Citi Bike Rebalancing Optimization Using Simulation and Adaptive Learning

IEOR 4404: Simulation

Columbia University in the City of New York

Mili Roy, M.S. Operations Research *mr3964@columbia.edu*

Skand Upmanyu, M.S. Business Analytics *su2236@columbia.edu*

***Abstract*:** The convenience and eco-friendly nature of Citi Bike makes it a popular mode of transportation used in NYC and its surrounding areas. Moreover, it provides a low-cost alternative to traditional public transportation systems. The building blocks of Citi Bike's bike sharing system are stations with fixed docks. In such a system, the revenue generated depends on the availability of bikes for the customers. The customer satisfaction can be measured by the availability of bikes for pick-ups and open docks for returns. The bike inventory at every station during the day is highly affected by customer behavior and travel patterns. As a result, regular rebalancing is carried out using trucks which carry bikes from low demand stations to high demand stations. However, with over 15,000 bikes and 1,000 stations[1], creating a schedule for truck rebalancing to ensure bike and dock availability at every station is a large-scale optimization problem. In this research, a simulation framework is developed to identify the optimal morning configuration of bikes (number of bikes at each station) in order to minimize the customers lost due to bike unavailability and inconvenience caused due to dock unavailability.

I. Introduction

In absence of bike rebalancing process, the availability of bikes at a station is governed by two major factors:

1. The customer arrival process for taking the bikes
2. The customer arrival process for docking the bikes

In such a situation, finding the optimal number of bikes at a station using data analysis of past trips data is not possible. Data analysis can only provide insights about the customers who were able to find a bike and travel to the destination. Similarly, it is also not possible to find insights about dock unavailability using data analysis as there is no distinction between a customer who wanted to reach a given destination and a customer who had to travel to another destination due to dock unavailability.

We can overcome these limitations by developing a simulation model for Citi Bike and changing the input parameters (customer arrival rate, destination probabilities, travel time etc.) and track the number of customers lost and docks unavailable for a given morning configuration (number of bikes at each station at the starting of the day). Once we have the optimal morning configuration which achieves the best trade-off between customers lost and docks unavailability, we can schedule trucks at the end of the day to recreate the optimal configuration for the start of the next day.

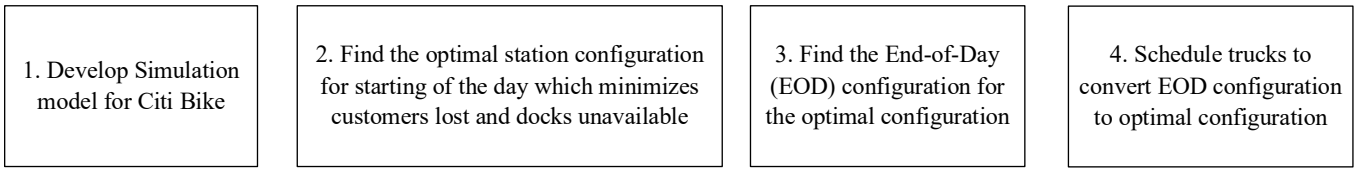


Figure 1.1 Process flow

[1] Citi Bike: https://www.citibikenyc.com/

II. Value Proposition

Finding the optimal configuration for Citi Bike will be beneficial for the company as it will:

Revenue	1. Increase revenue by capturing the customers who were not able to book a ride due to bike unavailability
Customer Satisfaction	2. Increase customer satisfaction by decreasing the inconvenience caused due to bike and dock unavailability
Brand Value	3. Improve brand value for Citi Bike by making it a more reliable mode of transport due to increased availability of bikes

III. Data Overview

For estimating the parameters of our simulation model, we used the publicly available Citi Bike's trip data from August 2020 to October 2020. Below is a brief snapshot of the data:

Time frame	August 2020 - October 2020
No, of days	92
No, of trips	6,867,174
No. of active stations	1,166
No. of unique bikes	20,568
No. of docks	36,106
Source (trips)	https://www.citibikenyc.com/system-data
Source (station capacities)	https://gbfs.citibikenyc.com/gbfs/en/station_information.json

Note: The above data only captures the trips for which a customer was able to find a bike at a given station. Since the customers who are lost as a result of bike unavailability do not have a digital footprint, we need to make some assumptions in order to model the actual arrival rate for a given station. For this purpose, we have assumed that the actual arrivals are 5% higher than the ones that are captured in the trips data above.

III. Simulation Model

In order to identify the number of customers lost and docks unavailable for a given starting configuration of all stations, we created a simulation model for Citi Bike which models customer arrival at every station and simulates the trips to the destination. Below is a brief summary of the simulation model processes:

1. **Customer arrival:** A Non-Homogenous Poisson Process (NHPP) for each station. (1,166 processes).
   - **1.1.** Each NHPP is modeled using a piece-wise function by binning the 24 hour time period of a day into 8 bins (12:00am – 2:59am, 3:00am – 5:59am…).
   - **1.2.** The arrival rate (λ) is considered to be constant in a given time bin and is estimated using past trips data.
   - **1.3.** The arrivals are generated using Thinning Method for a NHPP
   - **1.4.** Upon arrival, customer checks the availability of bikes. If a bike is available, customer books a ride in the process below, otherwise, the customer is tracked as "lost".

2. **Book a ride and travel:** Simulates the customer traveling to the destination.
   - **2.1.** Samples a destination and travel time for the customer using parameters from the past trips data

2.2. The destination is samples using inverse transform of the PMF and efficiency is improved by sorting the PMF and outcomes before using inverse transform method

2.3. Travel time between each station follows a normal distribution: $\sim N(\mu, \sigma)$ for which parameters are estimated using past trips data.

3. **Docking the bike:** Simulates the docking process. If a dock is available, the docking process is complete, otherwise, is tracked as "dock not available" and customer travels to the geographically nearest station until the customer is able to find a dock for the bike.

   3.1. The distance between two stations is calculated using the Geodesic Distance[2] on an ellipsoid. The latitude and longitude of the stations are used to calculate this distance under the assumption that the station which is closest using the Geodesic distance is also closest in the real-world road system.

The availability of bikes and docks at each station is stored in a dataframe which gets updated each time a customer books a ride and docks a bike. For illustration, the simulation system setup looks as follows:

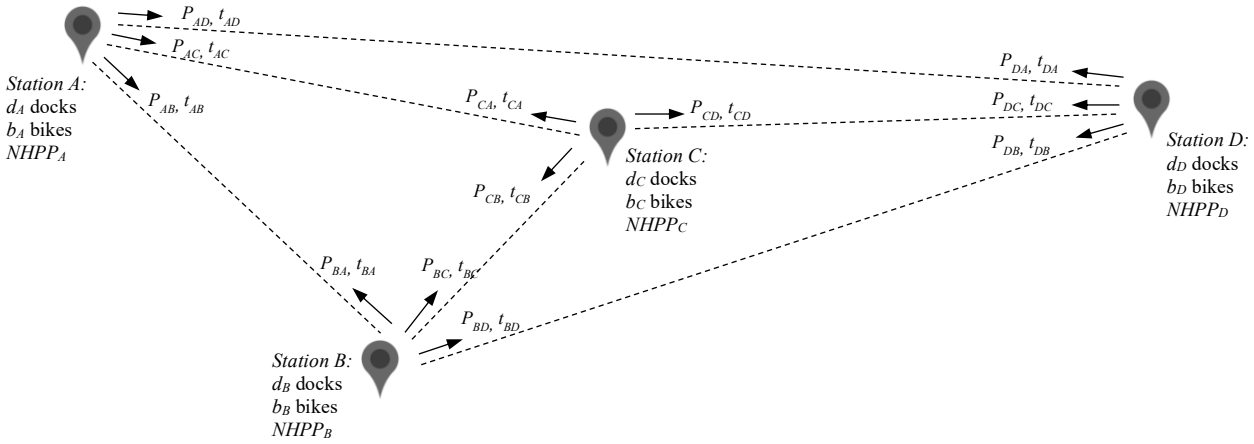


Figure 3.1 Simulation system setup

The dataframe below keeps a track of docks and bikes available at each station.

Station ID	Station Capacity	Bikes Available
A	d_A	b_A
B	d_B	b_B
C	d_C	b_C
D	d_D	b_D

Table 3.1 Tracking docks and bikes at each station during simulation

The number of docks available at station i is calculated as (Station Capacity – Bikes available) = $d_i - b_i$
The station capacity for all stations remains constant as the total number of docks does not change during the day. However, the number of available bikes changes twice for each trip from station i to station j

- When a customer books a ride from station i:
  $b_i = b_i - 1$
- When a reaches station j and docks the bike:
  $b_j = b_j + 1$

[2] Geodesic distance: https://en.wikipedia.org/wiki/Geodesics_on_an_ellipsoid

At each station, the process flow of customer arrivals and booking rides is visualized below. The outputs of simulations are shaded in grey.

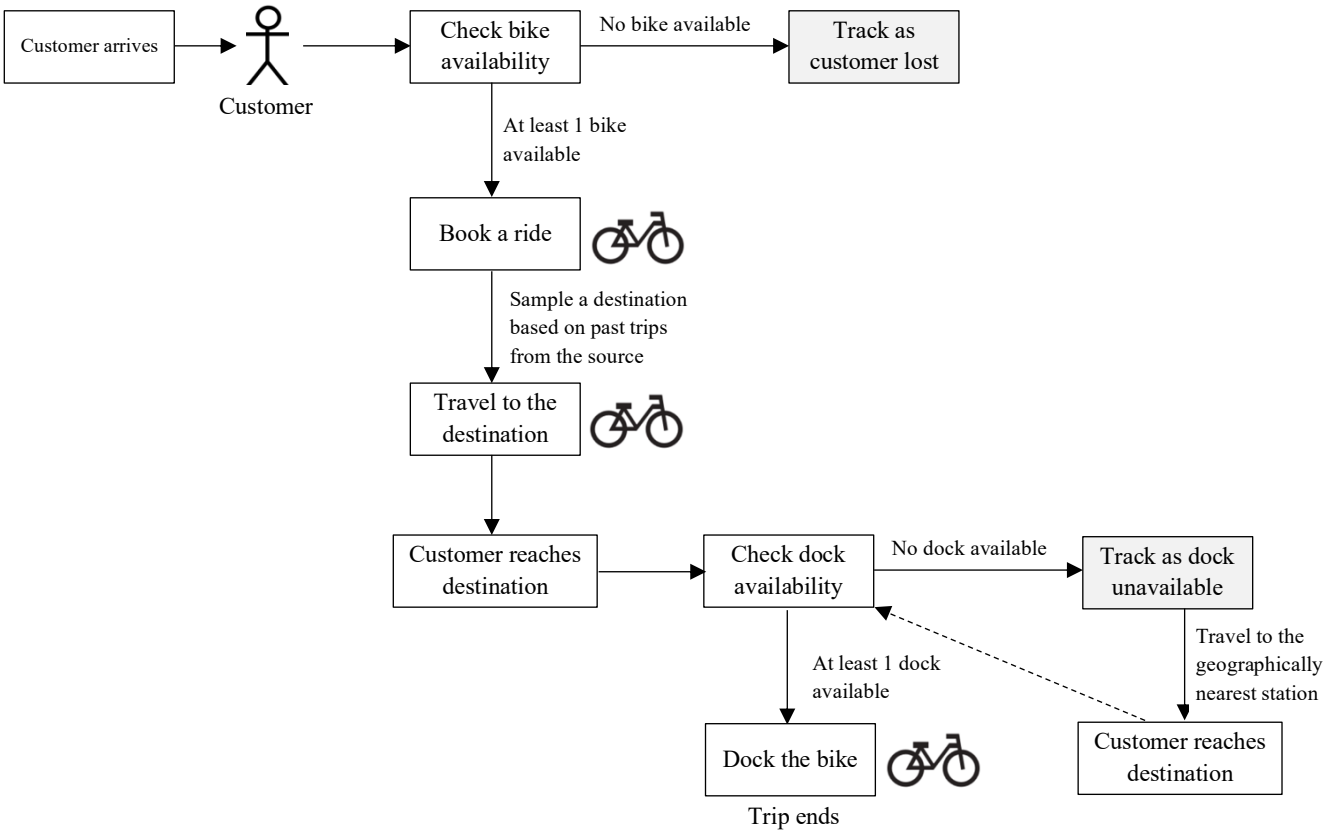


Figure 3.2 Simulation process at each station

IV. Estimating Model Parameters

The following parameters are estimated using the trips data and are used as inputs to the simulation model described above.

4.1 Customer Arrivals

The customer arrival process is modeled as a Non Homogenous Poisson Process where the arrival rate (λ) follows a piece-wise funciton. In other words, the arrival rate (λ) is considered to be constant in a given time bin where bins are created by splitting the 24 hour time interval into 8 bins (12:00am – 2:59am, 3:00am – 5:59am…). It is important to have different λ for different time of the day as we expect different arrival rates at different times of the day (more demand during the day and less demand at night).

The following formula is used to estimating λ for a between t_1 and t_2:

$$\lambda(i, t_1, t_2) = \frac{Total\ arrivals\ at\ station\ i\ between\ t_1\ and\ \ t_2\ for\ all\ days}{Total\ no.of\ days\ in\ the\ data}$$

As an example, for station *A* and time period *9:00am* to *12:00pm*, λ is calculated as follows:

$$\lambda(A, 9{:}00am, 12{:}00pm) = \frac{Total\ arrivals\ at\ station\ A\ between\ 9{:}00am\ and\ 12{:}00pm\ for\ all\ 92\ days}{92\ days}$$

As a result, we have 8 values of λ for each station corresponding to the 8 time bins.

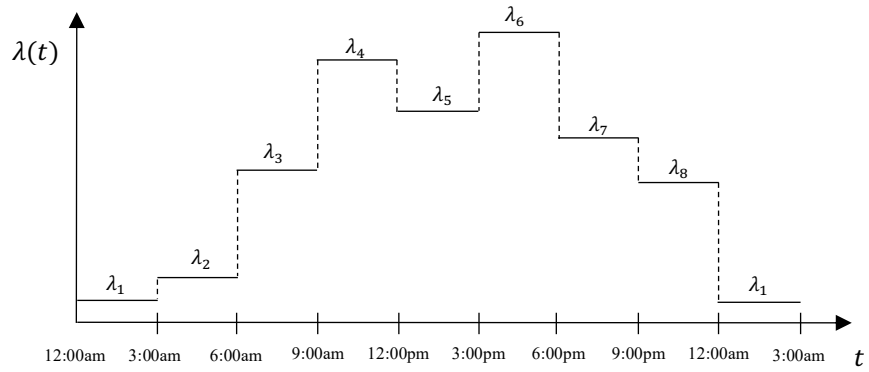


Figure 4.1 Piece-wise arrival rate for stations

The arrivals from the above piece-wise function are generated using **Thinning Method**.

4.2 Destination Probabilities

At each arrival at a station, we need to sample a destination for the customer in order to simulate the travel process. This destination is samples using a Probability Mass Function (PMF) which is estimated using all past trips which originated from that station. The formula used to derive the PMF is shown below:

$$P(i, j, t_1, t_2) = \frac{Total\ trips\ from\ station\ i\ to\ j\ between\ t_1\ and\ \ t_2\ for\ all\ days}{Total\ trips\ from\ station\ i\ between\ t_1\ and\ \ t_2\ for\ all\ days}$$

As a result, we have 8 PMFs for each station corresponding to the 8 time bins. Since customers tend to travel towards their office during the morning time and towards residential areas during the night time, it is important to have different PMFs for different time of the day

4.3 Travel Time

The travel time between each station is assumed to follow a normal distribution. We expect some riders to ride slow, while some might ride really fast. However, most riders have similar riding speed and therefore, a normal distribution should be able to model such a pattern effectively. The travel time betwee station *i* and *j* can be described as:

$$Travel\ time\ (i, j) \sim N(\mu, \sigma)$$

The parameters μ and σ are estimated using past trips between two stations using the formula described below:

$$\mu(i, j) = \ Mean(All\ travel\ time\ for\ trips\ from\ station\ i\ to\ j)$$
$$\sigma(i, j) = \ Standard\ Deviation(All\ travel\ time\ for\ trips\ from\ station\ i\ to\ j)$$

Note: In rare cases where a negative travel time is sampled from the above normal distribution, the sample is ignored and a new sample is generated until we observe a positive sample

4.4 Distance between stations

In order to identify the closest station when a customer does not find a dock available, we need to calculate distances between each station pair. For this purpose we have used the Geodesic Distance which is the shortest distance between two points on the surface of an ellipsoid (earth). The latitudes and longitudes of the two stations help in determining the coordinates of the two points in Geodesic Distance. The distance between station *i* and *j* can be described as:

$$distance(i, j) = \ Geodesic\ Distance(Lat(i), Lon(i), Lat(j), Lon(j)))$$

V. Improving Efficiency of Simulation Model

Our Simulation model involves one arrival process for each station which results in 1,166 process running simultanesouly during simulation. Moreover, each process further initiates multiple other process which sample a destination using inverse transform, perform thinning method, sample from normal distribution for travel time, and try to dock a bike until a dock becomes available. When we created this model for the first time, it took 3 hours to perform one round of simulation for the entire day (24 hours). Therefore, it was important to optimize the model in order to perform multiple rounds of simulation for reliable results.

After analyzing each line of code for efficiency, the bottleneck was extracting the PMF for sampling the destination. The probabilies of going from station *i* to station *j* and time bin *t* were stored in a dataframe of dimensions (N station x N station x Time Bins) x 1 = (1166 x 1166 x 8) x 1. One extraction of PMF from this dataframe for station *i* and time bin *t* took about 0.02 seconds.

However, reshaping the dataframe to dimensions (N station x Time Bins) x N station = (1166 x 8) x 1166 reduces the time for extraction by a huge factor of 60. After further other tweaks, we were able to achieve a runtime of 100 seconds for one round of simulation for the entire day (24 hours).

VI. Exploring Policies for Best Policy Identification

For a given starting station configurations, we get the outputs as follows from the simulation model:

1. Number of customers lost at each station and each time bin
2. Number of docks unavailable events at each station and each time bin
3. The end of day configuration at each station

In order to find the most optimal stations configuration, we need to iterate over multiple bike configurations to identify which configuration leads to the minimum number of customers lost / docks unavailable.

However, this process requires iteratiing over very large number of station configurations. Let's try to find out how many different configurations are possible.

Let x_i indicate the number of bikes available at station *i* at the starting of the day and c_i indicate the station capacity. For a total of 20,568 Citi Bikes and 1,166 stations, we have:

$$x_1 + x_2 + x_1 + x_2 + x_3 \dots + x_{1166} = 20{,}568$$
$$0 \leq x_i \leq c_i \quad \forall\, i \in \{1, 2, 3, \dots 1166\}$$
$$x_i \in \mathrm{I} \quad \forall\, i \in \{1, 2, 3, \dots 1166\}$$

The above set of equations is difficult to solve and has more possible solutions than a normal computer can iterate over. Moreover, with out simulation model taking 100 seconds for a single round of simulation, it will take forever to iterate over all possible configurations.

Therefore, instead of iterating over all configurations, we need to algorithmically update our starting configuration based on the simulation model outputs in order to minimize customers lost. This is why we introduce the concept of Adaptive Learning for iteratively updating our starting configuration to minimize customers lost and unavailable docks.

VII. Adaptive Learning

7.1 Introduction

Let us imagine that we only had two Citi Bike stations in NYC. In this case, if we find that at the end of the day, Station A lost 15 customers between 12:00am and 3:00am and Station B lost 0 customers between 12:00am and 3:00am, a simple solution to rebalance would be to move a few bikes, *N* from Station B to Station A (assuming that rebalancing only takes place at midnight). The idea is to capture customers which were lost at the starting of the day as this is the time which we can be directly impacted by rebalancing.

Expanding this idea to multiple stations, let us imagine that the model output of the simulation model looks as follows. We sort the customers lost dataframe in descending order based on customers lost at each time bin giving highest priority of sorting to 12:00am – 3:00am time bin, seconds priority to 3:00am – 6:00am and so on (based on the idea of capturing customers which were lost at the starting of the day). The updated configuration for the next iteration is obtained by moving *N* bikes from Station Z to Station A, $N - k$ bikes from Station Y to Station B, $N - 2k$ bikes from Station X and Station C and so on.
where:
N is defined as the learning rate
k is the reduction in learning rate for middle rows

This achieves the following:

- The stations which lost most customers at the starting of the day receive the greatest number of bikes
- The stations which lost few customers at the starting of the day also receive some bikes but fewer than the stations which lost more customers

Station	Capacity	Customers Lost (12:00am – 3:00am)	Customers Lost (3:00am – 6:00am)	Customers Lost (6:00am – 9:00am)	Customers Lost (9:00am – 12:00am)
A	20	45	15	3	23
B	25	40	36	3	5
C	15	39	12	4	5
…	…	…	…	…	…
X	20	5	1	3	4
Y	15	1	30	11	21
Z	20	0	12	16	0

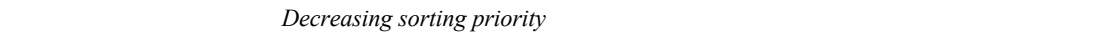


Table 7.1 Updating configurations using sorted Customers Lost Dataframe

After updating the configuration, another round of simulation is performed to get the model outputs. The new model outputs are used to update the configuration again. This algorithm is expected to decrease the number of customers lost monotonically in absence of any randomness. However, as we have lots of randomness in our simulation model, we might see observe some fluctuations but the overall trend is expected to be decreasing with the number of iterations.

The idea is further expanded to the docks unavailability where we move bikes from the stations where most dock unavailable events occurred to the stations where least dock unavailable events occurred.

The idea is even further expanded to minimize both customers lost and docks unavailable at the same time using a weighted loss creation. The technical details are covered in the next section.

7.2 Technical details

1. *Learning Rate:* The learning rate (no. of bikes to be moved at a time between two stations is a function of the position of the station in the sorted customers lost dataframe. With *n* station in total, the formula for the number of bikes to be moved between station *i* and *n - i* is calculated as:

$$No.of\ bikes\ to\ be\ moved\ (i, n-i) = Learning\ Rate\ \text{x}\ \frac{\frac{\#\ Stations}{2} - i}{\frac{\#\ Stations}{2}}$$

When *i = 0* (most customers lost), no. of bikes to be moved is maximum. When *i = # Station / 2,* least number of customers are lost and no. of bikes to be moved is 0.

2. *Insufficient bikes/docks:* In cases where the number of available bikes/docks is less than the number of bikes to be moved, the maximum possible number of bikes are moved and the remaining bikes are moved to/from the next station.

3. *Minimizing Customers Lost and Docks Unavailable together:* For a two-way optimization of minimizing customers lost and docks unavailable, a weighted loss is created at each station and each time bin as:

$$Loss\ = w_{customer\ lost}\ \text{x}\ Customers\ Lost(Normalized) - w_{docks\ unavailable}\ \text{x}\ Docks\ Unavailable\ (Normalized)$$

where:
$w_{customer\ lost}$ is the importance (weight) given to customers lost
$w_{docks\ unavailable}$ is the importance (weight) given to dock unavailable

The above created loss helps in moving bikes to the station if customers are lost and moving bikes out of the station if docks are unavailable. The normalization is performed to bring the two terms in the same scale.

4. *Configuration for initializing adaptive learning:* For initializing the adaptive learning algorithms, we need an initial configuration which will be used for first iteration. For this purpose, we distributed bikes to all stations in equal proportion. With 20,568 bikes and 36,106 total docks, each station was filled with approximately 57% capacity.

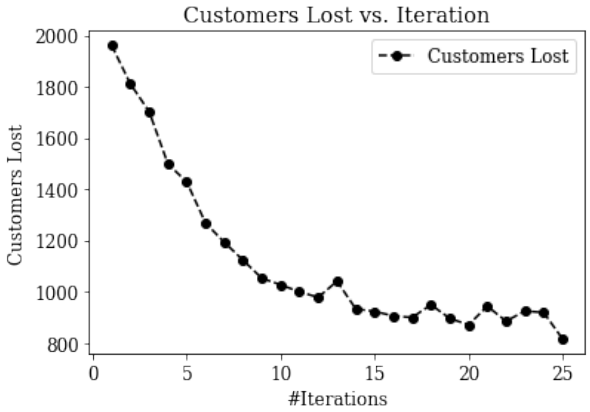
5. *Adaptive learning parameters:*

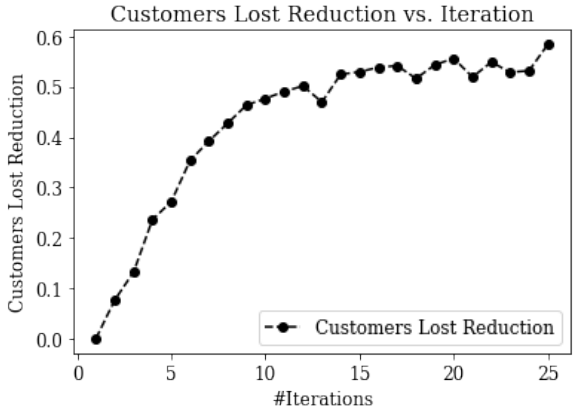
Name	Description	Value
learning_rate	The maximum number of bikes that can be moved between two stations	1
n_sim	Number of simulations to be performed for each iteration before updating. The mean of the loss is used for updating the configuration (increases robustness)	10
n_iter	Total number of iterations to be performed (stopping criteria)	25
wt_cust_lost_loss	Importance (weight) for customers lost	0, 0.5, 0.75, 1
wt_docks_na_loss	Importance (weight) for dock unavailable	0, 0.5, 0.25, 1

VIII. Results

1. *Minimizing Customers Lost:*

$w_{customer\ lost} = 1$
$w_{docks\ unavailable} = 0$

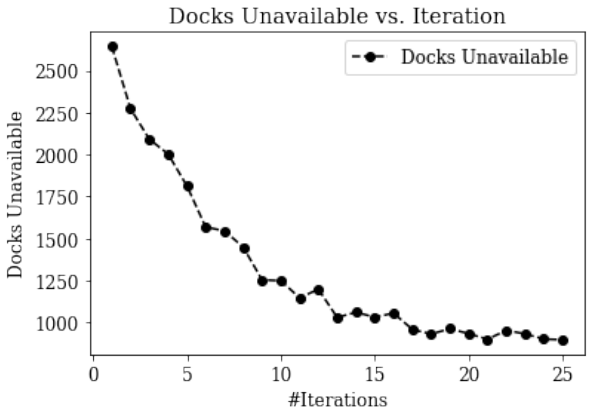


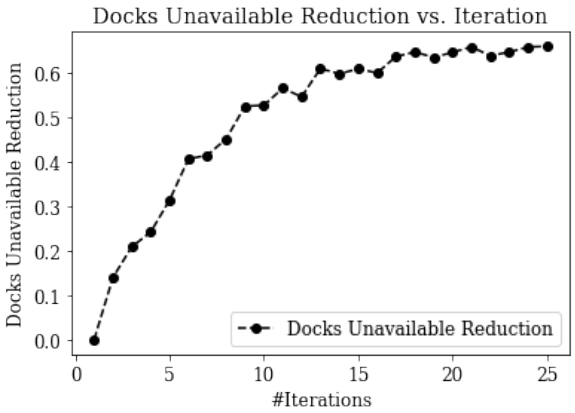


- The maximum reduction is achieved at iteration 25
- There is no significant reduction in customers lost after 15 iterations.
- The Expected value of customers lost is reduced from 1961.4 to 816.1
- The Reduction achieved in Expected value of customers lost is 58.4%
- However, the number of docks unavailable increases from 2555.6 to 6155.3

2. *Minimizing Docks Unavailable:*

$w_{customer\ lost} = 0$
$w_{docks\ unavailable} = 1$

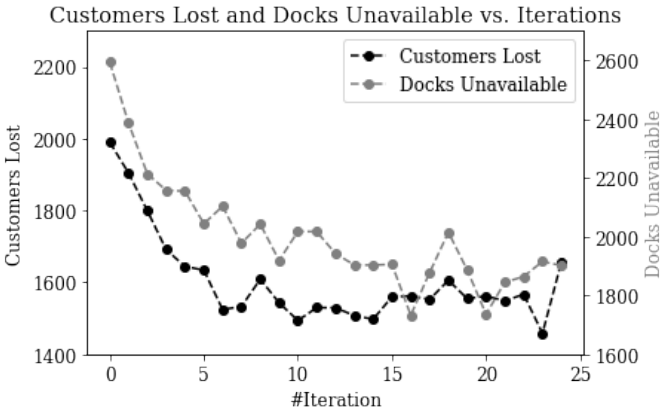


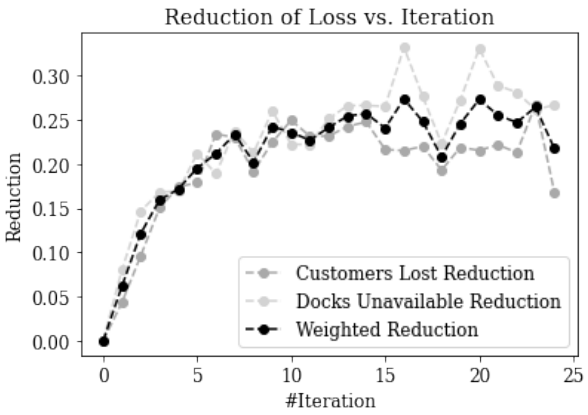


- The maximum reduction is achieved at iteration 25
- There is no significant reduction in customers lost after 20 iterations.
- The Expected value of docks unavailable is reduced from 2645.0 to 896.7
- The Reduction achieved in Expected value of docks unavailable is 66.1%
- However, the number customers lost increases from 2025.0 to 3248.8

3. *Minimizing Customers Lost and Docks Unavailable equally:*

$w_{customer\ lost} = 0.5$
$w_{docks\ unavailable} = 0.5$

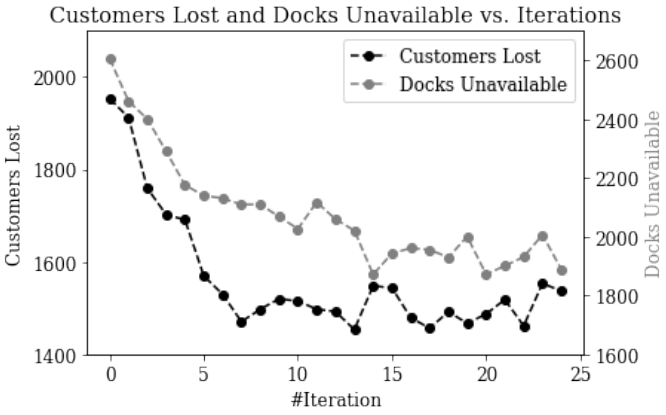


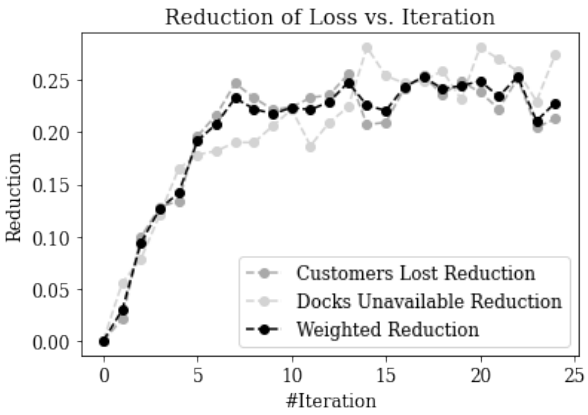


- The maximum weighted reduction is achieved at iteration 17
- There is no significant reduction in customers lost after 10 iterations.
- The Expected value of customers lost is reduced from 1990.1 to 1562.0
- The Expected value of docks unavailable is reduced from 2593.2 to 1732.5
- The Weighted Reduction achieved is 27.35%

4. *Minimizing Customers Lost and Docks Unavailable with unequal importance:*

$w_{customer\ lost} = 0.75$
$w_{docks\ unavailable} = 0.25$





- The maximum weighted reduction is achieved at iteration 23
- There is no significant reduction in customers lost after 10 iterations.
- The Expected value of customers lost is reduced from 1952.8 to 1461.4
- The Expected value of docks unavailable is reduced from 2604.7 to 1930.9
- The Weighted Reduction achieved is 25.34%

IX. Analysis of Optimal Configuration for Developing Business Strategies

We analyzed the optimal configuration to identify some business strategies and insights. For the optimal configuration for minimizing the customers lost, we found that the most filled stations are the ones near residential areas and close to the river. While the least filled stations are close to schools and offices where the bikes are carried by the customers in the morning and therefore, these stations do not need to be filled with bikes. This also includes some stations in South Bronx where Citi Bike has recently expanded their coverage[3] and people in these areas are not used to using Citi Bike yet leading to less demand at these stations.

Top 5 most filled stations in optimal configuration:

Station	Total docks	Bike filled	Insights
Bank St & Hudson St	27	27	West Village residential area. High demand in morning but no bikes reaching the station.
1 Ave & E 30 St	28	28	A hospital station near the east river. More demand for morning rides. Also, more demand among healthcare workers after COVID-19.
S 5 Pl & S 5 St	43	43	Residential area. High demand in morning but no bikes reaching the station.
Madison St & Clinton St	27	27	Station next to a public housing complex. Hence, increasing demand. Also close to the river which riders prefer.
Stanton St & Mangin St	19	19	Station next to a public housing complex. Hence, increasing demand. Also close to the river which riders prefer.

Top 5 least filled stations in optimal configuration:

Station	Total docks	Bike filled	Insights
Washington Ave & E 174 St	19	0	Near schools where students bring Citi Bikes to these stations.
Boston Rd & Prospect Ave	19	0	South Bronx where customers are not used to using Citi Bike yet.
Fulton Ave & E 168 St	21	0	
E 167 St & Franklin Ave	20	0	
Lafayette Ave & Hunts Point Ave	22	0	

These insights can help Citi Bike in identifying the need for promoting their services in South Bronx and identifying high demand areas in the morning like residential areas and stations close to the river.

[3] Source: https://ny.curbed.com/2020/4/29/21241426/citi-bike-expansion-coronavirus-bronx-upper-manhattan-biking

X. Cost Analysis

As Citi bike is a business, it will need to do some cost-benefit analysis in moving the bikes from the typical EOD configuration to the ideal morning configuration. As it costs Citi Bike to use trucks to move bikes from station A to station B, we want to make sure that it is profitable by examining the revenue generated from rebalancing. We will use a simplified cost-revenue structure to determine the profitability analysis.

Let us assume that we are only focused on reducing customers lost we choose the optimal configuration for this objective. Using the optimal configuration, we run 10 rounds of simulation to get mean EOD configuration for the given optimal configuration. We need to analyze the difference between the EOD configuration and optimal configuration to identify the parts of the rebalancing that are profitable for the company.

Citi Bike uses trucks to rebalance the stations. Let us assume that *each truck has a capacity of 15 bikes* and *each tuck trip costs $20 between two stations*. From Citi Bike's website, we note that the *revenue generated from a single 30 minutes ride is $3*.

For each station which observes an increase in the number of bikes, we calculate the number of trucks required to increase the bikes at that station, calculate the cost of booking these trucks and compare it with the revenue that will be recovered from the customers that will be recovered with this rebalancing. If the revenue recovered is higher than the cost of the truck, we make the decision to go ahead with rebalancing at that station. The calculation for 4 example stations is shown below:

Station ID	Total Docks	Optimal bikes	EOD bikes	Δ Increase bikes	#Trucks	Truck cost	Cost recovered	Transfer decision	Profit
223	33	33	19	14	1	20	42	Yes	22
82	27	18	13	5	1	20	15	No	0
83	62	35	36	0	0	0	0	-	0
173	70	42	17	25	2	40	75	Yes	35

Under these assumptions and using the sum of the Profit column, we are able to **increase profit by $2,158 per day.**

Assumptions:

- One truck transfers bikes only between 2 stations (no truck reuse)
- The number of customers recovered at a station is equal to the number of bikes increased at the station
- No customers are lost when moving the bikes out of a station (optimal configuration is better than EOD configuration)
- A new truck is required to for additional bikes even if the number of bikes to be transferred is 1 bike higher than the truck capacity
- Company will go ahead with the transfer between 2 stations, if it helps earn even $1 from the rebalancing

XI. Conclusion

Simulation made it possible to determine the optimal bike configuration using historical data and assumptions. Citi Bike can use this information to improve bike availability, determine staffing, develop initiatives like Bike Angels[4].

However, there are certain limitations for which improvements can be made to the model for further analysis:

- Incorporate maintenance of bikes and stations into the model as bikes as both can break and require maintenance
- Implement truck deployment and rebalancing into the simulation model (perhaps during different times of the day)
- Determine the optimal route for the trucks to take to rebalance the stations

[4] Bike Angels: https://www.citibikenyc.com/bikeangels

XII. Links to Resources and Contributions

1. Code: https://colab.research.google.com/drive/1JMP18fz1Sgjm8PE6CIfCpJDHaa9Fx3d5?usp=sharing
2. Data Used:
   - Trips: https://www.citibikenyc.com/system-data
   - Station capacities: https://gbfs.citibikenyc.com/gbfs/en/station_information.json
3. GitHub: https://github.com/skandupmanyu/Citibike_Simulation
4. Video: https://drive.google.com/file/d/18NR0707r2bNs5AUTuv385brMwEyW3hVq/view

Contributions:

S. No	Contribution	Team Member
1.	Project Ideation	Mili Roy
2.	Project Proposal	Mili Roy
3.	Estimating Parameters from Data	Mili Roy and Skand Upmanyu
4.	Customer Arrival Process	Mili Roy
5.	Customer Travel and Docking Processes	Skand Upmanyu
6.	Optimizing Time Efficiency of Simulation Model	Skand Upmanyu
7.	Adaptive Learning	Skand Upmanyu
8.	Business Strategies and Insights from Optimal Configuration	Mili Roy
9.	Cost Analysis	Mili Roy and Skand Upmanyu
10.	Video Narration	Mili Roy
11.	Video Editing	Skand Upmanyu
12.	Project Report	Mili Roy and Skand Upmanyu

XIII. References

1. Y. Jin, C. Ruiz, H. Liao and H. Pierson, "A Simulation Framework for the Rebalancing and Maintenance of Bicycle-sharing Systems," 2019 Winter Simulation Conference (WSC), National Harbor, MD, USA, 2019, pp. 819-829, doi: 10.1109/WSC40007.2019.9004805.

2. Jian, Nanjing et al. “Simulation optimization for a large-scale bike-sharing system.” 2016 Winter Simulation Conference (WSC) (2016): 602-613.

3. Tao, S., & Pender, J. (2020). A STOCHASTIC ANALYSIS OF BIKE-SHARING SYSTEMS. Probability in the Engineering and Informational Sciences, 1-58. doi:10.1017/S0269964820000297