A Heuristic for Rebalancing Bike Sharing Systems Based on a Deferred Acceptance Algorithm

Mohammed Elhenawy
Virginia Tech Transportation Institute
3500 Transportation Research Plaza
Blacksburg, VA 24061, USA
elhenawy@vt.edu

Hesham Rakha
Virginia Tech Transportation Institute
3500 Transportation Research Plaza
Blacksburg, VA 24061, USA
hrakha@vt.edu

***Abstract*—Many cities around the world are making investments in innovative public transportation infrastructure. Bike-sharing systems (BSS) are one example of this. One key recurring BSS operation problem is the rebalancing of bikes at regular intervals. This redistribution of bikes from full stations to empty stations is required to meet expected demand patterns. Rebalancing entails finding the shortest Hamiltonian cycle in a graph that has the minimal residual. This is an NP-hard problem, and there are a number of algorithms that can be used to find optimal tours, but none are feasible for large instances, since they all grow exponentially. In this paper, we propose a new rebalancing algorithm based on the deferred acceptance algorithm, a well-known game theory algorithm consisting of two phases: tour construction and tour improvement. The proposed algorithm models the rebalancing problem as two disjoint sets of players. Each player in these sets has their own objective function, which is used to build the preference list for players on the other side. Experimental results for the proposed algorithm are promising.**

Keywords—bike-sharing system; rebalancing; game theory; deferred acceptance algorithm.

I. INTRODUCTION

Traffic congestion has become an everyday problem in many urban areas, bringing with it negative environmental effects. During periods of congestion, cars cannot run efficiently, resulting in air pollution, carbon dioxide (CO_2) emissions, and increased fuel use. In 2007, wasted fuel and lost productivity cost Americans $87.2 billion. This number reached $115 billion in 2009 [1]. Congestion also increases travel time. For example, in 1993, driving under congested conditions caused a delay of about 1.2 minutes per kilometer of travel on arterial roads [2]. This congestion problem has only become worse, as evidenced by a report from the Texas Transportation Institute, which indicated that the number of Americans' wasted hours due to traffic congestion increased fivefold between 1982 and 2005.

This problem can be mitigated in urban areas by encouraging more people to use public transportation. However, in big cities, trains and buses generally deliver riders to transit stations near the center of the city, and they then need another transportation mode to get to their final destination. This is referred to as "the last mile problem." Which is defined as "the short distance between home and public transit or transit stations and the workplace, which may be too far to walk."[3] A well-operated bike-sharing system (BSS) can help solve this problem, both relieving roadway congestion and enabling riders to reach their final destination.

Many cities in the US are making investments in BSS. A technical report distributed by the Bureau of Transportation in April 2016 indicated that there are 2,655 BSS stations in 65 US cities, and that 86.3 percent of these stations are connected to another means of scheduled public transportation [4]. These numbers show BSS' potential as a solution for mitigating congestion. However, these systems suffer from a central recurring problem: rebalancing. Rebalancing is a daily problem for operators, who have to find an efficient way to redistribute (i.e., rebalance) bikes from full stations to empty stations to meet expected demand patterns. This redistribution problem is a generalization of the well-known traveling salesman problem (TSP), which involves finding the shortest route passing through each of a collection of locations and then returning to a starting point. This problem was first proposed in [5] as a one-commodity pickup and delivery traveling salesman problem (1-PDTSP). 1-PDTSP is NP-hard, so heuristic optimization techniques are applied to determine a near optimum tour (i.e., route).

Heuristic optimization techniques incorporate stochastic search elements guided by mechanisms that reinforce the search towards promising solutions. They attempt to converge to the optimum solution after a large number of search iterations. However, there is no guarantee that they will find the global optimum, and may instead end up at a local optimum. All heuristic optimization methods have the following main steps [6]: (1) choose an arbitrary initial solution; (2) use some generation rule to iteratively construct new solutions; (3) use the objective function to evaluate these new solutions and report the best solution; and (4) terminate the iterative search when the stopping criterion is satisfied. Simulated annealing (SA), tabu search (TS), genetic algorithms (GAs) and ant colony optimization (ACO) are all heuristic optimization techniques that mimic natural phenomena to solve problems involving very large instances, NP-hard or NP-complete problems of moderate and large sizes, non-analytical models of optimization problems, and problems with uncertainty or robustness issues.

In this paper, we propose an algorithm for solving the static bicycle rebalancing problem (SBRP). The proposed algorithm consists of two stages: tour construction and tour

This work was funded by the Mid-Atlantic Transportation Sustainability Center – Region 3 University Transportation Center (MATS-UTC).

978-1-5090-6484-7/17/$31.00 ©2017 IEEE

improvement. In the tour construction stage, we model the SBRP using two sets of disjointed players. Each player constructs their own set of preferences for the opposite set of players, then the deferred acceptance algorithm is used to find stable matches between the two sets. In the tour improvement stage, a 2-opt algorithm is used to search the neighborhood of the constructed tour to find a better tour. The proposed algorithm has the advantage of being able to easily consider a change in the constraints by restricting each player's preference list changes to the new constraints. Accordingly, this algorithm can be easily used for dynamic rebalancing provided the demand predictions are known. Moreover, the matching algorithm and local search algorithm run in polynomial time, which means the proposed algorithm can solve large instances and is suitable for real-time application.

Following the introduction, this paper is organized into sections. In section II, related work is discussed. Section III briefly presents a background of the methods used in this paper. The problem statement is given in section IV. The proposed algorithm is discussed in Section V, and a toy example is introduced to illustrate the idea. In Section VI, the results of the experimental work are outlined. Finally, the paper concludes, in Section VII.

II. Related work

BSS rebalancing is a crucial system maintenance task, as it impacts customer satisfaction and can result in a significant reduction in customer demand [7]. Due to its importance, BSS rebalancing has been studied extensively in the literature to determine the best way to maintain the number of bikes at each station at a certain level.

In general, rebalancing can be classified as either static, dynamic, or incentivized. In both static and dynamic rebalancing, BSS operators usually use a fleet of trucks to perform the task. Static rebalancing is generally referred to in the literature as the static bicycle repositioning problem (SBRP). The common assumption of SBRP algorithms is that the number of bikes at each station either remains the same or changes slightly, and does not affect the rebalancing outcome. Thus, SBRP is usually done at night or when moving the bikes does not impact the system operations. The dynamic bicycle repositioning problem (DBRP) assumes that moving bikes will have a significant impact on BSS user demand, which will affect the rebalancing outcome. As such, demand predictions have to be input into the algorithm for solving the DBRP so that they are incorporated into the solution. Incentivized rebalancing is based on providing BSS users with incentives to contribute to the system rebalancing. The BSS sends control signals to users suggesting slight changes to their planned journeys, providing them with alternate routes, or offering the option to return bikes for system credit.

Hernández-Pérez and Salazar-González used the branch-and-cut algorithm to solve the 1-PDTSP for problems with up to 75 vertices [5]. In order to solve bigger instances with up to 500 vertices, they proposed two heuristic algorithms [8]. The first heuristic is based on a greedy algorithm and k-optimality criterion, and the second heuristic is based on a branch-and-cut algorithm. Benchimol et al. proposed an integer programming model to solve the SBRP and routing problem with a single vehicle that is allowed to visit the same station more than one time [9]. Rainer-Harbach et al. solved the 1-PDTSP for multiple trucks with different capacities which start and end the redistribution tour at separate locations with no storage space for bikes [10]. The authors decomposed the problem into two simpler problems: routing and pick-up/drop-off planning. They started by establishing the vehicle routing schedule, then used an integer programming model to find the optimal pick-up/drop-off plan associated with that tour. Schuijbroek et al. proposed a real-time and scalable solution to the 1-PDTSP by clustering stations using a maximum spanning star approximation [11]. In this solution, the vehicle is assigned to each cluster and the redistribution tour is constructed to satisfy the required service level. Li et al. used the 2-step approach to solve the rebalancing problem considering multiple types of bicycles [1]. They first used a hybrid generic search to construct the truck tour, then determined the pick-up/drop-off plan using a greedy heuristic algorithm. Ho and Szeto proposed an iterated tabu search heuristic to solve the SBRP [12]. In their formulation of the problem, they assumed that the depot was both a pick-up and drop-off node and that the depot had sufficient bikes and capacity such that the truck could stop at the depot more than once during the rebalancing procedure. Shi et al. used a modified version of a GA to solve the 1-PDTSP for instances with between 20 to 500 vertices [13]. In their proposed GA, they adapted the pheromone idea for crossover and replaced the mutation operator with a local search procedure.

Contardo et al. used the Dantzig-Wolfe and Benders decomposition techniques to find a real-time solution for the DBRP [14] using the most recent demand before the repositioning decisions were made. However, their approach does not work well with rapidly changing demand. Shu et al. used Poisson distribution to predict user demand for the entire planning horizon to find an improved solution for the DBRP. Ghosh et al. [15] addressed the DBRP by considering the change in demand during the day [16]. The authors decomposed the problem into two sub-problems—bike repositioning and vehicle routing—by employing Lagrangian dual decomposition and abstraction mechanisms.

III. Methods

A. Deferred Acceptance Algorithm

This algorithm was first proposed by Gale and Shapley to find a stable assignment of n men and n women where each person has an ordered preference list of the opposite sex for marriage [17]. The algorithm consists of a set of rules that finds a stable assignment, where no couple has an incentive to quit their assigned mate and form a new match.

The algorithm begins with each man proposing to the first woman in his preference list. Then each woman keeps the most interesting proposal based on her preference list and rejects other proposals, if any. At the end of the first round, any woman who received a proposal has a proposal in the deferred acceptance state. The men rejected in the previous round then propose to their second choices. Consequently, each woman keeps the best of the current and deferred offers and rejects any others. This process continues until each man has a deferred proposal. At this point, each of the women

accepts the proposal she holds and the algorithm terminates. The worst case and average complexity of this algorithm are $\mathcal{O}(n^2)$ and $\mathcal{O}(n \log n)$, respectively [18, 19].

B. The 2-opt Local Search Algorithm

The 2-opt algorithm is a tour improving algorithm. Once a solution is established, the 2-opt algorithm can be used to iteratively improve the current solution. The algorithm locally searches the neighborhood by removing two edges from the solution and reconnects the two paths created so that the new solution is valid. The new solution replaces the original if it minimizes the objective function. The process of removing two edges, constructing a new valid solution, and evaluating the new solution continues until no 2-opt improvements can be found. The quality of the solution found by the 2-opt depends on the initial solution constructed by deferred acceptance algorithm. The complexity of the 2-opt algorithm is $\mathcal{O}(N^2)$.

IV. Problem Statement

In order to state the problem, we must first define the term "NotSpot," which is adopted from [20]. A NotSpot is a bike station that is empty or full, rendering it useless to people who want to pick up or return a bike. A bike station that has a number of bikes less than a certain threshold t_{Empty}, or greater than a threshold t_{Full}, is considered a NotSpot as well. The thresholds t_{Empty} and t_{Full} can be estimated for each station from its historical dataset. However, the estimation of these thresholds is beyond the scope of this paper.

Given a fully-connected graph $G = (V, E)$, where V is the depot and the set of N NotSpots in the bike system, and E consists of all the links connecting the vertices in the graph. Each node i in the graph is assigned an integer β_i that equals the number of needed/pick-up bikes. β_i is positive if it represents the number of bikes that need to be taken at a NotSpot, making station i a pick-up station. β_i is negative if it represents the number of needed bikes at a NotSpot, making station i a drop-off station. Each link j is assigned a cost γ_j.

The research question we need to address is, given a list of NotSpots, the distances between each pair of NotSpots, the demand of the NotSpots, and only one truck with maximum capacity Q, what is the optimum tour for the truck to rebalance the NotSpot stations?

The tour is optimum in the sense that the objective function $\sum_{j \in E} \gamma_j + \sum_{i \in V} R_i$ is minimized subject to $\begin{cases} AC_i > 0 \ if \ \beta_i > 0 \\ \mathrm{Q} - AC_i \geq 0 \ if \ \beta_i < 0 \end{cases}$ where R_i, Q, and AC_i are the residual at NotSpot i, the maximum capacity, and the available bike spots on the truck before serving NotSpot i, respectively. This constraint guarantees that no full truck arrives at a pick-up NotSpot station and no empty truck arrives at a drop-off NotSpot station.

Recall that this problem is 1-PDTSP. In the context of the BSS, we assume we have only one type of bike, so it is a one commodity problem. Moreover, we assume that the tour starts from the depot and ends at the depot. In this problem, the truck could start the tour from the depot with any number of bikes less than or equal to its maximum capacity.

V. The Proposed Algorithm

The proposed algorithm consists of two phases. The first phase, which is the main phase, is the tour construction. Tour construction is based on applying the deferred acceptance algorithm N times to match two disjoint sets: the partial tour and the NotSpot stations. The matching is done such that at the end of the construction phase, each constructed tour includes all NotSpot stations and each NotSpot station appears only once in the tour. The second phase uses the 2-opt algorithm to improve the constructed tours and then selects the best tour as the solution to the given problem. In the following subsections, we will describe the two phases in more details.

A. Tour Construction Using the Deferred Acceptance Algorithm

The ultimate goal of the proposed tour construction algorithm is constructing optimized N tours. Each tour consists of N NotSpots in addition to the depot, and is a Hamiltonian path. We modeled the problem of the tour construction as a cooperative game between two disjoint sets of players. The first player's set consists of N partial tours, where each tour is represented by the current load H_{i-1} of the truck after serving the last NotSpot in the partial tour. The objective of each partial tour is finding the next NotSpot i to serve. Thus, each partial tour builds an ascending list based on

$$(\alpha Q - H_i)^2 + \beta * D \quad (1)$$

Where Q is the maximum truck capacity; α is a constant between 0.1 and 1.0 ; H_i is the number of available bikes on the truck after serving NotSpot i; β is a constant $\in [0,1]$; D is the distance in meters between the last station in the tour and NotSpot i.

The partial tour's preference list contains only the NotSpots that are not shown in its current path, and is ordered such that the most preferred NotSpot has the minimum value of Equation (1).

The other player's set consists of N NotSpots. Each NotSpot i builds its preference list such that it lists only partial tours that do not yet include i. The partial tours preference list is ordered such that the tour that minimizes the residual at NotSpot i is at the top. The last partial tour on the list is the tour with the worst residual at NotSpot i. In other words, the top tour in this player's list satisfies the station demand the most and the last tour satisfies the demand the least.

Once the preference list is built for each player, we find a stable match between the partial tours and the NotSpots using the deferred acceptance algorithm. After matching, each partial trip gets expanded by stacking its matched NotSpot to its end. Moreover, the current capacity of the truck assumed to make the tour is updated. Finally, we determine whether or not the partial trip includes all the NotSpots. If all NotSpots are NOT included, then each player's preference list is rebuilt and another game is played. If all NotSpots ARE included, the

algorithm is stopped. The pseudocode of the proposed algorithm is shown in Table I.

B. Tour Improvement Using 2-opt Local Search Algorithm

At the end of the tour construction phase, we will get N different tours, with each tour consisting of N NotSpots. In the tour improvement phase, we evaluate the cost of each of the N constructed tours using Equation (2).

$$\beta * \sum_{j \in E} D_j + \sum_{i \in V} R_i^2 \quad (2)$$

Where, D_j is the length of link j in meters; R_i is the residual at NotSpot i; β is the same constant used in Equation (1).

For each constructed tour, we find all possible 2-opt algorithms and evaluate each. If the best 2-opt tour has a lower cost than its original constructed tour, then the constructed tour is replaced by its 2-opt. After finishing the local search for all N constructed tours, we choose the tour that has the lowest cost as a solution to the rebalancing problem. The pseudocode of the tour improvement algorithm is shown in Table II.

TABLE I. PSEUDOCODE OF THE PROPOSED ALGORITHM FOR TOUR CONSTRUCTION

1. Construct N partial tours, each consisting of the depot and one station of N NotSpot.
2. Update the truck capacity for each partial tour.
3. For $f = 1: N - 1$
   - i. Each NotSpot i builds its preference of partial tour $k \in \{1,2,\dots,N\}$ based on the residual at NotSpot i if it was stacked to partial tour k.
   - ii. Each tour builds its stations preference list using $(\alpha Q - H)^2 + \beta * D$.
   - iii. Run the deferred acceptance algorithm to match the stations with the partial tours.
   - iv. Update the current available bikes on each tour truck after expanding the partial tour.
4. Return N constructed tours, with each tour being a Hamiltonian path.

TABLE II. PSEUDOCODE OF THE PROPOSED ALGORITHM FOR TOUR IMPROVEMENT

1. For each tour constructed by the deferred acceptance algorithm, calculate the total tour cost using (2).
2. For $f = 1: N$
   - i. Find all possible 2-opts of the constructed tour f.
   - ii. For each 2-opt of the constructed tour f, calculate the total tour cost and find the best modified tour.
   - iii. If the best 2-opt tour has less cost than the original tour constructed by the deferred acceptance algorithm, replace the original tour f with its modified 2-opt tour.
3. The solution is best trip out of the N tours.

C. Tour Construction Example

In order to illustrate the proposed algorithm for tour construction, we consider a fully connected undirected graph consisting of three NotSpot stations and depot. For simplicity, we assume the vertices form a square with each side 1,000 meters long, with the exception of the link between $S1$ and $S2$, which has a length of 1,200 meters, as shown in Fig. 1. We set β, Q and α equal to 0.002, 10, and 0.5, respectively. Moreover, we assume the truck has four bikes when it leaves the depot.

The tour construction algorithm consists of the following steps:

1. Construct three partial tours, where each consists of the depot and one of the three NotSpot stations, then update the truck capacity H of each tour as shown in Equation (3).

$$Depot \rightarrow S1 \quad (H_{S1} = 10)$$
$$Depot \rightarrow S2 \quad (H_{S2} = 6) \quad (3)$$
$$Depot \rightarrow S3 \quad (H_{S3} = 0)$$

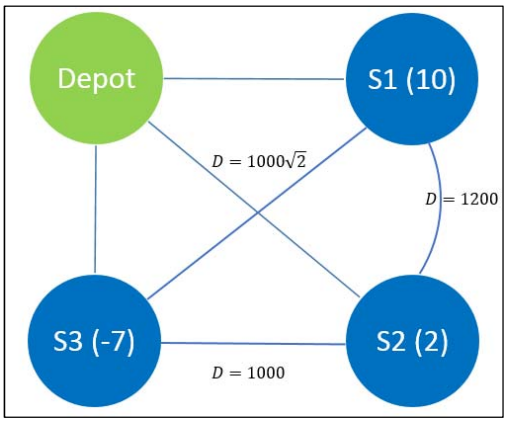

Fig. 1. Toy example graph where the number in parentheses is the NotSpot demand.

2. Set the preference list for each partial tour using Equation (1). The parameter α is set to 0.5 to give preference to pick-up or drop-off stations that bring the truck to its half capacity.
3. Set the preference list for each NotSpot such that the partial tours that minimize the residual at the NotSpot are on the top of the list as shown in Fig. 2.
4. Match the NotSpots with the partial tours using the deferred acceptance algorithm. Based on the matching outcome, the three partial tours are expanded as shown in Fig. 3 and Equation (4).

$$Depot \rightarrow S1 \rightarrow S3$$
$$Depot \rightarrow S2 \rightarrow S1 \quad (4)$$

$$Depot \rightarrow S3 \rightarrow S2$$

5. Update the currently available bikes number for each truck, following the expanded partial tours.
6. Check the number of stations in the partial tour; if the number is less than three, then go to step 2.

The first set of players	The second set of players
Depot→S1 preference list	**S1 preference list**
S3 $(10/2-(10-7))^2+.002*1000\sqrt{2}$	**Depot→S3**$(R_{S1}=0)$
S2 $(10/2-(10))^2+.002*1200$	**Depot→S2**$(R_{S1}=6)$
Depot→S2 preference list	**S2 preference list**
S3 $(10/2-(0))^2+.002*1000$	**Depot→S3**$(R_{S2}=0)$
S1 $(10/2-(10))^2+.002*1200$	**Depot→S1**$(R_{S2}=2)$
Depot→S3 preference list	**S3 preference list**
S2 $(10/2-(2))^2+.002*1000$	**Depot→S1**$(R_{S3}=0)$
S1 $(10/2-(10))^2+.002*1000\sqrt{2}$	**Depot→S2**$(R_{S3}=1)$

Fig. 2. The preference list for the two sets of players.

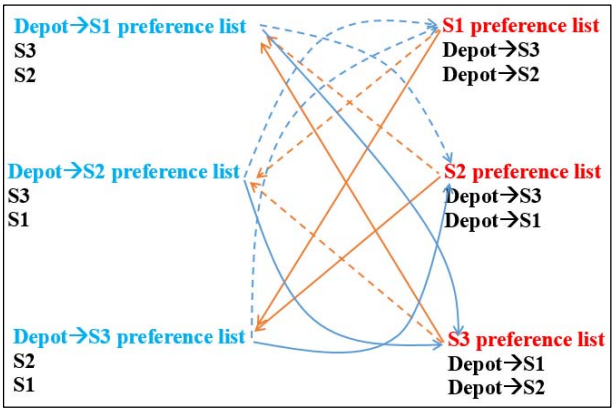

Fig. 3. When the partial tours make offers, the first and second partial tours choose S3 and the third partial tour chooses S2. S3 accepts the first partial tour (S3's first choice) and S2 accepts the second partial tour. In a second stage, the second partial tour makes an offer to S1, which accepts, and the game results in a stable match.

VI. Experimental Work

The proposed algorithm was coded in Matlab 2015b. One advantage of the proposed algorithm is that it has only two hyperparameters: α and β . In the first set of experiments, we set α and β equal to 0.5 and 0.2, respectively. The proposed algorithm was tested using real data collected by Dell'Amico et al. from La Spezia, Italy; Wisconsin, US; and Ottawa, Canada [21]. For the sake of comparison with another heuristic algorithms, we chose the same instances used to test the ACO algorithm proposed in [22]. The ACO algorithm is inspired by the behavior of real ants. Real ants can find the shortest path from the nest to food sources by communicating information among individuals regarding a path by depositing a certain amount of pheromone while walking. Each searching ant probabilistically prefers to follow the path with the highest pheromone levels rather than paths with lower pheromone levels. Readers who are interested in the implementation of the ACO for rebalancing BSSs should consult [22].

Table III shows the comparison between our proposed algorithm and the ACO. The results show that our proposed algorithm returns a better solution for most of the instances. The instances used in the comparison to ACO have an absolute total demand close to zero, which ensures that the system had enough bikes for balancing the BSS.

We also tested the algorithm using other real instances where the absolute total demand was not close to zero. The instances were collected from Denver, United States; Dublin, Ireland; Ciudad de Mexico, Mexico and Minneapolis, United States and they have absolute total demands of 35, 64, 87 and 92, respectively. In solving this set of instances, we varied α from 0.1 to 1.0 and determined the best solution.

TABLE III. Performance Comparison of the Proposed Algorithm and the Best Known Solution Value for Real World Instances

			ACO [22]		Proposed Algorithm	
	N	Q	Total distance	Total residual	Total residual	Total residual
LaSpezia	20	30	21,518	1	21,475	1
		20	23,488	1	21,475	1
		10	23,908	1	23,529	1
Ottawa	21	30	17,178	0	17,166	0
		20	17,178	0	17,166	0
		10	17,604	2	17,166	2
Madison	28	30	34,029		35,877	8
		20	34,706	8	34,420	8
		10	38,677	8	37,271	8

TABLE IV. Performance Comparison of the Proposed Algorithm and the Best Known Solution Value for Real World Instances

			Best known solution value[23,21]	Proposed Algorithm		
	N	Q	Total distance	Total distance	Total residual	α
Denver	51	30	51,583	53,740	5	0.7
		20	53,369	57,116	15	0.9
		10	67,025	63,281	25	0.4
Dublin	45	30	33,548	41,274	34	0.7
		20	39,786	40,489	44	1.0
		11	54,392	48,172	53	1.0
Ciudad de Mexico	90	30	88,227	79,234	57	0.2
		20	116,418	88,213	67	0.3
		17	109,573	91,188	70	0.3
Minneapolis	116	30	137,843	186,716	62	1.0
		20	186,449	237,856	72	0.4
		10	298,886	294,405	82	0.6

As shown in Table IV, the proposed algorithm returned solution values close to the best known solutions in a number of instances. Moreover, our algorithm returned solution values better than the best known solution values for other instances, especially when the maximum capacity of the truck was small. We should note that the residual shown in the table is due to the imbalance in the total demand. For example, the Minneapolis instance had an absolute total demand of 92, which means that the number of pick-up bikes was more than the number of drop-off bikes by 92 bikes. In this case, when the truck's maximum capacity was 10, we found that the best

solution had a residual of 82 bikes, with 10 bikes loaded on the truck at the end of the tour.

VII. Conclusion

We proposed a novel approach, based on the deferred acceptance algorithm, to solve the SBRP problem in BSSs. The proposed algorithm consists of two stages. The first is the tour construction stage, which begins by assuming that each NotSpot and the depot form a partial tour. Each partial tour is then grown by matching the partial tours with the NotSpots. The matched pairs of partial tours and NotSpots are subsequently stacked, and the matching game is repeated until each partial tour is composed of all NotSpots. The second stage takes the constructed tours and uses the 2-opt algorithm to search its neighborhood for a better solution. We compared the proposed algorithm to the ACO, and the results show that our proposed algorithm outperforms the ACO in most instances. Moreover, we solved medium size instances with up to 116 NotSpots, then compared our solution to the best-known solution. We found that our solution was close to the best-known solution for some instances and better than the best-known solution for other instances.

References

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

[2] R. Arnott and K. Small, "The Economics of Traffic Congestion," American Scientist, vol. 82, pp. 446-455, 1994.

[3] S. Shaheen, S. Guzman, and H. Zhang, "Bikesharing in Europe, the Americas, and Asia: past, present, and future," Transportation Research Record: Journal of the Transportation Research Board, pp. 159-167, 2010.

[4] T. Firestine, "Bike-Share Stations in the U.S.," D. o. t. B. o. T. Statistics, Ed., ed, 2016.

[5] H. Hernández-Pérez and J.-J. Salazar-González, "A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery," Discrete Applied Mathematics, vol. 145, pp. 126-139, 12/30/ 2004.

[6] K. Y. Lee and M. A. El-Sharkawi, Modern heuristic optimization techniques: theory and applications to power systems vol. 39: John Wiley & Sons, 2008.

[7] C. Fricker and N. Gast, "Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity," EURO Journal on Transportation and Logistics, vol. 5, pp. 261-291, 2016.

[8] H. Hernández-Pérez and J.-J. Salazar-González, "Heuristics for the one-commodity pickup-and-delivery traveling salesman problem," Transportation Science, vol. 38, pp. 245-255, 2004.

[9] M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, et al., "Balancing the stations of a self service “bike hire” system," RAIRO-Operations Research, vol. 45, pp. 37-61, 2011.

[10] M. Rainer-Harbach, P. Papazek, B. Hu, and G. R. Raidl, "Balancing bicycle sharing systems: A variable neighborhood search approach," in European Conference on Evolutionary Computation in Combinatorial Optimization, 2013, pp. 121-132.

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

[12] S. C. Ho and W. Y. Szeto, "Solving a static repositioning problem in bike-sharing systems using iterated tabu search," Transportation Research Part E: Logistics and Transportation Review, vol. 69, pp. 180-198, 9// 2014.

[13] S. Xiaoyan, Z. Fanggeng, and G. Yancheng, "Genetic algorithm for the one-commodity pickup-and-delivery vehicle routing problem," in 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009, pp. 175-179.

[14] C. Contardo, C. Morency, and L.-M. Rousseau, Balancing a dynamic public bike-sharing system vol. 4: Cirrelt Montreal, 2012.

[15] J. Shu, M. C. Chou, Q. Liu, C.-P. Teo, and I.-L. Wang, "Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems," Operations Research, vol. 61, pp. 1346-1359, 2013.

[16] S. Ghosh, P. Varakantham, Y. Adulyasak, and P. Jaillet, "Dynamic Repositioning to Reduce Lost Demand in Bike Sharing Systems," Journal of Artificial Intelligence Research, vol. 58, pp. 387-430, 2017.

[17] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," The American Mathematical Monthly, vol. 69, pp. 9-15, 1962.

[18] L. B. Wilson, "An analysis of the stable marriage assignment algorithm," BIT Numerical Mathematics, vol. 12, pp. 569-575, 1972.

[19] C. Ng and D. S. Hirschberg, "Lower bounds for the stable marriage problem and its variants," SIAM Journal on Computing, vol. 19, pp. 71-77, 1990.

[20] S. Goodyear. (2014). Mapping the Imbalances of New York's Popular, Troubled Bike-Share. Available: http://www.citylab.com/commute/2014/03/mapping-imbalances-new-yorks-popular-troubled-bike-share/8699/

[21] M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani, "The bike sharing rebalancing problem: Mathematical formulations and benchmark instances," Omega, vol. 45, pp. 7-19, 6// 2014.

[22] C. W. Bortner, C. Gürkan, and B. Kell, "Ant Colony Optimization Applied to the Bike Sharing Problem."

[23] G. Erdoğan, M. Battarra, and R. Wolfler Calvo, "An exact algorithm for the static rebalancing problem arising in bicycle sharing systems," European Journal of Operational Research, vol. 245, pp. 667-679, 9/16/ 2015.