# Poster: Vehicle Routing with Pickup and Delivery: A Greedy Approach

Turash Mosharraf, Jie Wu, and Huanyang Zheng
Department of Computer and Information Sciences, Temple University
turash.mosharraf,jiewu,huanyang.zheng@temple.edu

## ABSTRACT

Recently, bicycles have become popular in large cities and many bicycle sharing companies have established their own bike sharing systems. Extra vehicles are used to transport bikes among different bike stations, such that the number of bikes in each bike station can be balanced. This paper studies a bike rebalancing schedule that minimizes the total vehicle shipping distances. Starting with a minimum length path covering all bike stations, the proposed algorithm iteratively updates its configuration, whenever the vehicle capacity constraint is violated. Finally, experiments demonstrate the efficiency and effectiveness of the proposed algorithm.

## CCS CONCEPTS

• **Networks** → Cyber-physical networks;

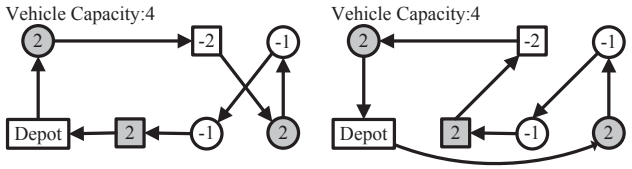
## KEYWORDS

Vehicle routing; bike sharing systems; pickup and delivery.

## 1 INTRODUCTION

Nowadays sustainable transportation systems have earned increasing attention. Sharing transportation systems, such as car-pooling, car-sharing, and bike-sharing, have been implemented in many industrialized countries. Accessibility and affordability promote a short-term bike rental systems as win-wins for just about anyone willing to ditch his car for a bike. The development of bike sharing systems (BSSs) is due not only to the sustainability of the system, but also because of its door-to-door feature. In general, BSSs are based on a bike fleet, bike stations, and a number of users. Customers arrive at rental stations of a company (e.g., Citi Bike), utilize bicycles for some amount of time, and then return the bicycles to the same station or to a different one run by the same company. The success of a BSS, from a strategic planning point of view, depends on the system design in terms of number, location, and capacity of stations as well as on the consistency of the available bike fleets.

The greater diffusion and usage of the BSS is limited by its own drawbacks. The BSS is mainly used for medium-short distances and for one-way trips, leading to to an unbalanced distribution of the bikes over time and space. Therefore, in order to increase the system capacity and user satisfaction, it is necessary to properly relocate bikes among stations of the BSS. Relocation is performed based on a desired inventory level (determined by historical data) for each stations. A significant number of works provide highly accurate prediction for desired inventory level. Some researchers propose determining a maximum and minimum inventory level to ensure more flexibility [2]. However, after obtaining the desired inventory, it is important to optimally route the rebalancing vehicles to reduce gas usage and service time while meeting the need of every station. The rebalancing operation can be performed during work time or break time. For simplicity, most researchers assume that the number of incoming and outgoing bikes during the rebalancing period is negligible. Fig. 1(a) shows a simple scenario with one company. The grey stations have more than enough bicycles, and they need to be transferred to the white stations with shortages. The amount of shortage or surplus is shown along with the stations. A feasible path for a rebalancing vehicle with capacity 4 is depicted in the figure.




(a) Single Company (b) Multiple Companies

**Figure 1: Vehicle paths to rebalance bikes**

In real life, there are often multiple companies exist in a system, as shown in Fig. 1(b). This figure has the same set of stations as in Fig. 1(a), but it has two companies. The circle and square denote the stations of these two companies, respectively. Note that the feasible path in Fig. 1(a) is no longer feasible in Fig. 1(b) because of the additional constraint that each bike must be collected and returned to the same company. This constraint adds another level of complexity to the vehicle routing problem, and the resulting tour may be more costly than the single company case. We found several existing solutions that perform very efficiently in the single company scenario. However, very few of them can be effectively scaled for the multiple companies scenario. Our work can be considered as a step to fill this gap.


*CHANTS'17, , October 20, 2017, Snowbird, UT, USA*

ACM ISBN ISBN 978-1-4503-5144-7/17/10.
https://doi.org/10.1145/3124087.3124101

## 2 PROBLEM FORMULATION

The BSS is modeled as a graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes (bicycle stations) and $E = \{(v_i, v_j) \,|\, v_i, v_j \in V\}$ is the set of edges (roads connecting bicycle stations). Each station $v_i$ has a demand $q_i$ and belongs to a company $t_i$. Here, $q_i$ can be either positive or negative. We mainly consider two companies, i.e., $t_i$ can be either 0 or 1. Bicycles of one company cannot be kept in the stations of the other company. The weight of the edge from $v_i$ to $v_j$ is denoted as $d_{ij}$, representing the distance between two bike stations. A single vehicle with capacity $Q$ is used to transport bikes among all bike stations for all companies. The objective is to find a Hamiltonian tour among all bike stations to minimize the total bike shipping cost, in terms of distances. The constraint is to satisfy the demand of each bike station and the capacity of the vehicle.

A binary decision variable, $x_{ij}$, is used to indicate whether a vehicle travels directly from node $v_i$ to $v_j$. Let $\theta_{ij}^t$ denote the number of bicycles of company $t$ carried by the vehicle from $v_i$ to $v_j$. The problem is formulated as follows:

$$\min \quad \sum\nolimits_i \sum\nolimits_j d_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum\nolimits_i x_{ij} = 1, \ \forall j \text{ and } \sum\nolimits_j x_{ij} = 1, \ \forall i, j \tag{1}$$

$$\sum_{i \in U, j \notin U} x_{ij} \geq 1, \text{ and } \sum_{i \notin U, j \in U} x_{ij} \geq 1, \ \forall U \subseteq V \tag{2}$$

$$\sum\nolimits_j \theta_{ij}^t - \sum\nolimits_k \theta_{ki}^t = q_i, \forall\, i, j, k, t = t_i \tag{3}$$

$$\theta_{ij}^t \geq 0, \forall t \text{ and } \sum\nolimits_t \theta_{ij}^t \leq Q \cdot x_{ij} \tag{4}$$

$$x_{ij} \in \{0, 1\} \tag{5}$$

Constraint (1) ensures that each vehicle visits a station exactly once. Constraint (2) ensures that the tour becomes a disconnected cycle covering all the stations rather than be partitioned into different subtours. Constraint (3) ensures the demand of each bike station. Constraint (4) is the capacity constraint that ensures that the balance of bicycles from each company is non-negative and that the total balance does not exceed the vehicle capacity at any instance. Constraint (5) means that the decision variable $x_{ij}$ is binary. Our problem is NP-hard by a simple reduction from [3]. Therefore, we use a heuristic approach to provide a reasonably good solution.

## 3 METHODOLOGY

### 3.1 Idea and Algorithm

We start with the definition of the feasibility:

*Definition 3.1.* A path is feasible if constraints (3)-(5) can hold for each station on the path. A set of stations is feasible if there is a feasible path that covers this set of stations.

Our idea is to gradually build paths from a set of starting bike stations. We start with an approximate tour for the TSP, considering all the bike stations but ignoring the vehicle capacity constraint. This is because the more closely the TSP tour is followed, the smaller the total distance is [5]. We have one more definition:

**Algorithm 1** ALG1

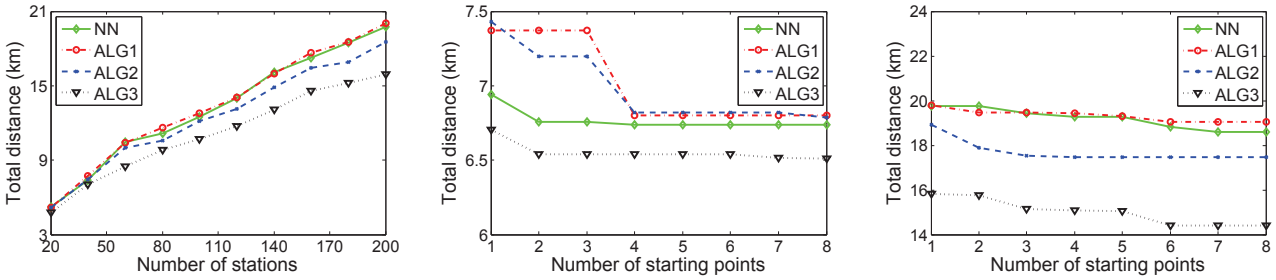
**Input:** set of stations $V = \{v_1, v_2, ..., v_n\}$, their geographic locations and their demands, vehicle capacity $Q$, and number of random start $k$

**Output:** a feasible path $p^*$

1: Create edges between all pairs of stations (edge weight is the corresponding geographic distance)
2: Find approximate TSP path $T$ without considering demand, using Christofides' algorithm [1]
3: Initialize $S \leftarrow$ random subset of size $k$ from $V$
4: **for** each starting bike station $s \in S$ **do**
5: set $p \leftarrow \emptyset$, set $s' \leftarrow$ the last biking station on $T$ starting from $s$ without violating constraints (3)-(5)
6: **for** each bike station $v_i$ on $T$ from $s$ to $s'$ **do**
7: add $v_i$ to $p$, update $\theta \leftarrow \theta + q_i$ and $T \leftarrow T \setminus \{v_i\}$
8: **while** $T \neq \emptyset$ **do**
9: $s''$ and $s'''$ $\leftarrow$ the forwarding path from $s''$ to $s'''$ is feasible and longest among all feasible ones, if $p$ goes to $s''$ as the next station
10: **for** each bike station $v_i$ on $T$ from $s''$ to $s'''$ **do**
11: add $v_i$ to $p$, update $\theta \leftarrow \theta + q_i$ and $T \leftarrow T \setminus \{v_i\}$
12: Update $p^* \leftarrow p$ if $p^*$ has a smaller path length than $p$
13: **return** $p^*$ as a feasible path

*Definition 3.2.* Given a TSP tour $T$, a current state $s$, and a station $v$, a forwarding path is a path that starts at $v$ and follows $T$ as long as it does not violate constraints (3)-(5). If visiting $v$ from current state $s$ results in an infeasible path, then no forwarding path exists for $s$.

We greedily favor the bike station with the longest forwarding path. The forwarding path is followed until the capacity constraint is violated. In this case, the current path becomes fixed and the remaining capacity of the vehicle is updated. Consequently, next starting point is searched with respect to the longest forwarding path under the capacity constraint. Algorithm 1 (or ALG1 for short) is proposed for the scenario, in which all bike stations belong to a single company. In line 1, ALG1 determines the geographic distances among bike stations. In line 2, an approximate TSP tour $T$ using Christofides' algorithm [1] is computed. In line 3, $k$ random starting points are selected and are checked later. A feasible path is obtained for each of these starting points, and the final solution is the shortest one of these feasible paths (lines 4 to 12). In lines 5 to 7, a feasible path is initialized. Given the starting point, the vehicle will go as far as possible under the capacity constraint. In lines 8 to 11, forwarding paths are searched iteratively. In each iteration, ALG1 greedily finds the forwarding path that is feasible and longest of all the feasible paths in line 9. In lines 10 to 12, the forwarding path above is used as the next station for the vehicle. A feasible solution is reached when the fixed path contains all stations. Once a solution is found for each of the starting stations in $S$, the solutions are compared and the best path is stored in line 12. Finally, the resulting path is returned in line 13.




(a) Impact of number of stations (b) Starting points for 40 stations (c) Starting points for 200 stations

**Figure 2: Performance comparison for single company case**

The time complexity is $O(n^3)$. The initial TSP tour takes $O(n^3)$. After that, there can be at most $O(n)$ iterations. At each iteration, at most $O(n)$ forwarding lengths are computed, and each computation can take at most $O(n)$. Therefore, the total complexity is $O(n^3)$ and is the same as the TSP.

### 3.2 Algorithm Variations

ALG1 has two variations using other greedy metrics. The first variation is ALG2. Instead of using the longest feasible path in line 9, ALG2 uses the feasible path that minimizes the ratio of (i) the distance from the current station to the furthest station in this path to (ii) the length of this path. The second variation is ALG3, which uses the feasible path that minimizes the ratio of (i) the distance from the current station to the starting station of this path to (ii) the length of this path. Moreover, ALG1, ALG2, and ALG3 can be extended to scenarios with multiple companies. Note that bicycles of one company cannot be kept in stations of the other company, but the vehicle can carry bicycles of different companies under its capacity constraint. Our key observation is that the forwarding path does not interact with the number of companies. As a result, we only need to modify lines 5 and 9 for multiple companies.

## 4 EXPERIMENTS

### 4.1 Experimental Setup

The experiments are performed on both real and simulated datasets. We chose 200 bike stations of Citi Bike and Cycle Central Park located at New York city. The demands are placed following a normal distribution from [10, -10]. Each vehicle can carry 40 bicycles. We generate the demand of each stations using normal distribution. We ran experiments for both small and large instances of the problem by selecting different subsets of stations. We run two types of experiments. First, we vary the number of stations from 20 to 200. Second, we vary the number of starting points from 1 to 8. Nearest Neighbor (NN) [4] is the baseline.

### 4.2 Distance and stations

The results are shown in Fig. 2(a). As we increase the number of stations, the performance of our algorithms improves. ALG3 has the best performance. This is because ALG3 considers both the forwarding lengths and the distance traversed to achieve that forwarding. Therefore, ALG3 provides a better estimation of per unit distance than other algorithms. For example, if any station has a large forwarding length but is very far away from the current position, choosing this station as the next starting point will involve a large rewiring distance. This will, in turn, hurt the overall performance.

### 4.3 Distance and starting points

The results are shown in Figs. 2(b) and 2(c) for small and large data, respectively. For any number of stations, using multiple starting points gives a better solution. However, the rate of performance improvement decreases and eventually becomes saturated. Therefore, a large number of starting points is not necessary to get a reasonably good solution. When the number of stations is as small as 40, NN sometimes outperforms ALG1. However, for, reasonably large number of stations (like 200), ALG1, ALG2, and ALG3 significantly outperform NN. Among the extensions, ALG3 shows the best performance due to its greedy metric.

## 5 CONCLUSION

This paper presents a greedy algorithm to rebalance bikes in bike-sharing systems. The proposed algorithm is applicable to a variety of other related problems, such as the airline crew scheduling problem. Experiments demonstrate the efficiency and effectiveness of the proposed algorithm.

## REFERENCES

[1] Hyung-Chan An, Robert Kleinberg, and David B Shmoys. 2015. Improving Christofides' algorithm for the S-T path TSP. *J. ACM* 62, 5 (2015), 34.

[2] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. 2012. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media.

[3] Hipólito Hernández-Pérez and Juan-José Salazar-González. 2004. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics* 145, 1 (2004), 126–139.

[4] Hipólito Hernández-Pérez and Juan-José Salazar-González. 2007. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks* 50, 4 (2007), 258–272.

[5] Ning Wang and Jie Wu. 2016. Opportunistic wifi offloading in a vehicular environment: Waiting or downloading now?. In *IEEE INFOCOM*. 1–9.