Optimal Re-balancing for Bike Sharing System

Xiaodi Duan, Tiantian Lin and Ziqian Qiu
Electrical and Computer Engineering
University of Toronto, Canada
E-mails:xiaodi.duan@mail.utoronto.ca
tiantian.lin@mail.utoronto.ca
ziqian.qiu@mail.utoronto.ca

***Abstract*—In this paper, we propose a solution to the bike rebalancing problem in a bike-sharing system using Ant Colony Optimization (ACO). To compare the performance of ACO with other methods, we also apply an Adaptive Genetic Algorithm (GA) to the problem. Our ACO algorithm considers various factors, such as the number of bikes available at each station, the demand for bikes at each station, and the distance between stations. Through computational experiments, we find that our ACO algorithm is able to effectively rebalance the bike-sharing system, resulting in improved utilization of bikes and a reduction in the number of empty or full stations. In contrast, the GA approach did not perform as well as ACO in terms of solution quality and computational efficiency. Overall, our results demonstrate the effectiveness of ACO for solving the bike rebalancing problem and highlight the potential for ACO to improve the efficiency and effectiveness of bike-sharing systems.**

I. Problem Characterization

Bike redistribution is an important and challenging aspect of bike-sharing systems because it helps to ensure that there are always enough bikes available for users at each station. This is particularly important in cities like Toronto with high demand for public transportation, where it may not always be possible to simply add more stations or increase the number of bikes at each station to meet this demand. The availability of bikes at bike-sharing stations can vary significantly depending on factors such as the station's location and patterns of bike usage in the surrounding area. For example, central stations in cities where many people work in the city center may be filled with bikes in the morning as commuters take them to work, while stations in the outskirts may be empty. Similarly, stations located on hills may be less likely to be used because people may prefer to leave their bikes at stations that are easier to access [1]. By using trucks to move bikes from stations that are not being heavily used to stations that are in high demand, bike-sharing systems can efficiently manage their available bike inventory and ensure that users always have access to a bike when they need it. This helps to make bike-sharing systems more sustainable and cost-effective, as it reduces the need for additional infrastructure and bike purchases. Additionally, bike redistribution helps to reduce urban traffic and air pollution by encouraging people to use bikes instead of cars for their daily transportation needs.

This study proposes a model that both predicts bike demands in different stations using regression analysis, as well as optimizes a route for redistribution using ACO. However, since the public environment is constantly changing, it's challenging to keep track of every movement and calculate the best route for such redistribution in a dynamic environment. Instead, we decided to focus our solution on the redistribution at night, when the movement of bikes is negligible. In addition, the usage of bikes is affected by the demand for commuting on weekdays and weekends. Therefore, we will analyze these two time zones separately.

II. Literature Review

The research community have shown their interest in the realm of vehicle redistribution problems, especially with bikes since it's one of the most rapidly growing industry. In a nutshell, the redistribution of bikes is done by a vehicle, usually a truck that travels between different stations and rearranges the number of bikes by picking up and dropping off. This operation can be divided into two categories: static and dynamic. A static redistribution usually happens at midnight or any time slot where the change in the number of bikes in the majority of stations is minimal and negligible. In contrast, a dynamic redistribution assumes that user activities can strongly affect the demand of each station and has taken many factors into account. An example would be traffic analysis with respect to the change of time proposed by Chiariotti et al. [3], as well as the constantly changing weather conditions in the middle of the day. Compared with static re-balancing, dynamic re-balancing is expensive since the system is remarkably large and complex [4]. Considering the variety of disadvantages in dynamic redistribution problems, we decided to approach the bike re-balancing problem statically.

A. Demand analysis

To solve a redistribution problem for static bike-sharing systems, understanding the demand of customers at each station and their impact is vital. Alvarez-Valdes et al. approached this problem by forecasting the unsatisfied demand of each station for the next day [5]. The demand is divided into two categories: 1) the user willing to withdraw a bike but arrives at an empty rack, and 2) the user willing to return a bike but finds the station full. Inspired by this categorization, we used positive and negative numbers to quantify the demands

at each station, in which a positive sign is "bikes needed" and its corresponding value represents the number of bikes.

B. Optimization operations

Many scholars proposed a solution to optimize the redistribution problem. Jia et al. approached this by finding the shortest routes for repositioning vehicles using the Artificial Bee Colony algorithm [6]. However, the demand for bikes varies by different factors such as traffic or commute purposes on weekdays and weekends. Therefore, we will analyze these two conditions separately. Another approach is using Ant Colony Optimization (ACO), which is first proposed to solve the Traveling Salesman Problem (TSP) [7]. Gaspero et al. combined ACO with Constraint Programming (CP) [8] and Bell et al. used multiple ACO to successfully find a solution for TSP that is known to be within the top 1% of optimal solutions [9]. It's shown that many scholars have used ACO to solve vehicle redistribution problems and have encouraging results, we will also approach this problem using ACO.

III. Problem Formulation and Modeling

A. Bike Station Demand Prediction Notation

The bike station demand is defined as the number of bikes that a station needs. In this prediction, we do not consider the bike stations that are under maintenance or out of service.

The total number of pick-up bikes in a station during a day, we denote s_{out}. The total number of drop-off bikes in a station during a day, we denote s_{in}. Then, using the formulation:

$$s_{delta} = s_{in} - s_{out} \tag{1}$$

A negative s_{delta} means on this day, customers are mostly riding bikes from this station to other stations, which implies that the station has s_{delta} fewer bikes than its actual demand. Similarly, a positive s_{delta} would mean that the station has $-s_{delta}$ more bikes than its actual demand.

Due to the distinctive distributions on different days, we are using the s_{delta} on weekdays.

B. Bike Re-balancing Optimization Notation

We denote a set V_{total} to represent the s_{delta} in each $station_i$, where n_{total} is the total number of bike stations.

$$V_{total} = \{v_1, v_2, v_3, ..., v_{n_{total}}\} \tag{2}$$

V is used to denote the s_{delta} for each $station_i$ which has a non-zero s_{delta}. Given the number of stations n_{total} in total, stations that need more bikes $n_{dropoff}$, and the number of stations that could provide more bikes n_{pickup}, in this problem, we denote:

$$V = \{v_1, v_2, ..., v_{n_{dropoff}}, v_{n_{dropoff}+1}, ..., v_{n_{dropoff}+n_{pickup}}\} \tag{3}$$

$$n = n_{dropoff} + n_{pickup} \leq n_{total} \tag{4}$$

n is the total number of bike stations that need to be re-balanced. In the implementation, we further selected only 20 stations in downtown Toronto with large $|s_{delta}|$.

The next task is to find the best route r_{best} that could visit both of these two kinds of stations and balance the number of bikes with the least travel distance.

C. Bike Demand Prediction with Analysis

We need to approximate the demand for each bike station in order to calculate the average number of bikes before the re-balancing at the end of the day. However, we are unable to track the exact data since our data set API is on a real-time basis. Therefore, we calculated the demand gap using temporary data at midnight from the API to demonstrate a common status before re-balancing operations take place. The data we used was from Tue Dec 13 2022 23:46:28 GMT-0500 (EST).

After that, we could know whether each station's demand value $s_{predict}$ is positive or negative and do the following multiobjective re-balancing route optimization.

D. Re-balancing Bike Optimization with Ant Colony Algorithm

We are using the ACO algorithm for each station to find the shortest route. The ACO algorithm simulates the behaviour of real ant colonies and the use of ant pheromone trails to determine the best route, in our case the shortest path to redistribute the most bikes.

The truck (used for redistributing bikes) is considered an individual ant, and its route is built by incrementally selecting bike stations until all stations have been visited. The visited stations are stored in the visited list. The decision for choosing the best route is based on the ant's probabilistic formula.

The Ant Colony Optimization algorithm and formula: In the first step, each ant selects a path, i.e. the order in which every location is visited. In the second step, the different paths are compared, and in the last step, the ants update the pheromone levels on each edge.

- Edge selection: The probability of the move from state x to state y of the k^{th} ant is:

$$p_{xy}^k = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\Sigma_{z \in allowed_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})} \tag{5}$$

  For ant k, p_{xy}^k denotes the probability of moving from state x to state y computed with attractiveness η_{xy} of the move and the trail level τ_{xy} of the move. η_{xy} is computed by the heuristic indicating the prior desirability of the move and τ_{xy} indicates the proficiency this move has been in the past, representing a posterior indication of the desirability of the move α and β are the parameters applied. [2]
  In our proposal, we add an indicator $contribution$ which implies the edge/node selection's optimal contribution to the overall task defined by $minimize(\sum_{i=1}^{n} \mid v_i \mid)$. We apply this indicator by Algorithm1 in which we select the candidates based on their potential contributions to the overall rebalancing task. This also ensures that the operator will not visit a node and do nothing.

Algorithm 1 Candidate Nodes Selection Algorithm

Initialize $contributions = \emptyset$
Initialize $candidates = \emptyset$
for each $node \in unvisited\_nodes$ **do**
  Initialize $c = 0$
  if $task > 0$ **then**
    $c \leftarrow min(num\_bikes, task)$
  else
    $c \leftarrow min(truck\_size - num\_bikes, |task|)$
  end if
  $contributions \leftarrow contributions + c$
  if $c \neq 0$ **then**
    $candidates \leftarrow candidates + node$
  end if

To emulate the situation when an operator makes decisions about which station to go to next, we are adding a local update to the attractiveness level η_{xy} when an ant makes a contribution to minimize the demand gap for y, e.g. if the ant currently at x carries 5 bikes and the station y_1 needs 3 more bikes to fill its demand gap, while station y_2 needs bikes, it is more likely that the operator would visit station y_2 because this move is more efficient. The equation for attractiveness η is:

$$\eta_{xy}^k = P \cdot c_y^k / l_{xy} \tag{6}$$

where P is a weighted parameter and c_y^k is the *contribution coefficient* which equals to the number of bikes that need to be moved at station y. l_{xy}^k is the cost of the move from x to y, which can be defined with the distance d_{xy}.

- Pheromone update: Pheromone update happens after the ants complete their tours by changing the trail level τ_{xy} based on whether it's a good or bad solution. One possible update rule is:

$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_k^m \Delta\tau_{xy}^k \tag{7}$$

ρ is the pheromone evaporation coefficient, m is the total number of ants, and $\sum_k^m \Delta\tau_{xy}^k$ is the pheromone deposited by ant k.
In a typical TSP problem,

$$\Delta\tau_{xy}^k = \begin{cases} Q/L_k & \text{if ant } k \text{ makes the } xy \text{ move} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where Q is a constant and L_k is the cost of the solution. In our case, we can update the pheromone by the solution's *task completeness level*, in other words, how the task $minimize \sum_{i=1}^n \mid v_i \mid$ is done; then we may update τ_{xy}^k by:

$$\Delta\tau_{xy}^k = \begin{cases} Q \cdot C_k/L_k & \text{if ant } k \text{ makes the } xy \text{ move} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where C_k is the completeness of the overall re-balancing task, quantified by $\sum_{i=1}^n \mid v_i \mid$). v_i is the initial demand gap defined in (2).

IV. Proposed Solution

In this section, we will explain the steps used to implement ACO on the current data set. First, we introduce how we use the K-means algorithm to simplify the task and significantly reduce the running time. Moreover, we discuss the performances of the ACO model and the PSO algorithm.

A. Data Preparation

We first read the CSV file of bike stations and find all the stations located within downtown Toronto defined by a set of boundary coordinates. Note the stations may be reconstructed or removed in the past year, hence we eliminated all invalid data by deleting rows containing stations that no longer exist. In addition, we used the K-means algorithm with 19 clusters to identify the regions in order to better visualize the demand gap.

We quantified the incoming and outgoing bikes as the offset of each node. The flow of bikes is demonstrated by the edges. A node with a positive value (green) indicates that the station has more bikes leaving than entering, and vice versa. The demand gap for each region is demonstrated in Fig1.

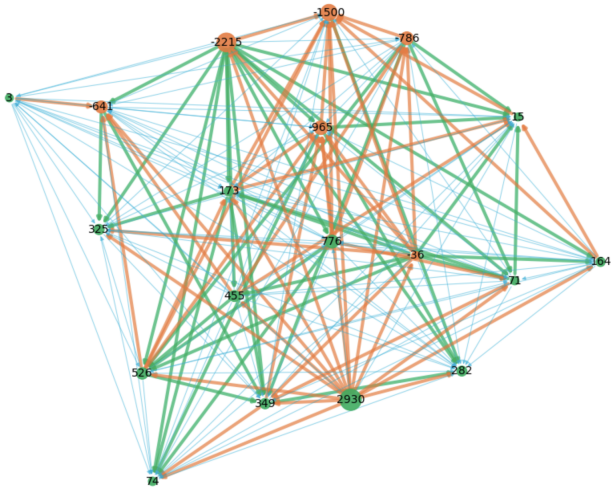

Fig. 1. clustered stations' bike number divergence

B. Demand Calculation and find unbalanced stations

We calculated the demand based on an average/median of the number of bikes in 2022 and filtered based on the offset in Fig1. After filtering, there are 20 stations that need to be re-balanced.

C. Ant Colony Optimization

The ACO solver first initiates the node search space with algorithm1, and generates the solution set based on the probability defined in eq.5. The solver may update the global best solution by three different indices: 1) minimizing total distance traveled, 2) maximizing completion score, and 3) minimizing ratio of distance/completion score. The results are further discussed in sec.4.

D. Comparative Study with Adaptive Genetic Algorithm

First, we created a function that takes a list of available bike stations and returns a measure of its quality depending on the parsed parameters. Same as ACO, the evaluation metrics are optimized by 1) minimizing distance travelled, 2) maximizing completion score, and 3) minimizing ratio of distance/completion_score. These criteria will be used to evaluate the quality of each solution and optimize the fitness result. We then create an initial population of solutions by picking a random candidate from the candidate's list defined in alg.1 recursively. It is done recursively because the candidate list will update upon the current node selection. Then by repeatedly applying crossover and mutation operations to the current population of solutions and using the fitness function to select the best ones to use as parents for the next generation. The function checks the mutation rate of the GA every 10 iterations and adjusts it based on the performance of the GA, and we stop the algorithm when the maximum number of iterations is reached.

V. Performance Evaluation

In this section, we will analyze the results from the proposed solutions, their performance, and their comparison.

A. Ant Colony Optimization

The first ACO solver is optimized by minimizing the total distance travelled. This would be an intuitively bad index, mostly because of the high probability of falling into a local best result. However, we presume this is the closest behaviour of a real-world operator who would most likely look around what is the nearest and forget about the distant ones. The result shows that the best solution provides the shortest path, but a significantly low completion score.

The second ACO solver is optimized by maximizing the completion score. Note that due to the insufficiency of extra bikes in this area of interest, the completion score can never reach 100%. The result of a 57.6% completion score aligns with our assumption, however, at the cost of a longer distance.

The third ACO solver is optimized by minimizing the ratio of distance to completion score. It generates the same high completion score of 57.6% while providing a shorter route compared with the second solver. Fig.2 shows the final result of the third solver on the map.

Fig. 2. ACO solved by minimizing distance/completion_score

B. Genetic Algorithm

Similar to the ACO solvers, the three GA solvers are optimized by the same three indices respectively. The output shows that by minimizing distance it has the shortest route of only 19333.1 meters and a decent task completion score of 52.5% while optimizing by maximizing completion score results in a longer route of 27496.1 meters. However, in the second solver, the completion score remains as high as the second ACO solver, which indicates that this is the global best result. In the third solver, the completion_score of 57.6% is still the highest, and in this case, it has a shorter route than the previous one, with only 21054.5 meters long.

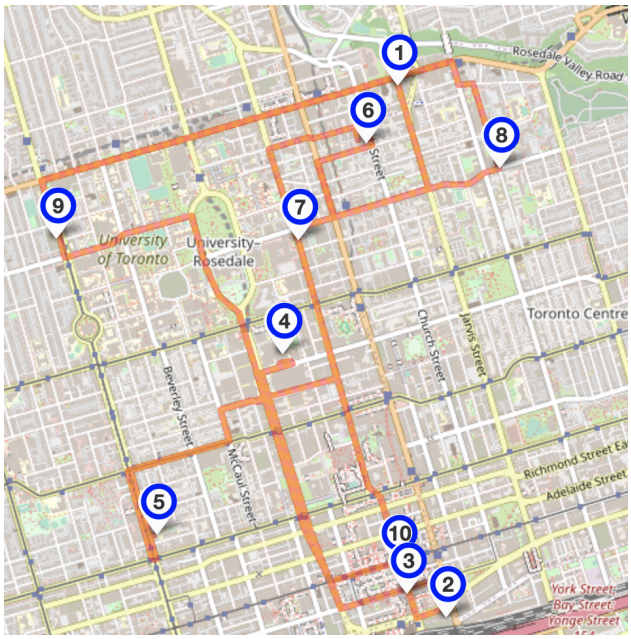

Fig. 3. aga3

C. Adaptive Genetic Algorithm

To compare the result of GA with its adaptive versions, we applied the Rechenberg's '1/5 success rule'. The result is pretty similar to the ones in GA, which is probably due to the limited data size.

TABLE I

solver	Best Solution Distance / meter	Best Solution Completion score	Best Solution Distance / Completion score	CPU Time
ACO1	32468.12	0.232	37285.14	0.98s
ACO2	13635.0	0.576	23681.84	0.99s
ACO3	13803.0	0.576	24147.31	0.98s
GA1	19333.1	0.525	36807.24	0.08s
GA2	27496.1	0.576	47756.30	0.06s
GA3	21054.5	0.576	36568.34	0.005s

VI. Conclusion

In this project, we aimed to optimize the bike-sharing system using the Ant Colony Optimization algorithm and Genetic Algorithm. We improved upon the standard ACO algorithm by introducing a new variable, named completion score. This completion score helps us to find the optimal route for a rebalancing truck in Downtown Toronto. By incorporating this variable, our multi-objective solution was able to optimize for both distance and completion level.

To compare with the Ant Colony Optimization algorithm, we used the Genetic Algorithm to simulate the nodes. From the performance, we could know that the Genetic Algorithm has a similar result compared with the Ant colony Optimization algorithm. However, the Ant Colony Optimization's best solution provides a lowest distance per completion score solution, which results in a shortest path with the highest completion score reached. Therefore, this bike-sharing system is more suitable for the ACO algorithm to perform bike re-balancing. However, this is at the cost of the much longer CPU running time of ACO.

Furthermore, we compared our solution with a simulated scenario in which a trucker makes decisions about which station to visit next by only optimizing for distance. Our results showed that while this approach would generate a much shorter route, the overall completion level is significantly lower than that of our proposed solution.

In conclusion, our work has demonstrated the effectiveness of using the ACO algorithm with the added completion score variable for optimizing bike share systems. This approach can improve the efficiency and effectiveness of these systems, ultimately benefiting both the users and operators of the system.

References

[1] T. Urli, "Hybrid meta-heuristics for combinatorial optimization," Constraints, vol. 20, no. 4, pp. 473–473, 2015. tp=&arnumber=7959977. [Accessed: 18-Oct-2022].

[2] "Ant colony optimization algorithms,"Wikipedia, 28-Sep-2022. [Online]. Available: https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms. [Accessed: 18-Oct-2022].

[3] F. Chiariotti, C. Pielli, A. Zanella, and M. Zorzi, "A Dynamic Approach to Rebalancing Bike-Sharing Systems," Sensors, vol. 18, no. 2, p. 512, Feb. 2018, doi: 10.3390/s18020512.

[4] B. Lahoorpoor, H. Faroqi, A. Sadeghi-Niaraki, and S.-M. Choi, "Spatial Cluster-Based Model for Static Rebalancing Bike Sharing Problem," Sustainability, vol. 11, no. 11, p. 3205, Jun. 2019, doi: 10.3390/su11113205.

[5] R. Alvarez-Valdes, J. M. Belenguer, E. Benavent, J. D. Bermudez, F. Muñoz, E. Vercher, and F. Verdejo, "Optimizing the level of service quality of a bike-sharing system," Omega, vol. 62, pp. 163–175, 2016.

[6] H. Jia et al., "Multiobjective Bike Repositioning in Bike-Sharing Systems via a Modified Artificial Bee Colony Algorithm," in IEEE Transactions on Automation Science and Engineering, vol. 17, no. 2, pp. 909-920, April 2020, doi: 10.1109/TASE.2019.2950964.

[7] Ciro GC, Dugardin F, Yalaoui F, Kelly R. Open shop scheduling problem with a multi-skills resource constraint: a genetic algorithm and an ant colony optimisation approach. International Journal of Production Research. 2016;54(16):4854–4881

[8] L. Di Gaspero, A. Rendl, and T. Urli, "A hybrid ACO+CP for balancing bicycle sharing systems," Hybrid Metaheuristics, pp. 198–212, 2013.

[9] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," Advanced Engineering Informatics, vol. 18, no. 1, pp. 41–48, 2004.