A Greedy Approach for Vehicle Routing when Rebalancing Bike Sharing Systems

Yubin Duan, Jie Wu and Huanyang Zheng
Department of Computer and Information Sciences, Temple University, USA
Email: {yubin.duan, jiewu, huanyang.zheng}@temple.edu

***Abstract*—With the bloom of the sharing economy, bike sharing systems have earned increasing attention, and a great amount of bike sharing systems have been established in major cities. Users of these systems mainly conduct one-way trips, which leads to an unbalanced distribution of the bikes over time and space. The system operators could hire a fleet of vehicles to move bikes among bike stations for rebalancing. We focus on a routing schedule problem for each vehicle used in the rebalancing process and aims to minimize its moving distance. For the problem, we propose a greedy algorithm which can be easily extended to a parallel version. The scheduled route for a vehicle is adapted from the Hamiltonian path covering all unbalanced bike stations. The algorithm greedily adjusts the route if the vehicle cannot moving along the Hamiltonian path due to capacity limitation violation. Different from previous approaches, our algorithm has a more flexible tradeoff between running time consumption and optimality of the output. Finally, we conduct experiments on both real-world and synthetic datasets and compare the performance of our algorithm with a classic approach.**

***Index Terms*—vehicle routing, capacitated traveling salesman problem, bike sharing systems, greedy algorithms.**

I. Introduction

Recently, bike sharing systems (BSSs) have been widely adopted in major cities [1]. Research shows that applying bike sharing systems not only brings benefits to the environment, but also to the economy. Accessibility and affordability of BSSs greatly motivate users to ditch their cars for bikes. According to [2], an average of 40% of bike share members in 4 North American cities drive less after joining the scheme. In addition, as a solution to the "first/last mile" connectivity issues brings BSSs huge potential economic benefits. Besides these advantages, the social benefits which are brought from the development of BSSs also include benefits for health, transport flexibility, and cost savings for individuals. Despite that the profits brought by BSSs is great and attractive, maintaining the sustainability of the system is challenging.

The development of BSSs is strongly relay on the sustainability of the system. However, users of BSS mainly conduct one-way trips, which may bring the bike unbalance problem to the system. The unbalance distribution of bikes may further generate out-of-service stations. Specifically, the BSSs are mainly used for one-way trips and this mode may lead to an unbalanced distribution of the bikes [3], especially in rush hours. The amount of full and empty stations may increase under the extreme distribution. Consequently, it leads to an higher probability of failure to find an available dock when a user returning a bike or to find an available bike when renting.

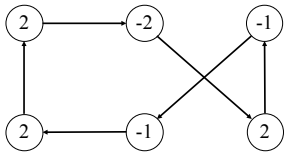


Fig. 1. A rebalance senario.

To avoid such kind of failure, operators need to rebalance the bike distribution among bike stations by deploying a fleet of trucks to avoid the full or empty inventory. Researches have shown that the cost of rebalancing contributes to a large part in operation cost for bike sharing companies. Therefore, it is important to optimize the rebalancing cost.

One way to optimize the rebalancing cost is to minimize the total transport distance of the fleet when rebalancing the bikes. The rebalancing problem mainly contains two challenges. The first is to estimate the inventory demands for each station. A significant number of researchers has provided several usage prediction methods, such as [4–6]. The other challenge occurs afte the desired inventory is determined. It is also important to optimally schedule routes for vehicles in the fleet to reduce their travel distance while meeting the demand of every station. The limited capacities of vehicles bring difficulties to the optimization. In this paper, we mainly focus on the second challenge and propose a heuristic greedy approach to construct a route that can minimize the transport distance for each vehicle used in the rebalancing.

We assume that a BSS is divided into several regions and each region contains one truck for rebalancing based on the fact that number of researches have studied region division methods, such as [4, 7]. Therefore, we focus on scheduling a route for the single vehicle used in a region. Given the location and the rebalance demands of each bike station, our vehicle routing problem can be seen as an CTSPPD problem, which has been proven to be a NP-hard problem [8]. Specifically, our goal is to minimize the tour distance for each vehicle used for rebalancing under several constraints. First of all, the number of bikes on the vehicle cannot exceed the limited capacity. Also, the demands of bike station shall be satisfied after rebalancing operation. In addition, the vehicle needs to depart from a station and end at the same station as for the consistence of the rebalancing operation. Fig. 1 shows a simple scenario of our problem. Each vertex in the figure represents a bike station, and the numbers labeled on the

stations show the demands. The positive number means the station has more than enough bicycles, and the extra bikes need to be transferred to other stations. The negative number means the station faces shortages. Arrows in Fig. 1 show a feasible path for a vehicle whose capacity limitation is 4.

The main contributions of the paper are summarized as follows:

- We summarize one classic algorithm [9] which can be used in our vehicle routing problem, and analyze the weakness of this approach.
- We propose a new greedy approach, and show that our greedy algorithm can be easily extend to parallel version.
- We provide the properties of our greedy algorithm including the complexity and feasibility analysis.
- We simulate our algorithm on both real-world and synthetic datasets, and compare the performance between our algorithm with the previous approach.

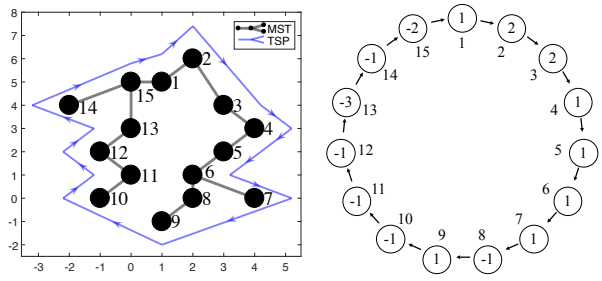
The remainder of the paper is organized as follows. Section II discusses related works and compares them with our approach. Section III presents a formal mathematical model of the problem. Section IV describes a classic algorithm to our problem and analyzes its drawbacks. Section V presents our greedy algorithm, the algorithm variations and its properties. Section VI shows the performance of our algorithm in both real-world and synthetic dataset. Finally, section VII concludes the paper.

II. Related Work

Hiring fleets of trucks to rebalance the BSS can be classified as a Capacitated Vehicle Routing with Pickup and Delivery problem (CVRPD). The vehicle routing problem is a classic problem that has been studied for more than thirty years [10–13]. Parragh et al. [14] surveyed the variations of Vehicle Routing with Pickup and Delivery problems. In our problem, we consider that the service area of a BSS can be divided into multiple regions and each region is assigned one vehicle to conduct the bike rebalancing. Several possible division methods were introduced in [4, 7], and we focus on reviewing works related with the single vehicle routing.

The TSP with Pickup and Delivery (TSPPD) proposed in [11] is similar to our problem. In the TSPPD, a vehicle with a limited capacity is hired to pick up and deliver products. The difference is that the products picked up by the vehicle must be transport to the depot and cannot be sent to users who need the delivery. Moshiev et al. [11] introduced applications of TSPPD and developed a heuristic algorithm that considers to conduct pick-ups and deliveries along the TSP tour. Anily and Mosheiov [13] introduced an algorithm with 2 approximation ratio by using the minimum spanning tree. Gendreau et al. in [15] proposed two heuristics, where the one utilizes the exact solution of a special case and the other one takes advantage of the tabu search.

Hernandez et al. [10] introduced the One-commodity Pickup-and-Delivery TSP (1-PDTSP), where a single vehicle needs to pickup products from suppliers and deliver to requesters, and proposed a branch-and-bound algorithm to find



(a) Locations & TSP tour (b) Unbalance situation

Fig. 2. The locations of stations and unbalance situations. The black vertices and edges in (a) represent the minimum spanning tree used to calculate the approximate TSP tour (shown by the blue tour in (a)). The number on each vertex in (b) illustrate the demands of each station.

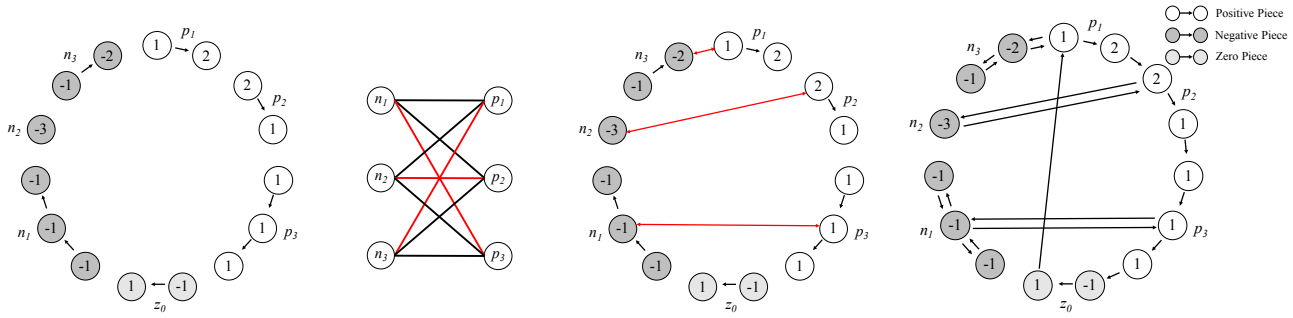
an optimal solution. Although they considers the capability limitation of the vehicle, they adds additional constraint that each pickup or delivery locations can be visited only once. However, we show that there may be no feasible solutions under the additional constraint in section V.

In BSS rebalancing operations, deciding a reasonable target level of each station is also important, since the vehicle routing cannot be determined without a specific target inventory for each station. Raviv et al. [16] proposed an appraoch to minimize the expected system-wide user dissatisfaction under a time limit. Schuijbroek et al. [7] proposed to determine a range of desirable inventory instead of a fixed value. This allows much more edibility in the routing step. To the best of our knowledge, they are the first to introduce a dual-bounded service level constraints. It increases the flexibility of the rebalance operation. As for the cost of rebalancing, although we consider to use transportation distance to quantify the rebalancing cost, Lin et el. [17] proposed an approach that takes road regulations, traffic as well as geographical factors into consideration.

III. Problem Formulation

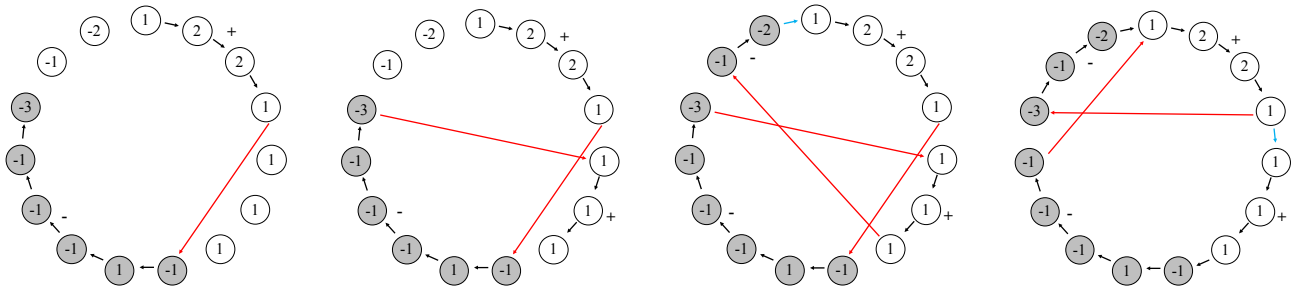
The rebalancing problem is modeled in a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set denoting bicycle stations and $E = \{(v_i, v_j)|v_i, v_j \in V\}$ is the edge set denoting roads between bicycle stations. Each station v_i is associated with a demand $d_i \in \mathbb{Z}$. A positive d_i means the station i has extra bikes, and a negative d_i means the station is lack of bikes. The weight of the edge (v_i, v_j) is denoted as w_{ij} , which represents the distance between bike stations v_i and v_j. We consider that a vehicle with capacity C is hired to move bikes among unbalanced stations to satisfy the demand for each station. The objective is to minimize total bike shipping distances for the vehicle under constraints: the bike demand of each station should be satisfied, and the route for the vehicle should be feasible. The definition of a feasible path and a feasible route is given as follows.

***Definition** 1:* Given a path T, a vehicle capacity C, the path T is a feasible path if the number of bikes on the vehicle (denoted as n) always satisfies such equation: $0 \leq n \leq C$. The path T is a feasible route if T contains all stations.



(a) Partition (b) The bipartite graph (c) Matching result between pieces (pairs are linked by red lines) (d) The final constructed route

Fig. 3. (a), (b), (c) and (d) illustrate the procedures to construct the route for rebalancing in the classic algorithm. The white, grey and light grey pieces in the figure means the positive-pieces, negative-pieces and zero-pieces correspondingly. See section IV for details.



(a) Jumping to a negative part (b) Jumping to a positive part (c) An admissible route (d) Another admissible route

Fig. 4. (a), (b) and (c) illustrate the procedures to construct the route for rebalancing in our greedy algorithm. The red arrows indicates the direction of jumps. (d) shows the route generated by choosing a different starting station. See section V for details.

Strictly, our problem can be abstracted by a binary optimization problem. Let x_{ij} denote the binary decision variable. The variable equals to 1 if the vehicle directly travels from station v_i to v_j. Let θ_{ij} denote the number of bikes loaded on the vehicle when it is moving from v_i to v_j.

The problem is formulated as follows:

$$\min \sum_i \sum_j w_{ij} x_{ij}$$

$$\text{s.t.} \sum_{i \in U, j \notin U} x_{ij} \geq 1, \text{ and } \sum_{i \notin U, j \in U} x_{ij} \geq 1, \ \forall U \subseteq V \quad (1)$$

$$\sum_j \theta_{ij} - \sum_k \theta_{ki} = d_i, \forall i, j, k \quad (2)$$

$$\theta_{ij} \geq 0, \text{ and } \theta_{ij} \leq C \cdot x_{ij} \quad (3)$$

$$x_{ij} \in \{0, 1\} \quad (4)$$

Constraint (1) ensures that the route for a vehicle covers all the stations and constructs a cycle, rather than a combination of subtours. Constraint (2) ensures that the rebalance operation can fulfill the bike demand required by each station. Constraint (3) is the capacity constraint. It ensures that the amount of bikes loaded on the vehicle won't exceed its capacity. Constraint (4) ensures the decision variable x_{ij} is binary.

IV. A Classic Algorithm

In this section, we briefly introduce the classic algorithm proposed in [9] and illustrate the drawbacks of the algorithm by using a special example.

The general idea of the classic algorithm is to modify a TSP tour to satisfy the capacity constraint in CVRPD problem. To present how the algorithm works, we go through it on an example shown in Fig. 2. In this example, the capacity of the vehicle used for rebalance is 6, i.e. $C = 6$. Fig. 2(a) shows the locations of each bike station and the TSP tour founded by [18]. Fig. 2(b) presents the unbalance situation of each station. The positive integer in the nodes means the number of extra bikes in the station and the negative inverter shows the shortness of bikes in the corresponding station. The number next to the nodes indicates the label of the node corresponding to Fig. 2(b).

The procedure of the classic algorithm is shown as Fig. 3. The algorithm first divides the TSP route into 3 different kinds of pieces, which are positive-pieces, negative-pieces, and zero-pieces (denoted by $+$, $-$, 0 in Fig. 3 respectively). To divide the TSP path, the classic algorithm randomly choose a starting point (which is not the actual starting point of the rebalancing route) and suppose there is a vehicle with amount of $C/2$ bikes loaded at the chosen point. Then, let the imaginary vehicle transport the bikes between each station along the clockwise (or counterclockwise) TSP route. When the amount of bikes loaded on the vehicle equals to 0 or $C/2$ or C, the edge from the current station to the next station will be deleted, and pieces are generated. If the number of bikes increased (decreased or unchanged) along the piece, then the piece is

denoted as positive-piece (negative-piece or zero-piece).

After deciding the pieces, the classic algorithm then constructs a bipartite graph (shown as Fig. 3(c)) and computes the minimum-weight perfect matching. Specifically, vertices in the bipartite graph are divided into two sets, representing positive-pieces and negative-pieces respectively. The edge between each pair of vertices in different sets (i.e. a pair of a positive-piece and a negative-piece) is weighted by the shortest distance between two pieces. The shortest distance is found out by comparing distance between each pair of stations between positive and negative-piece, since each piece can contain more than one actual bike stations. After constructing the bipartite graph, the minimum bipartite matching algorithm is used to pair the vertices in the two sets. Correspondingly, the positive-pieces and negative-pieces are matched as shown in Fig. 3(c).

After the minimum matching is found, the route for the vehicle can be construct by the following procedure. Traverse in clockwise/counterclockwise direction from the randomly chosen starting point. When encountering a zero-piece, the vehicle should move along the piece and perform pickups and deliveries. When encountering the first piece of a matched positive-negative pair, the vehicle should service the pair as following sequence: suppose the vehicle encounter a positive-piece first, and it is matched with a negative-piece by edge e. The vehicle should service the positive-piece (traverse and perform pickups and deliveries), until e is encountered. Then, move to the negative-piece along e and service the negative-piece. Then move back to the point in the positive-piece where the vehicle left off and continue. When encountering the second piece of a positive-negative pair, the vehicle should not conduct pickups and deliveries when traversing the piece. It is because that they have already been served. Although the starting point used here is chosen randomly, and the imagery vehicle contains $C/2$ bikes initially, [9] has proved that a feasible starting point always can be founded on the tour constructed for a vehicle with no bikes loaded.

We summarize the drawbacks of the classic algorithm as follows. In the partition phase of the classic algorithm, the TSP tour is cut into small pieces. When a vehicle passing each piece, the maximum fluctuate of goods loaded in the vehicle will not expired $C/2$ instead of C. In their algorithm, $C/2$ is used as the threshold to ensure the feasibility of the constructed route. However, compare with using C as threshold, this strategy is more likely to generate shorter pieces in terms of distance. That is to say, it may generate more pieces, and a larger number of pieces may lead to a larger number of jumps in the constructed route. Then it may lead to a longer route. In addition, the classic algorithm is based on the bipartite matching algorithm, which is not easy to parallelize.

V. Algorithmic Design

A. A Greedy Approach

Inspired by the classic algorithm, we proposed a new greedy algorithm and several variations. The core idea of the classic algorithm remains. We also try to modify the TSP tour to fit capacity limitation and force the vehicle to visit stations facing shortage (surplus) after the vehicle's load increased (decreased). We first introduce Length Greedy Algorithm (or LGA for short) to present our greedy approach. In line 1, LGA constructs the approximate TSP tour T by using Christofides' algorithm [18]. Then, the involved parameters are initialized in line 2. From line 3 to 5, LGA initializes a feasible path. Specifically, based on the starting point chosen in line 2, the vehicle goes along the TSP tour as far as possible until the capacity constraint is not satisfied. In line 7, LGA greedily chooses the path which is the longest feasible path of all potentially feasible paths as the forwarding path. From line 8 to 9, the algorithm adds the forwarding path founded in line 7 into the vehicle's path set p. The path is extended at each iteration. A feasible path for balancing all bike stations in V is generated if all stations are included in the path p (or T turns to empty). Finally, the solution is returned in line 10.

Algorithm 1 LGA

Input: Set of stations $V = \{v_1, v_2, ..., v_n\}$ along with geographic locations and demands, vehicle's capacity C.
Output: A feasible route p.

1: $T \leftarrow$ the approximate TSP path of V without considering rebalance demand.
2: $p \leftarrow \emptyset$, randomly choose $s \in V$ as starting station.
3: $s' \leftarrow$ the last bike station such that constraints (2)-(3) are not violated following the path from s to it in T.
4: **for** each station v_i in the path from s to s' in T **do**
5: Insert v_i into p, and remove v_i from T
6: **while** $T \neq \emptyset$ **do**
7: Update s and $s' \leftarrow$ stations s and s' such that the path from station s to s' in T is the longest feasible path.
8: **for** each station v_i in the path from s to s' in T **do**
9: Insert v_i into p, and remove v_i from T
10: **return** p as the feasible route

Using the same example shown in Fig. 2, we illustrate the procedures of our algorithm in Fig. 4. Specifically, We choose the station 1 as the starting station and the clockwise direction as default detection. Beginning from station 1, it can proceed to station 4 at most, since there is no room left to load the bike in station 5. Therefore, it has to jump to another station, which is chosen greedily. That is to say, the station for next jump should guarantee that the vehicle can proceed along the TSP tour as long as possible. That certain station in our example is station 8 and the first jump from station 5 to 8 is shown by the red line in Fig. 4(a). From that station, the vehicle can proceed to station 13 until there is no bike to unload. Using the same greedy strategy, the next jump chosen for our vehicle to jump to is station 5 (shown in Fig. 4(b)). Proceeding form station 5, the vehicle can gain 3 bikes, therefore the path from station 5 to station 7 can be denoted as a positive part of TSP. At station 7, the vehicle can continue moving to station 8 without violate the capacity constraint. However, since station 8 is

Algorithm 2 LGA with Multi-starts (LGA-k)

Input: Set of stations $V = \{v_1, v_2, ..., v_n\}$ along with geographic locations and demands, number of starting stations k, and vehicle's capacity C

Output: A feasible route p^*

1: The same as step 1 of LGA.
2: Set $p \leftarrow \emptyset$, $S \leftarrow$ randomly choose k stations in V
3: **for** each station $s \in S$ as the starting station **do**
4: The same as step 3-9 of LGA.
5: Update $p^* \leftarrow p$ if the path length of p is smaller than p^*.
6: **return** p^* as a feasible route.

already visited and the demand is also satisfied, the vehicle will bypass this station. It is the same situation for stations 9 to 13. As the result, the vehicle will directly jump to station 14 (bypass stations 8-13), and supply stations 14 and 15, as shown in Fig. 4(c). After leaving station 15, the demands of all stations are satisfied. Therefore, the vehicle will back to our starting station, and the rebalance task is finished. The total distance of the route is 29. Apparently, it is shorter than the route constructed by the classic algorithm since the number of jumps is smaller.

The starting bike station is critical for the algorithm's performance. Different starting bike stations can lead to different final route lengths. This can be shown by comparing the routes in Fig. 4(c) and Fig. 4(d). That is to say, using a set of independent starting points may improve the overall performance. This leads to the multi-start version of our greedy algorithm, and it is shown in Algorithm 2.

B. Algorithm Variations

In aforementioned greedy approach, the longest feasible path is used as the greedy criterion. That is to say, we want to keep the TSP route as long as possible. However, it may not be able to give the optimal result, since the length of jumping from a station to the next station is not counted. Therefore, to take the jumping cost into account, we proposed another different greedy metrics as variations:

Variation1: Let l_f denote the length of path starting form the jump-out station to the last station in the forwarding path. Let l_p denote the length of the forwarding path. Rather than choosing the longest path, we can choose the feasible path which minimizes the l_f/l_p as the forwarding path.

Variation2: We can also choose the forwarding path which minimizes l_n/l_p, where l_n denotes the length of the path starting from the jump-out station to the first station in the forwarding path and l_p still means the length of this path.

C. Properties

The time complexity of LGA is $O(n^3)$. The calculation of the TSP tour costs $O(n^3)$ time. In the following iteration, at least one node will be chosen and removed from T. Therefore, LGA can conduct $O(n)$ iterations at most. During each iteration, at most $O(n)$ stations can be checked as potential forwarding stations, and each check costs $O(n)$ time at most. Thus, the time complexity of LGA is $O(n^3)$ which is also the time complexity of the classic approach. Although LGA share the same time complexity with the classic algorithm, our algorithm can be easily parallelized.

Besides discussing the time complexity, we also pay attention to the feasibility of our algorithm. If feasible routes exist, our approach can always find out as least one feasible route by adding all station into the starting station set. Our algorithm can always proceed to a forwarding station and add/drop one bike after each iteration, unless the feasible path is not exist. Because the total number of bikes that need to be moved is certain and finite, our algorithm can always construct a feasible route if all stations are tested as the starting station. That is to say, the feasibility of our algorithm is determined by the existence of feasible route.

In our problem, each station can be visited more than once. Under this condition, at least one feasible route exists if the sum of demands among all stations is 0. To construct the path, a naive approach is to use the vehicle pickup one bike in a supplement station and head to a station with negative demands right after the pickup. A feasible route can always be constructed since the sum of extra bikes in each supplement station is equals to the sum of bikes needed in each demand station. Admittedly, the path constructed by the naive approach may contain unnecessary detours, however, it shows the existence of feasible route.

As an extension to our problem, we can constrain that each station can be visited only once, which means the entire demand of each station has to be satisfied when the station is visited. We denote this scenario as non-split scenario. After adding this constraint to our problem, there may exist no feasible route even if the sum of demands equals to 0. It can be easily proven by constructing an example in which the feasible route doesn't exist. Assume there are 5 bike stations that need to be rebalanced. The demand of each station is 6, 6, 6, -10, and -8 respectively. If there is only one vehicle with a capacity of 10 that can be used in rebalancing operation, no feasible route can be found. More specifically, the vehicle has to visit a station which has 6 extra bikes to load some bikes as supplement; but after 6 bikes are loaded, the vehicle has no where to go. It cannot go to another station to unload bikes, since 6 bikes in the vehicle cannot satisfy the needing either -10 or -8. It also cannot go to another station to load more bikes, since the vehicle does not have enough capacity to load all the bikes in other supply stations. Therefore, additional constraint has to be added to guarantee the existence of feasible route, and it will be shown in Theorem 1.

***Theorem** 1:* When only one vehicle with capacity C can be used in rebalancing operation under the non-split scenario, at least one feasible path exists if the absolute value of each station's demand dose not exceed $C/2$.

Proof: To prove that the feasible route always exists, we prove that the infeasible scenario will not appear. The infeasible scenario means that after visiting some stations, the vehicle cannot go any further. More specifically, let C denote the capacity of the vehicle, n denote the number of bikes on

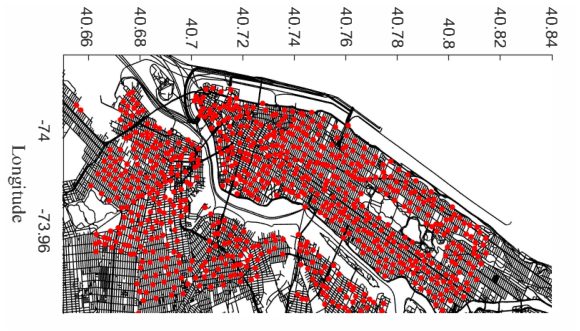


Fig. 5. Locations of NYC Citi Bike stations.

the vehicle, and d_i denote the demand of station i, and d_i can be positive or negative.Because of the additional constraint, $\forall\, i \in S\,, |d_i| < C/2$, where S is the set of all stations that has not been visited.

We use proof by contradiction to show that the infeasible scenario will not appear. By assuming that the infeasible scenario exists, we can conclude that $\forall i \in S$:

$$n + d_i > C \text{ if } d_i > 0,$$
$$n + d_i < 0 \text{ if } d_i < 0.$$

From the first and the second equation, we can conclude that $n > C - d_i > C - C/2 = C/2$ and $n < -d_i < C/2$, respectively. Since there is the contradiction, our hypothesis is not true. That is to say, the infeasible scenario will not appear. If the infeasible scenario will not appear, we can alway find at least one feasible route for the vehicle to visit all stations and finish rebalance operation. ■

VI. Experiment

A. NYC Citi Bike Dataset

The Citi Bike is a public bicycle sharing system serving New York City. The company publishes the Citi Bike trip histories and real-time system data online. We use their published data to construct the NYC Citi Bike dataset (or NYC dataset for short) and to test our algorithm. The NYC dataset contains 813 stations along with the longitude and latitude of each station. The distribution of bike stations in Manhattan is shown in Fig. 5. As for the demands of each station, we assume that all stations should share the same number of bikes after rebalancing. Therefore, the demands of each station is determined by the difference between the number of occupied bike docks in a randomly selected time with the average.

B. Synthetic Dataset

Synthetic datasets are used as a supplement of real-world dataset to test the performance of our proposed method and to compare with the classic algorithm. The synthetic datasets simulate different location distribution of bike stations and the unbalance scenario of each station. The station location distribution in all synthetic dataset obeys uniform distribution, but with different parameters. The absolute value of demands for each bike station is generated following a Poisson distribution with parameter $\lambda = 7$. The sign ($+$ or $-$) of each demand is randomly assigned with the same possibility, and our dataset guarantee that the sum of demands is 0. Totally, we create three different kinds of synthetic datasets. In the first dataset, the density of stations in each group is fixed and the number of stations is varied. The efficient area is therefore changed with the number of stations. This aims to simulate the scenario that the BSSs are extended to new areas (e.g. from urban to suburb area) and new stations are established with the same density.In the second dataset, the area of the stations in each group is fixed. The number of stations is changed, which changes the density of the stations. This aims to test the scenario that more stations are added into a certain district. In the third group of dataset, the number of stations is fixed. The density of stations is varied, and the efficient area of stations is therefore changed with density.

C. Experiment Settings

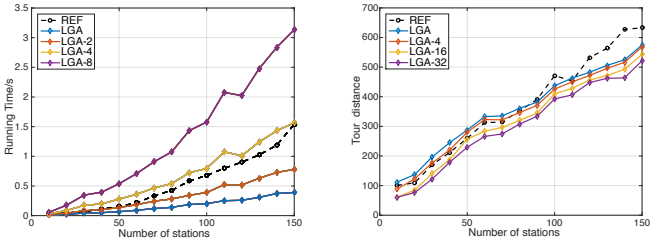
In the following experiment, one vehicle with capacity limit 40 is hired to rebalance the BSS. We first compare the performances in terms of route distance and the running times of our algorithm with the classic algorithm based on the real-world dataset.

Besides the running time and route distance, we also interest in the algorithm's performance in different dataset. Actually, by using different dataset, we are trying to find out which factor influence the algorithm's performance most. Therefore, we test our algorithm and the classic algorithm on all three type of datasets. The final results are obtained by taking the average of results from 50 independent experiments.

D. Evaluation Result

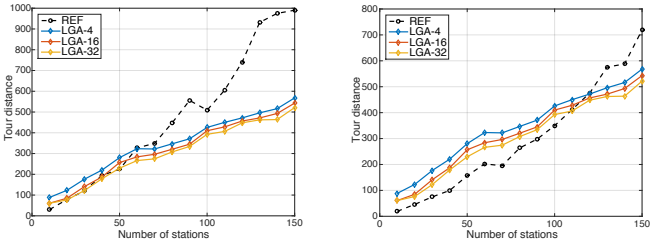
First we illustrate the comparison result of running time in Fig. 6(a). We found that as the number of stations increases, the running time of both our algorithm and the classic algorithm (labels as REF in the figure) increases. By choosing smaller starting station set (choosing 1 or 2 stations), our algorithm is always faster than the classic algorithm. With choosing more than 3 starting stations, our algorithm is slower, but can always generate shorter route as shown in Fig. 6(b).

The performance testing on the three kinds of synthetic dataset is shown in Fig. 7, Fig. 8 and Fig. 9. From Fig. 7, we find out that it is easier for our algorithm to get a shorter route distance in lower density. We found that the average length of the solution is smaller when density increases. This can be explained by the observation that the feasibility constraint becomes more difficult to satisfy as the density increases. Therefore, it is relatively hard to add a feasible next start. Moreover, we found that in Fig. 8 , our algorithm outperform the baseline. The performance of our algorithm improves along with the increase of number of stations. Results are shown in Figs. 7 and 9 shows that the performance gap between our algorithm and the baseline is greater when the number of stations increases. This behavior is expected because selecting certain number of starting station is not



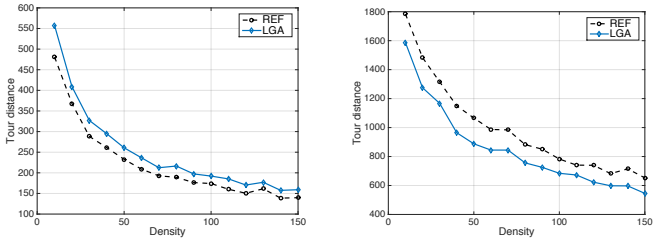
(a) Running time comparison (b) Route distance comparison

Fig. 6. Performance comparison in terms of time consuming and distances.



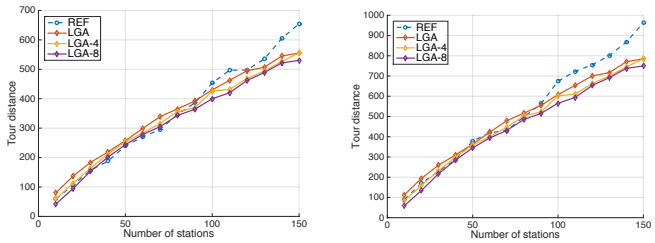
(a) Density = 50 (b) Density = 150

Fig. 7. Performance comparison in different densities.



(a) Number of station = 50 (b) Number of station = 150

Fig. 8. Performance comparison in different number of stations.



(a) Area size = 15 (b) Area size = 30

Fig. 9. Performance comparison in different area sizes.

enough, and based on only the distance of the station from the current starting stations leads to a sub-optimal solution.

VII. Conclusion

This paper presents a greedy algorithm for vehicle routing schedule when rebalancing bike distribution in bike sharing systems. We focus on scheduling a route for each vehicle used in the rebalance precess. Our algorithm first generates a Hamiltonian path that contains all unbalanced bike stations, and then greedily updates the route if the vehicle capacity constraint is violated. The choices of the starting station in the Hamiltonian path is critical for the algorithm's performance, and we extend our algorithm to consider multiple start stations. Besides, two variations on greedy strategies are proposed. Experiments on both real-world and synthetic datasets show that our approach provides the system operator a flexible tradeoff between running time consumption and the optimality. In the future work, we plan to apply the parallel version of our algorithm into real-world scenario and compare the time consuming with existing approaches.

Acknowledgment

This research was supported in part by NSF grants CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and IIP 1439672.

References

[1] E. Fishman, "Bikeshare: A review of recent literature," *Transport Reviews*, vol. 36, no. 1, pp. 92–113, 2016.

[2] S. A. Shaheen, E. W. Martin, A. P. Cohen, N. D. Chan, and M. Pogodzinski, "Public bikesharing in north america during a period of rapid expansion: Understanding business models, industry trends & user impacts," *MTI Report*, pp. 12–29, 2014.

[3] P. DeMaio, "Bike-sharing: History, impacts, models of provision, and future," *Journal of public transportation*, vol. 12, no. 4, p. 3, 2009.

[4] J. Liu, L. Sun, W. Chen, and H. Xiong, "Rebalancing bike sharing systems: A multi-source data smart optimization," in *Proc. of ACM SIGKDD*, 2016, pp. 1005–1014.

[5] L. Chen *et al.*, "Dynamic cluster-based over-demand prediction in bike sharing systems," in *Proc. of ACM Ubicomp*, 2016, pp. 841–852.

[6] Y. Li *et al.*, "Traffic prediction in a bike-sharing system," in *Proc. of ACM SIGSPATIAL*, 2015, p. 33.

[7] J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve, "Inventory rebalancing and vehicle routing in bike sharing systems," 2013.

[8] S. Anily and J. Bramel, "Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries," *Nav. Res. Logist.*, vol. 46, no. 6, pp. 654–670, 1999.

[9] M. Charikar, S. Khuller, and B. Raghavachari, "Algorithms for capacitated vehicle routing," *SIAM Journal on Computing*, vol. 31, no. 3, pp. 665–682, 2001.

[10] H. Hernández-Pérez and J.-J. Salazar-González, "A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery," *Discrete Appl. Math.*, vol. 145, no. 1, pp. 126–139, 2004.

[11] G. Mosheiov, "The travelling salesman problem with pick-up and delivery," *Eur. J. Oper. Res.*, vol. 79, no. 2, pp. 299–310, 1994.

[12] N. Wang and J. Wu, "Trajectory scheduling for timely data report in underwater wireless sensor networks," in *Proc. of IEEE GLOBECOM*. IEEE, 2015, pp. 1–6.

[13] S. Anily and G. Mosheiov, "The traveling salesman problem with delivery and backhauls," *Operations Research Letters*, vol. 16, no. 1, pp. 11–18, 1994.

[14] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems," *Journal für Betriebswirtschaft*, vol. 58, no. 1, pp. 21–51, 2008.

[15] M. Gendreau, G. Laporte, and D. Vigo, "Heuristics for the traveling salesman problem with pickup and delivery," *Computers & Operations Research*, vol. 26, no. 7, pp. 699–714, 1999.

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

[17] J.-H. Lin and T.-C. Chou, "A geo-aware and vrp-based public bicycle redistribution system," *International Journal of Vehicular Technology*, vol. 2012, 2012.

[18] H.-C. An, R. Kleinberg, and D. B. Shmoys, "Improving Christofides' algorithm for the S-T path TSP," *Journal of the ACM*, vol. 62, no. 5, p. 34, 2015.