Improvements to Worker Assignment in Bike Sharing Systems

Trent Johnson
Department of Mathematics
Georgia Institute of Technology

Jie Wu
Department of Computer and Information Sciences
Temple University

***Abstract*—Bike-sharing systems (BSSs) are widely used in cities worldwide as they offer an affordable, eco-friendly method of transport. However, the rate of renting and returning bikes from stations is not always equal. The stations with imbalanced demand can become out of service by having all docks filled or emptied. These out-of-service stations can lead to a worse user experience and fewer people using BSSs. Researchers are trying to solve the rebalancing problem by developing algorithms that incentivize workers to pick up and drop off bicycles from different stations to balance the rent and return rates. Many of these algorithms focus on creating incentive and pricing models to encourage workers to go to imbalanced stations. Since they do not consider all of the placements of workers, this strategy may lead to inefficiencies where workers travel farther than they need. We can treat rebalancing as a *Worker Assignment Problem* by assigning worker stations to minimize the total distance traveled. We propose an algorithm that can approximate the optimal assignment significantly faster than other techniques with very high performance. The rapid speed allows for real-time use in augmenting pricing models and as a stand-alone method for worker assignment. Furthermore, we compare our approach against four existing algorithms on real-world data to evaluate computational speed and effectiveness.**

***Index Terms*—bike-sharing, weighted matching problem, bike rebalancing, urban computing.**

I. Introduction

In recent years we have seen a dramatic rise in the number of *bike sharing systems (BBSs)* around the world [1]. They are a flexible and cheap form of transportation that requires little infrastructure, provide affordable transportation to under-serviced areas, offer health benefits to users, are an environmentally-friendly mode of transportation, and help reduce traffic [1]. These benefits significantly improve a city on both an individual and a city-wide scale.

These systems can either be *docked*, users drop bikes off at stations, or *dock-less*, users can drop bikes off anywhere. In docked systems, many stations experience different levels of return and rental rates throughout the day. These imbalances can cause stations to either have a shortage of bicycles to rent or be near max capacity, which we will respectively refer to as *underflow* and *overflow* stations. Both underflow and overflow stations can become *out of service stations (OSSs)* if they either entirely run out of bikes or empty docks. OSSs are unusable to many users and often occur in areas and times with significant demand. This dissatisfaction decreases the

This research was supported in part by NSF REU site grant CNS 1757533.

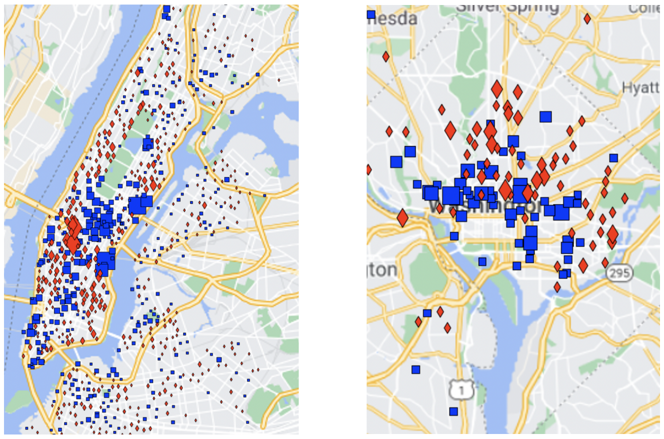
Fig. 1: Bikes at a bike sharing dock in New York City

usage, potential profits, and benefits of a BSS. Furthermore, for dock-less BSSs, there can be areas with too many or not enough bicycles, resulting in a similar issue. Thus, BSSs must rebalance stations and zones to reduce the number of OSSs.

Many algorithms have BSS users either give incentives [2], [3] or assignments [4] to change what stations they rebalance. When using incentives, the BSS will reward users some amount if they return bikes to OSSs. Therefore, any BSS user can participate and choose if/what station to rebalance. This allows for gamification with leaderboards and prizes, which can help increase the likelihood that users will rebalance. On the other hand, we could hire a group of workers from the pool of BSS users. As workers use the BSS, we can alternate routes to their destination, making them rebalance stations. Using workers allows for the total distance traveled to be minimized as we can control where they go.

Rebalancing can be separated into spatial and temporal domains by dividing the day into time slices in which the algorithm makes the workers' assignments or incentives [2], [4]. Duan and Wu defined *Worker Assignment Problem (WAP)* as the spatial component of rebalancing where workers are assigned tasks. Workers would walk from their source to an assigned overflow station, bike to an assigned underflow station, and walk to their destination. Thus, the problem focuses on matching overflow stations, underflow stations, and workers to minimize the total detour taken by workers. Unfortunately, WAP is NP-Hard, so solutions need to be approximated [4].

In addition, when solving BSS rebalancing, one can implement a more combinatorial route or a more machine learning and data-driven approach. Deep models are well suited to related issues such as predicting station demand [5], but they



(a) (b)

Fig. 2: The overflow (blue squares) and underflow (red diamond) stations after the a) NYC and b) Washington DC evening commute.

suffer from being very difficult to understand. Wu suggests that combinatorial techniques are well suited to the role of WAP and similar assignment problems. Some advantages of this approach include potentially having lower bounds on performance, no need for data, and being understandable. The ease of understanding is paramount as bike-sharing systems are often run by or in conjunction with governments, so understanding why the algorithm is making decisions is critical for accountability and transparency. Thus, we use combinatorial approaches rather than data-driven algorithms in this work.

However, when solving WAP, Duan and Wu restricted the number of workers, overflow, and underflow targets to be equal, so workers were given complete assignments, assignments with both underflow and overflow stations. Restricting worker counts is unrealistic as worker counts could be far high or lower than the targets. In this paper, we relax the definition of WAP and allow for any number of workers. When we have a surplus of workers, we can give partial assignments, assignments with only a single station to rebalance, to help reduce detours.

In this paper, we provide the following contributions:

- We extend the Worker Assignment Problem to include partial assignments and variable worker counts
- We introduce four algorithms into the field of BSS rebalancing
- Use real-world bike-sharing data to evaluate the performance and run-time of four algorithms
- Propose a new algorithm, *Iterative Round Search*, that has low detour assignments and run-time

II. Related Work

Earlier works have examined the rebalancing of BSSs from a plethora of perspectives. Designing better station layouts to reduce the rebalancing problem has been investigated by O'Dell et al. [6] who created a methodology for optimizing the placement of docks. Furthermore, Caggiani et al. used linear programming to maximize spatial equality [7] and Zhang et al. utilized particle swarm optimization and GPS data to create station placements to maximize the reductions in carbon emissions by the BBSs [8].

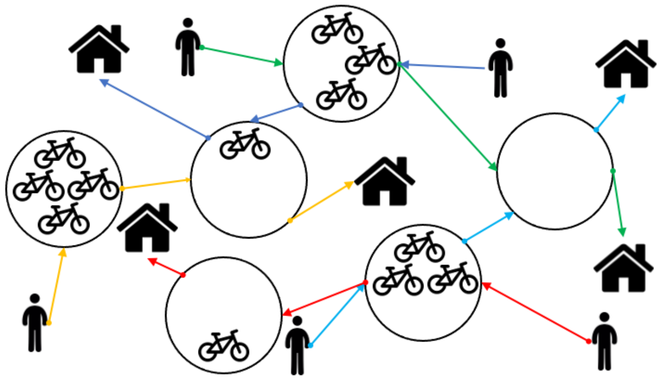


Fig. 3: Workers have a start and end destination and an assignment of stations is given to them that both rebalances stations and minimized detour

TABLE I: Run-time on Citi Data (s)

Variable	Definition
W, O, U	Set of workers, overflow stations, and underflow stations
w, o, u	Any worker, overflow station, or underflow station
$dis(w, o, u)$	Total distance from w's source, o, u, to w's destination

Algorithms for BSS rebalancing tend to look into two approaches. The first is to use trucks to move bikes from overflow stations to underflow stations. Past works have designed many algorithms for optimal routes for these trucks [9], [10], [11]. However, trucks are expensive and not very efficient. In addition, they cause congestion and pollution, which can reduce the advantages of having a bike-sharing system.

Thus, many researchers have been looking into using crowdsourcing BSS users to rebalance the system by offering incentives to alternative stations. Most user-based rebalancing strategies focus on finding optimal pricing and incentive schemes to most effectively encourage workers to go to the stations that are at high risk of going out of service [3],[12]. For example, Bike Angels is a machine learning algorithm used in Philadelphia, New York, and Washington DC that gives users points for renting and returning bicycles to overflow and underflow stations. These points can be utilized for prizes and discounts [2]. This approach requires a three-month data gathering and testing phase before full implementation. In addition, Pan created a pricing model using a reinforcement learning model to assign workers best [3]. However, these systems do not attempt to minimize the total detour taken by workers, so workers can travel further than they need to. All of the unnecessary detours taken by workers can increase the total cost of rebalancing extra distance, reduce worker satisfaction, and decrease the number of willing workers.

III. Problem Formulation

The *Worker Assignment Problem* has various formulations and perspectives. Furthermore, most of the algorithms evaluated in this paper are not designed for WAP, but they are

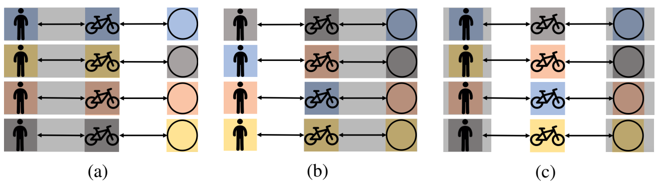


Fig. 4: Graphs with two fixed vertex sets a) shows workers and overflow fixed together b) shows overflow and underflow fixed together, and c) shows workers and underflow stations.

created for similar problems easily reducible to WAP. Thus, we will show these related problems and their reduction.

A. *Matching Problems*

In general, matching problems take a graph and seek to find a set of edges such that each vertex is part of at most one edge in the chosen set. We can give edges weights to create *Weighted Matching*, where we must find a set of disjoint edges, edges which do not share a vertex, with the maximal sum of weights. Perfect weighted matching (equivalent to the *Assignment Problem*) seeks to create a set of edges on a fully connected bipartite graph with the two vertex sets having equal sizes. A variation of Hungarian algorithm can solve bipartite matching with on a bipartite graph with vertex sets $V_1 > V_2$ in $O(V_1V_2log(V_1))$ [13].

B. *Weighted 3-Dimensional Matching*

Weighted 3-Dimensional Matching is an extension of Weighted Matching that uses 3-dimensional hypergraphs. A 3 dimension hypergraph $G = (V, E)$ has edges that connect three vertices rather than 2, so $E \subset V \times V \times V$. To solve Weighted 3-Dimensional Matching, we use a weighted 3D hypergraph to create a set of disjoint edges to maximize the sum of the chosen edges. Unfortunately, this problem is NP-Hard. [14]. This problem is a special case of Weighted 3-Dimensional Matching, where the graph must be tripartite. However, the 3-Index Assignment Problem is NP-Hard [15].

C. *Reduction*

We can solve WAP using 3-Dimensional Matching. First, we define W as the set of workers. Then for every overflow station, we will add it to the set O a number of times equal to its rebalancing target. Every element in O represents the need for one bike for a station. Likewise, for every underflow station, we will add the station to the set U a number of times equal to its rebalancing target. Thus, removing/sending a bike to each cloned station in O, U will remove/send the target amount of bikes to target stations. Note that $|O| = |U|$ by the definition of WAP.

To reduce WAP to 3-Dimensional Matching, we first create a hypergraph G with three sets of vertices W, O, and U. We define $\forall(w \in W, o \in O, u \in U)$, the total distance traveled by a worker w from their start, to o, to u, and to their destination to be $dis(w, o, u)$. Next, $\forall(w, o, u) \in (W \times O \times U)$, let (w, o, u) be an edge in G with a weight equal to a large constant minus $dis(w, o, u)$. We can use weighted 3-Dimensional Matching to obtain maximal matching of G. A worker and a station connection signify a worker adding/removing a single bicycle from that station in the final matching. Thus, the matching represents an assignment that fulfills the station targets and minimizes the total distance traveled.

IV. Algorithms

In this section, we describe the evaluated algorithms in the paper: Two Round Matching [4], Local Ratio [16], Hungarian Search [17] and Genetic Hungarian Search [17]. Furthermore, we present our algorithm: Iterative Round Search.

A. *Two Round Matching*

Two Round Matching is an algorithm developed by Duan and Wu [4] that performs two rounds of matching to minimize detour. This algorithm is the only algorithm designed specifically for WAP that we tested. It performs two rounds of maximal matching to achieve the best assignment. It first pairs stations and then assigns workers to the station pairs. However, this algorithm is restricted to problems with an equal number of workers, overflow, and underflow targets. Thus, we propose TRM* to extend the algorithm to variable worker counts.

In the first round, a graph $G = (V, E)$ is created with $V = O \cup U$ and $E = O \times U$ with edges having weight equal to a large constant minus $dis(o, u)$ for all $(o, u) \in E$. Next, we perform a weighted matching on G to get the assigned edges E'. Thus, we have created the minimum distance routes to rebalance from overflow to underflow stations. In the second round, we create G^* with the vertex set $V^* = W \cup O \cup U$ and edge set $E = E' \times W$ with edge weights equal to a large constant minus $dis(w, o, u)$ for all $w, o, u \in E$. We then do a final round of weighted matching of G^* to a set of edges. We can use those edges to assign an overflow and underflow station to each worker that minimizes the total distance traveled. Therefore, we have reduced WAP to 3-Dimensional Matching In addition, outputs of TRM* are 3-approximate solutions to WAP [4].

However, this algorithm only works with scenarios when workers, overflow targets, and underflow targets are equal. We will relax the need for workers to be equal to station targets and use Karp's algorithm instead of the Hungarian Method to perform the matching [13]. This allows the bipartite graph's vertex sets to be different sizes to utilize different numbers of stations and workers. [18]. Unfortunately, TRM* also cannot give partial assignments to workers. For $M = max\{U, W\}$, TRM* has a time complexity of $O((MO)log(M))$.

B. *Random Hungarian Search*

An approximation used to solve the 3-Index Assignment is called *Hungarian Search (HS)* [17]. In this process, we take the tripartite graph with vertex sets V_1, V_2, V_3 with assignment $A \subset V_1 \times V_2 \times V_3$. Then, for each assignment, we randomly pick two sets to fix together in the assignment. Then, we use a Weighted 2D-Matching Algorithm to optimize the cost of the assignments by changing the remaining set's assignments.

Jiang et al. used this algorithm using a graph initialized with random edges as a baseline [19]. We will denote *Random*

Algorithm 1: Two Round Matching* [4]

```
Input  : Sets of workers $W$, overflow target stations
         $O$, underflow target stations $U$
Output: The minimum weight worker assignment of G
for $o \in O, u \in U$ do
    $E \leftarrow E \cup (o, u)$;
    $e(o, u) \leftarrow dis(o, u)$;
end
$X \leftarrow$ min-cost perfect matching of $G(V, E)$;
$V^* \leftarrow W \cup O \cup U$, $E^* \leftarrow \emptyset$;
for $x \in X, w \in W$ do
    $E* \leftarrow E^* \cup (x, w)$;
    $e(x, w) \leftarrow dis(w, x)$;
end
return full min-cost matching of $G^*(V^*, E^*)$;
```

Algorithm 2: Hungarian Search [17]

```
Input  : $W$, $O$, $U$, worker assignment graph $G$.
Output: The minimum weight worker assignment of G
while $dis(G)$ is decreasing do
    for $V_1, V_2$ in shuffle($[(W, O), (O, U), (W, U)]$) do
        $G^* \leftarrow$Freeze all edges in $G$ between $V_1, V_2$;
        $G \leftarrow$ The full min-cost matching of $G^*$;
    end
end
return G;
```

Hungarian Search (RHS) as creating a random assignment and optimizing them with HS. The time complexity of running an iteration of RHS is $O((RMU)log(M))$.

C. Genetic Hungarian Search

Huang and Lin developed a genetic algorithm denoted as *Genetic Hungarian Search (GHS)* that used HS to optimize each potential assignment after selection occurred [17]. In a standard genetic algorithm, we create a population of P potential assignments (or species) and define the fitness of each species to the total distance traveled. Then, according to their fitness values, species are randomly chosen as parents and joined together to create offspring. The offspring are then mutated to form the next generation of assignments. This process is repeated for L generations or until the population has converged. However, for GHS, instead of a standard mutation, we will run HS on each offspring. This will both optimize the graphs and act as a way to change the offspring pseudo-randomly. Furthermore, the algorithm will use Partially Mapped Crossover on P parents to create a pool of $2P$ offspring. These offspring and then pruned of any duplicate children and the best P will be selected. If there are not P offspring after the duplicate pruning, the evolution is considered converged, and the algorithm terminates.

In our experiment, we simulated four generations with a population size of 8. The HS optimization of each species was multi-threaded to save time. In addition, we use Karp's

Algorithm 3: (Proposed) Iterative Round Search

```
Input  : $W$,$O$, $U$.
Output: The minimum weight worker assignment
$G(V, E) \leftarrow TRM^*(W, O, U)$;
$G^*(V, E) \leftarrow HS(W, O, U, G)$;
return $G^*$;
```

Matching Algorithm, so for a population size of P and L generations, we have a time complexity of $O((RPMU)log(M))$.

D. Iterative Round Search

We propose *Iterative Round Search (IRS)* as the best algorithm for this problem. In IRS, we first create a graph from TRM*. Then optimize the graph using Hungarian Search, which can only improve the 3-approximate TRM solution. Since TRM* is so fast, it adds little time to the computation. HS can then optimize a 3-approximate solution, enabling a rapid convergence for a close approximation. Other algorithms use HS for an initial graph to be later optimized by a genetic algorithm [17], or a global search [19]. Alternatively, IRS uses an excellent initialization for HS to quickly produce accurate approximations with no downstream optimization.

E. Local Ratio

Local Ratio (LR) was developed by Chan, and Lau [16]. Inspired by fractional coloring, a good ordering of edges is made via linear programming. For $G = (V, E)$, let σ_e be the edge weight for $e \in E$ and $N(v)$ being the set of the edges adjacent to v. Order edges based on the x values from the linear programming problem

$$\begin{aligned} \text{max} \quad & \sum_{e \in E} x_e \sigma_e & \\ \text{s.t.} \quad & \sum_{e \in N(v))} x_e \sigma_e \leq 1 & \forall x_e \\ & x_e \geq 0 & \forall x_e. \end{aligned}$$

Next, Local Ratio is used to iterative prunes poor edges and adds the first valid edge in the ordering to the matching. Once all edges have been added or removed, the resulting matching is a 2-approximate solution [16]. However, when solving this problem, a set of all possible WU^2 edge weights must be built. Thus, this algorithm is at a minimum of $O(WU^2)$ in addition to the linear programming.

V. Data and Experiment

For this experiment, we used publicly available data and created simulations to evaluate performance.

A. Data Sets

For this experiment, we used three publicly available *New York City CitiBike Data Set* [20], *Washington DC Capital Bike Share Data Set* [21], and *Boston Bluebike Data Set* [21]. These data sets have $2,718,594$ trips for CitiBike, $219,155$ trips for Capital, and $270,893$ trips for Bluebikes in May 2021.

For all data sets, we used data gathered from May 2021 and used UTM coordinates for each station. Since this data set

TABLE II: Increase in Travel Distance (%)

	TRM*	RHS	IRS	GHS	LR	TRM*	RHS	IRS	GHS	LR	TRM*	RHS	IRS	GHS	LR
1/5	0.14	0.09	0.05	0.16	**0.03**	0.10	0.05	**0.03**	0.1	**0.02**	0.1	0.05	**0.03**	0.1	**0.02**
1/4	0.15	0.09	**0.06**	0.15	**0.05**	0.12	0.06	**0.04**	0.1	**0.03**	0.12	0.06	**0.04**	0.1	**0.03**
1/3	0.16	0.1	**0.08**	0.15	**0.07**	0.13	0.07	**0.05**	0.11	**0.04**	0.13	0.07	**0.05**	0.1	**0.04**
1/2	0.21	0.13	**0.12**	0.16	**0.11**	0.16	0.09	**0.08**	0.12	**0.07**	0.15	0.09	**0.08**	0.11	**0.07**
1	0.47	**0.45**	**0.44**	**0.45**	**0.43**	0.43	0.4	0.4	0.41	**0.38**	0.41	0.37	0.37	0.38	**0.35**
2	0.17	**0.15**	**0.15**	**0.14**	**0.14**	0.17	0.16	0.16	**0.15**	**0.15**	0.17	**0.14**	**0.15**	**0.14**	**0.14**
3	**0.09**	**0.08**	**0.09**	**0.08**	**0.08**	0.10	**0.09**	0.1	**0.08**	**0.09**	0.1	**0.08**	**0.09**	**0.08**	**0.08**
4	**0.06**	**0.05**	**0.06**	**0.05**	**0.05**	**0.07**	**0.06**	**0.06**	**0.06**	**0.06**	**0.06**	**0.05**	**0.06**	**0.05**	**0.05**
5	**0.05**	**0.04**	**0.05**	**0.04**	**0.04**	**0.05**	**0.04**	**0.05**	**0.04**	**0.04**	**0.05**	**0.04**	**0.04**	**0.04**	**0.04**

TABLE III: Run-Times (s)

	TRM	RHS	IRS	GHS	LR	TRM	RHS	IRS	GHS	LR	TRM	RHS	IRS	GHS	Local Ratio
1/5	**0.03**	0.09	0.08	14.31	2.02	**0.03**	0.09	0.08	16.19	2.17	**0.03**	0.09	0.08	16.57	2.28
1/4	**0.03**	0.11	0.09	15.07	2.64	**0.03**	0.10	0.09	15.36	2.86	**0.03**	0.11	0.10	15.45	3.02
1/3	**0.03**	0.14	0.12	15.74	3.77	**0.03**	0.14	0.12	15.80	4.06	**0.03**	0.14	0.13	16.17	4.36
1/2	**0.04**	0.21	0.17	16.75	6.30	**0.04**	0.21	0.18	16.71	6.90	**0.04**	0.21	0.18	17.03	7.28
1	**0.05**	0.47	0.36	19.40	16.75	**0.05**	0.49	0.38	19.53	18.73	**0.06**	0.49	0.40	20.36	20.38
2	**0.86**	6.14	4.45	140.76	40.29	**0.92**	6.25	4.66	147.93	45.73	**0.96**	6.72	4.99	153.26	52.33
3	**1.7**	11.71	8.47	255.82	69.38	**1.8**	12.45	9.35	274.04	89.87	**1.87**	12.98	9.75	281.38	86.49
4	**2.52**	17.60	12.50	372.20	105.25	**2.69**	18.67	13.59	402.83	120.82	**2.79**	19.45	14.35	410.76	129.51
5	**3.34**	23.50	16.81	487.48	148.07	**3.6**	24.80	18.12	518.62	167.63	**3.7**	25.52	18.25	535.59	189.38

does not contain worker sources and destinations, we created workers by randomly picking start/end stations weighted by the number of trips that started/end at that station in the time interval. Then, we created a biker with source/destination location randomly within 500 meters of the station. Also, we created targets for the station equal to the net number of bikes that left/arrived at the station in the time slice.

B. Experimental Set-Up

The data sets were divided into time slices between 100 and 400 stations and workers. We could not use larger graphs as Local Ratio required more RAM than feasible for our computers. Time slices were randomly selected in May 2021 from 7 am to 7 pm. Finally, we fabricated station rebalancing targets by tracking how many bikes left/arrived at each station in the time slice. For non-binary station targets, we split each station into a number of copies equal to the rebalancing target at the same location with a target of ± 1. If there were fewer workers than station targets, workers had two assignment stations. On the other hand, if there were more workers than stations, they could have partial assignments. If a worker is not assigned a start or end station, they go to the station that minimizes the distance traveled. We approximated the actual geographic distance with Euclidean distance.

C. Results

In Table II, we display the percent change in distance that workers take by rebalancing instead of optimal routes. We can see that detour added steadily increases for all algorithms workers, and targets are equal. When worker counts are low, workers can be routed to almost any station with little competition for stations in desirable areas. However, as worker count increases, we start sending workers to stations far away to balance. This results in a more balanced system at the cost of efficiency. Once we start adding more workers, we then begin to have a larger pool of workers from more locations to reroute. Having more workers allows us to make more efficient assignments with our workers. Wu points that increased worker satisfaction leads to more worker recruitment [4]. If we hire workers, the distance traveled by the average worker decreases. Then, workers are more satisfied, so more workers are recruited. These relations create a positive feedback loop that results in a large pool of workers [4]. However, we only see these benefits when worker and target sizes are vastly different. Thus, a system should have fewer rebalancing targets for workers to effectively balance stations and utilize the worker feedback loop.

Another point of interest is that these algorithms give fewer detours in extreme worker shortage situations than worker surplus scenarios. The difference in detours may be due to ”bad” stations that are very hard to rebalance as they are out of the way of all usual worker paths. The worker shortage scenario can not assign workers to these stations, but these bad stations must be rebalanced with enough workers. This leads to less efficient assignments.

When looking at the individual algorithms, it is clear that Local Ratio does the best at minimizing detour in all cases but one. On the other hand, TRM* performs the worst overall. This makes sense as only two rounds of matching are performed. However, one surprising fact was TRM* performed relatively similar to the other algorithms despite not having partial assignments. Furthermore, our proposed algorithm usually performs second best, especially with worker shortages. Also, differences in algorithmic performance tend to drop as we added more workers because having more workers gives more flexibility in assigning.

Furthermore, in Table III we present the run-times of each algorithm. We can observe a steady increase in run-times as we add workers. In every case, TRM* is the fastest, with IRS being the second-fastest. Far behind are LR and GHS, which take up to 51 and 144 times longer to complete. LR uses copious amounts of RAM, which may be a reason for its slow speed.

Furthermore, we can examine the time complexity of each algorithm. Note that $M = max\{U, W\}$, and for GHS, P is the population size, and L is the number of generations. Since HS continues until it can no longer improve the score, we introduce R, which is the maximum number of rounds HS can complete. Thus, TRM* is $O((MU)log(M))$, GHS is $O(LPR(MU)log(M))$, IRS is $O(R(MU)log(M))$, and LR is $O(WS = U^2)$ plus linear programming. TRM* by far has the lowest time complexity. Furthermore, GHS has the worst time complexity as it performs Hierarchical Search PL times. From a time complexity standpoint, RHS and IRS are the same, but in run-time, IRS is faster despite doing an extra step beforehand. The faster run-time is due to TRM* rapidly getting a good initialization, so only a few rounds of optimization are needed for IRS, while RHS needs many iterations on the random graph before converging.

VI. Conclusion

This paper introduces the usage of 3-Dimensional Matching and 3-Index assignment algorithms as potential solutions to BSS rebalancing. Furthermore, we evaluated our proposed algorithm, Iterative Round Search, against four algorithms. Finally, we show that IRS has similar performance to the best algorithms tested, runs faster than these algorithms, and can have reduced time complexity. The speed is due to a rapid and good initialization from TRM*, and the excellent approximation is from the iterative optimization of HS. The fast speed of this algorithm can help reduce computing costs and allow for more frequent updates of assignments. The near real-time updates and high accuracy of IRS will reduce detours for workers, which will reduce the costs of hiring workers.

Furthermore, Iterative Round Search is not limited to bike-sharing. It works as an approximation to the general Weight 3-Dimensional Matching problem. Furthermore, it can be tailored to domain-specific uses by replacing TRM* with another quick heuristic. This can allow for applications in networking, logistics, and other industries.

Some areas of further research can be to investigate using WAP to extend pricing models. The speed of IRS allows us to calculate the total detour assuming a given user when to different stations. If a user going to a particular station lowers/raises the total amount of detour, we can (de)incentive them to go to that station. Users could then have personalized prices for stations to help reduce global costs. Also, adding a fairness constraint to make sure all workers are utilized equally and prevent worker dissatisfaction. Furthermore, a constraint on how far the worker has to walk or more heavily penalized long walks to stations could give workers more satisfactory assignments. These could further improve station rebalancing and increase the benefits of BSSs.

References

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

[2] H. Chung, D. Freund, and D. B. Shmoys, "Bike angels: An analysis of citi bike's incentive program," in *Proceedings of 1st ACM COMPASS*, no. 5, 2018, pp. 1–9.

[3] L. Pan, Q. Cai, Z. Fang, P. Tang, and L. Huang, "A deep reinforcement learning framework for rebalancing dockless bike sharing systems," in *Proceedings of the AAAI*, vol. 33, no. 01, 2019, pp. 1393–1400.

[4] Y. Duan and J. Wu, "Spatial-temporal inventory rebalancing for bike sharing systems with worker recruitment," *IEEE Transactions on Mobile Computing*, 2020.

[5] J. Wu, "Challenges and opportunities in algorithmic solutions for re-balancing in bike sharing systems," *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 721–733, 2020.

[6] L. Dell'Olio, A. Ibeas, and J. L. Moura, "Implementing bike-sharing systems," in *Proceedings of Institution of Civil Engineers-Municipal Engineer*, vol. 164, no. 2, 2011, pp. 89–101.

[7] L. Caggiani, R. Camporeale, B. Dimitrijević, and M. Vidović, "An approach to modeling bike-sharing systems based on spatial equity concept," *Transportation Research Procedia*, vol. 45, pp. 185–192, 2020.

[8] H. Zhang, X. Song, T. Xia, J. Zheng, D. Haung, R. Shibasaki, Y. Yan, and Y. Liang, "Maas in bike-sharing: smart phone gps data based layout optimization and emission reduction potential analysis," *Energy Procedia*, vol. 152, pp. 649–654, 2018.

[9] 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, 2014.

[10] M. Dell'Amico, M. Iori, S. Novellani, and T. Stutzle, "A destroy and repair algorithm for the bike sharing rebalancing problem," *Computers & Operations Research*, vol. 71, pp. 149–162, 2016.

[11] F. Cruz, A. Subramanian, B. P. Bruck, and M. Iori, "A heuristic algorithm for a single vehicle static bike sharing rebalancing problem," *Computers & Operations Research*, vol. 79, pp. 19–33, 2017.

[12] S. Ban and K. H. Hyun, "Designing a user participation-based bike rebalancing service," *Sustainability*, vol. 11, no. 8, 2019.

[13] R. M. Karp, "An algorithm to solve the m$\times$ n assignment problem in expected time o (mnlogn)," *Networks*, vol. 10, no. 2, pp. 143–152, 1980.

[14] R. Karp, "Reducibility among combinatorial problems," pp. 85–103, 1972.

[15] M. G. GJ and D. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*. WH Freedman and Company, 1979.

[16] Y. H. Chan and L. C. Lau, "On linear and semidefinite programming relaxations for hypergraph matching," *Math. Program*, vol. 135, no. 1, pp. 123–148, 2012.

[17] G. Huang and A. Lim, "A hybrid genetic algorithm for three-index assignment problem," in *IEEE CEC*, vol. 4, 2003, pp. 2762–2768.

[18] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[19] H. Jiang, J. Xuan, and X. Zhang, "An approximate muscle guided global optimization algorithm for the three-index assignment problem," in *IEEE CCE*, 2008, pp. 2404–2410.

[20] "Citibike system data," 2021. [Online]. Available: https://www.citibikenyc.com/system-data

[21] "Bluebikes bikeshare system data," 2021. [Online]. Available: https://www.capitalbikeshare.com/system-data