Analysis of Vélib', *International Journal of Sustainable Transportation* **7**(1): 85–106.

Papazek, P., Raidl, G. R., Rainer-Harbach, M. and Hu, B. (2013). A PILOT/VND/GRASP hybrid for the static balancing of public bicycle sharing systems, *in* R. Moreno-Díaz, F. Pichler and A. Quesada-Arencibia (eds), *Computer Aided Systems Theory - EUROCAST 2013*, Vol. 8111 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 372–379.

Parragh, S. N., Doerner, K. F. and Hartl, R. F. (2008). A survey on pickup and delivery problems, *Journal für Betriebswirtschaft* **58**(2): 81–117.

Rainer-Harbach, M., Papazek, P., Hu, B. and Raidl, G. R. (2013). Balancing bicycle sharing systems: A variable neighborhood search approach, *in* M. Middendorf and C. Blum (eds), *Evolutionary Computation in Combinatorial Optimization*, Vol. 7832 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 121–132.

Raviv, T. and Kolka, O. (2013). Optimal inventory management of a bike-sharing station, *IIE Transactions* **45**(10): 1077–1093.

Raviv, T., Tzur, M. and Forma, I. A. (2013). Static repositioning in a bike-sharing system: models and solution approaches, *EURO Journal on Transportation and Logistics* **2**(3): 187–229.

Romero, J. P., Ibeas, A., Moura, J. L., Benavente, J. and Alonso, B. (2012). A simulation-optimization approach to design efficient systems of bike-sharing, *Procedia - Social and Behavioral Sciences* **54**: 646 – 655.

Sayarshad, H., Tavassoli, S. and Zhao, F. (2012). A multi-periodic optimization formulation for bike planning and bike utilization, *Applied Mathematical Modelling* **36**(10): 4944–4951.

Schuijbroek, J., Hampshire, R. and van Hoeve, W.-J. (2013). Inventory rebalancing and vehicle routing in bike sharing systems, *Technical report*, Carnegie Mellon University, USA.

Shaheen, S. A., Zhang, H., Martin, E. and Guzman, S. (2011). China's hangzhou public bicycle. understanding early adoption and behavioral, *Transportation Research Record: Journal of the Transportation Research Board* **2247**: 33–41.

Shi, X., Zhao, F. and Gong, Y. (2009). Genetic algorithm for the one-commodity pickup-and-delivery vehicle routing problem, *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS 2009)*, Vol. 1, pp. 175–179.

Shu, J., Chou, M. C., Liu, Q., Teo, C.-P. and Wang, I.-L. (2013). Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems, *Operations Research* **61**(6): 1346–1359.

Szeto, W. Y., Wu, Y. and Ho, S. C. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem, *European Journal of Operational Research* **215**(1): 126–135.

Ting, C.-K. and Liao, X.-L. (2013). The selective pickup and delivery problem: Formulation and a memetic algorithm, *International Journal of Production Economics* **141**(1): 199–211.

Vogel, P., Greiser, T. and Mattfeld, D. C. (2011). Understanding bike-sharing systems using data mining: Exploring activity patterns, *Procedia - Social and Behavioral Sciences* **20**(0): 514 – 523.