Rebalancing Techniques for Station-Based Bike Sharing Systems

by

Dillon Schetley

Bachelor of Science, University of Pittsburgh, 2019

Submitted to the Graduate Faculty of the

School of Computing and Information in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This thesis was presented

by

Dillon Schetley

It was defended on

July 31, 2020

and approved by

Dr. Panos K. Chrysanthis, Professor, Department of Computer Science

Dr. Konstantinos Pelechrinis, Associate Professor, Department of Informatics and Networked Systems

Thesis Advisor: Dr. Alexandros Labrinidis, Professor and Chair, Department of Computer Science

Rebalancing Techniques for Station-Based Bike Sharing Systems

Dillon Schetley, BS

University of Pittsburgh, 2020

With bike-sharing systems that utilize fixed rent and drop off stations becoming popular in cities and metropolitan areas worldwide, the issue of station fill balance becomes apparent. It is important for the user experience and the organization's bottom line that bikes are available at the stations where they are needed and that stations do not become too crowded and thus prevent easy returns. There is not, however, a clear solution of how to perform this rebalancing. Considerations include how to determine the stations that most need to be rebalanced, how frequently to do this rebalancing in the system, and how many resources to expend doing it. Methodologies answering some of these questions have been proposed, but many do not provide all of the answers necessary to fully implement a real-world solution. Additionally, there is no benchmarking tool to fairly compare these rebalancing approaches on a given system.

This thesis proposes exactly this kind of tool in the form of a station-based bike-sharing system simulator. The simulator is modular and provides several parameters to allow the comparison of different systems, historical data, workloads, and rebalancing strategies. To demonstrate its capabilities, experiments were run comparing the effects of individual parameter changes and various combinations of parameter configurations on various metrics, including gross revenue and lost revenue from missed demand. Analysis of these experimental results gives not only a look into the simulation's uses as a comparative tool, but also provides information on alternatives to common predictive rebalancing strategies.

Table of Contents

List of Tables

List of Figures

List of Equations

Preface

I would like to thank my faculty thesis advisor, Professor Alexandros Labrinidis, for his practical guidance, encouraging ideas, and patience with me during the long and obstacle-filled process of working on this thesis.

I would also like to thank the other members of my defense committee, Dr. Panos K. Chrysanthis and Dr. Konstantinos Pelechrinis, for being available on somewhat short notice, as well as their understanding and flexibility while scheduling the defense itself.

I would also like to thank my family, most of all my parents, for supporting me in many ways throughout my life, especially during my research. Their support during long periods of isolation was invaluable.

I would also like to thank my friends and colleagues, for inspiring me and helping me to stay motivated during difficult times, and for providing a meaningful escape from my work when I needed it.

Finally, I would like to thank the University of Pittsburgh's Department of Computer Science for giving me the opportunity to streamline my pursuit of a Master's Degree.

1.0 Introduction

The main problem explored in this thesis is that of rebalancing in station-based bike-sharing systems. Before diving into that question, however, there are others to answer first. This chapter will explain why there is need for work on bike-sharing systems at all, especially in rebalancing. Additionally, the goals and main contributions are listed before providing the thesis's organizational structure.

1.1 Motivation

With all of the environmental concerns in today's society and the known significant contribution motor vehicles make to those issues, traveling by bike when possible has become a popular option. Bike-sharing systems have taken advantage of this market. In Pittsburgh, a known example of these systems is the Healthy Ride station-based system, in which the company distributes bikes across various docking stations where customers can pick a bike to rent. For a subscription or a time-based fee, the customer has the freedom to ride and either return the bike to its original location or to leave it at another of Healthy Ride's stations. This freedom can create a problem of *imbalance*: certain stations may be left with insufficient bikes for their customers while other stations are full or overflowing with bikes that may not be used. In response to this, Healthy Ride has a truck go around the stations and move bikes to rebalance the system. Taking this approach, however, raises some questions regarding when to do this rebalancing, which stations to choose, and how many bikes to pick up or drop off from each.

Substantial work has been done on predicting demand in these systems, ranging from reduction methods to Graph Convolutional Networks (GCNs), but not as much work has been done on efficient rebalancing algorithms and how demand prediction may fit into those approaches. Additionally, there are not modular, readily available, open-source tools for simulating and comparing various strategies' effects on potential systems. Proposed algorithms for rebalancing must either be provably better than existing methods or demonstrate impressive performance on a held-out test set of historical data. Finally, existing work does not include much insight into how to carry out proposed rebalancing strategies as far as resources (time, rebalancing agents, etc.) are concerned.

1.2 Objectives and Contributions

In response to these under-explored areas, the goal was to create a framework for the flexible side-by-side comparison of rebalancing methods and use that framework to perform experiments on various rebalancing heuristics.

The final contributions are as follows:

- A full bike system simulator, capable of processing data from different bike-sharing systems and different parameters for rebalancing.
- A collection of simple heuristic-based rebalancing algorithms.
- Possible resource configurations to carry out rebalancing strategies in the Pittsburgh area.
- Experimental results comparing different rebalancing tactics’ costs, effects on the system, and other metrics

1.3 Roadmap

The thesis is divided into the following chapters. Chapter 2.0 will discuss in detail the algorithms and other components used and tested within the simulation framework. Chapter 3.0 will describe the structure of the simulation framework, how it functions, and how it can be used. Chapter 4.0 will define the setup and environment of experiments run, and Chapter 5.0 will show and interpret the results of those experiments. In Chapter 6.0, related work will be described and reviewed in more depth. Finally, Chapter 7.0 contains potential future work to be done in this space and concluding remarks.

2.0 Algorithms and Components

The simulation framework can be broken into various components, some of which can utilize different models or algorithms. This chapter will describe major models and algorithms individually in detail; for how these parts interact, see Chapter 3.0.

2.1 Demand Distribution

To perform simulations of novel scenarios in the bike-sharing system, it is necessary to generate some sort of realistic customer demand, both for where customers are renting bikes (source station) and where they are returning them (destination). Historically, the Poisson distribution has been used and shown to be well suited for modeling departures and arrivals at given stations. Therefore, it seemed a good choice to serve as a model for the underlying true demand within the system. Additionally, the distribution is defined using solely the integer parameter λ, which represents both the mean and the variance. Because of this, it is trivial to fit the distribution based on available bike trip data. The probability mass function for the Poisson distribution is:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \qquad \textbf{Equation 1}$$

An issue with the Poisson distribution, however, is its assumption that the rate of arrivals is constant. Intuitively, bike demand is neither constant across stations nor across all possible times

of the day or year. To solve this problem, days are divided into 72 twenty-minute segments (hereafter referred to as Tslices), and separate Poisson distributions are fitted for each combination of source, destination, Tslice, and day of the year. For example, the value of λ for the combination of [1 2 27 5] would be used to model demand for trips starting at station 1 and ending at station 2 from 9:00-9:20 a.m. on January 5.

Additionally, it is necessary to model the duration of these trips in a way that is representative of reality, but not as an exact copy or overly predictable pattern. Customers renting bikes may ride directly to their destination and end the trip, but some may take lengthy detours before arriving at a different or the same source station. Distributing trip duration randomly according to a range of values may not accurately capture frequent usage patterns or other confounding factors. The simple approach used is to store all durations, given in seconds, found in historical data and index them according to $(source, target)$. When one or more durations are needed for a simulated bike trip, duration(s) are sampled with replacement for the appropriate trip. This encourages the selection of more common trip durations while providing occasional realistic duration alternatives.

2.2 Demand Predictor

Most work surrounding the management of bike-sharing systems, intuitively, includes some element of predicting demand. As detailed in Chapter 6.0, many works have been published on designing and improving models to predict station demand in different ways. Creating these models is not trivial, and attempting to innovate in this space would be outside the scope of this thesis. The state-of-the-art models, however, are often complex, require large amounts of data, and

have been tested on hourly time granularities. Therefore, instead of attempting to build the simulation framework around one of these models, using a simpler model would suffice. A clear choice to predict time-series data per station is a standard Long Short Term Memory (LSTM) network.

Using only historical data to train the predictor was insufficient as the data was too limited. Generating new data for a demand predictor could be unreliable, as the limited historical data may not be representative. This predictor network, however, is only a placeholder meant to perform well as a part of the simulation, not for implementation and use in the real world. Therefore, sampling from the simulation's true distribution (described in Section 2.1) would suffice to provide enough data to train the model to perform in this limited context. Samples from that Poisson distribution are used to create sequences of station fill level shifts per day. For example, station 1 gets demand for 10 outgoing trips, but from all stations only has incoming demand for 8. For this station, the net value would be 2. These values are calculated for all stations, and a training sequence is these lists of net shifts concatenated to represent shifts over a period of days. For this application, the predictor network should also be robust to different workloads. In other words, if it seems that the overall system is experiencing higher or lower amounts of demand, the predictions should reflect that change. To encourage this behavior, training samples were randomly drawn from not only the historical Poisson distribution but also copies of that distribution scaled by {0.5, 0.75, 1, 1.25 … 4}.

Through testing this prediction network, some adjustments were made to improve performance. In practice, the model performed better when the incoming demand portion of the data was scaled by a factor of 0.9. This seemed to be because most trips start and end at the same station, resulting in a net fill change of zero. This can result in the network predicting net zero

change too often, leading to minimal rebalancing. Before being fed to the model, all sequences are standardized to a range of [0 1] and split 80-20 into training and validation sets.

The most up to date network used in the simulator is an LSTM with 100 hidden units feeding into a fully connected regression layer, trained for 100 epochs with mini-batches of 144 using ADAM optimization with an initial learning rate of 0.01 and gradient clipping set at 1. As shown in Figure 1, the final Root Mean Squared Error (RMSE) on the validation set was 6.4118.

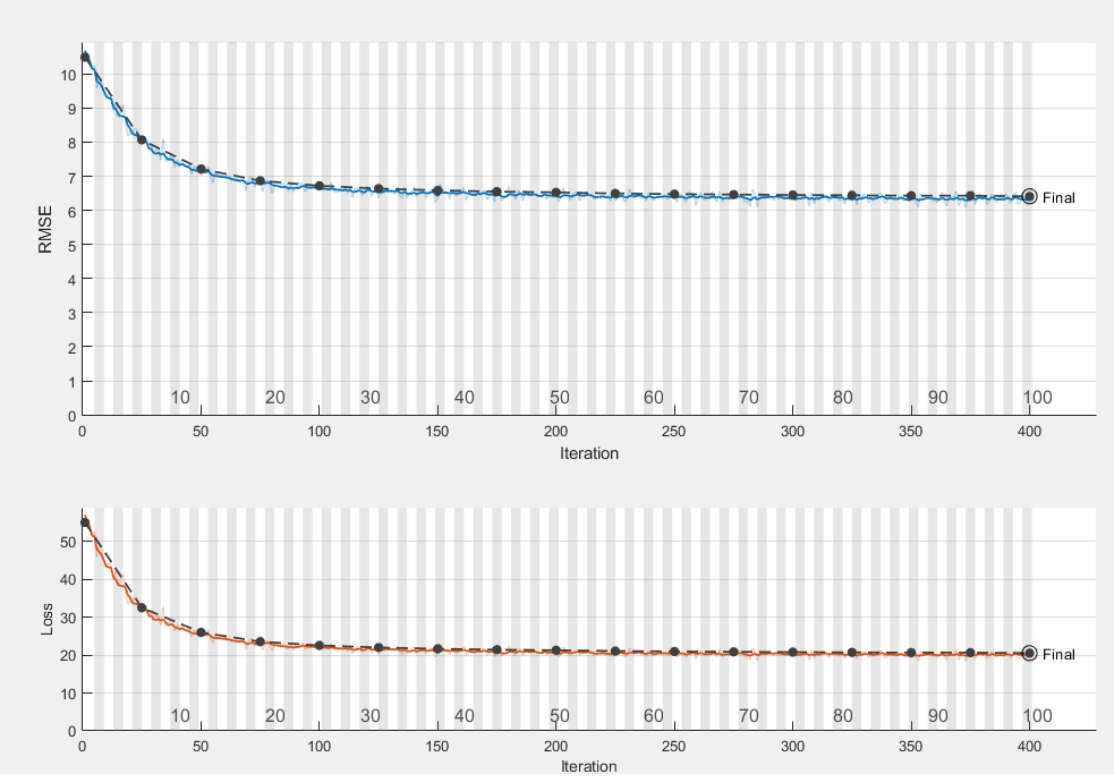

Figure 1 Plot of network training progress

When predicting demand within the simulation, because the model cannot know the true demand distribution a priori, it approximates it by aggregating trips it has seen since its last prediction. Treating these trips like the true demand distribution allows using the same calculations as before to determine station fill level changes. If the system is capable of meeting all demand,

then this calculation will equal the true demand. The same factor of 0.9 is used on these input sequences to skew the model towards demand. The model predicts demand in single-day increments, so in case of predictions spanning multiple days, the model is fed its previous prediction and the individual outputs are summed.

2.3 Rebalancing Algorithms

Part of the simulation's modularity comes from the ability to choose different algorithms to use in the rebalancing process. The algorithms in this section were designed, and so are explained, in pairs: one for finding demand stations where bikes would be deposited and one for finding supply stations where bikes will be withdrawn. The supply stations are chosen second, and so supply station algorithms can utilize the total number of bikes needed. For more information on how these algorithms are used, see Section 3.3.

2.3.1 Using a Prediction Model

The first pair of algorithms rely on the demand predictor described in Section 2.2. The output of the predictor is the expected change in fill levels over some number of days. Combining that with the current station fill levels gives an expected value for the fill levels after the days have elapsed. If a station is expected to be at or below a zero fill level (in other words, if the expected fill is not positive), it will be chosen as a demand station. The required number of bikes at each station returned by the algorithm will be the number needed to reach an expected fill level of one

or as many bikes as can fit in the station. Therefore, the final equation for bikes needed at station s with predicted fill P and fill level F is:

$$D_s(F_{curr}, P) = \begin{cases} \min(F_{max} - F_{curr}, F_{curr} - P + 1), & F_{curr} - P < 1 \\ 0, & \text{otherwise} \end{cases} \quad \textbf{Equation 2}$$

In order to find supply stations to provide the necessary bikes, the paired algorithm uses a similar process. If the expected fill value is positive, the prediction is that the station will have a surplus. Starting from the station with the greatest surplus, the algorithm takes all but two bikes from each station until the sum is equal to the number of bikes needed for demand. The decision to leave at least two bikes at every station is a design decision that worked best while tuning the simulator. Therefore, the final equation for bikes provided by station s is simply:

$$S_s(F_{curr}, P) = \max(F_{curr} - 2, 0) \quad \textbf{Equation 3}$$

2.3.2 Using Thresholds

Relying on demand predictors, even the state of the art, may introduce error into the system through inaccurate predictions and cause unreliable rebalancing. Using a deterministic heuristic might help to make rebalancing more stable. One such heuristic is using fill level thresholds to categorize stations. The demand algorithm uses a minimum fill ratio parameter. For example, a value of 0.4 would cause any station under 40% full to be considered a demand station. The algorithm would then return the number of bikes necessary for the station to be above this ratio.

The supply station algorithm works similarly, using a separate maximum fill parameter. Any stations over, for example, 90% full would provide bikes until they fit under the limit. Using these algorithms in tandem creates an allowed fill interval within which, ideally, all stations would fall. The primary avenue to introduce error into this approach is through poorly selected minimum and maximum fill values. For instance, if there were not enough bikes in the system to keep all of the stations above 40% full, the goal for this rebalancing method can never be achieved. This could lead to bikes being moved between stations without being able to satisfy the threshold or spreading the bikes evenly enough in the system that no station exceeds the supply station threshold and no rebalancing can be done.

2.3.3 Using Relative Fill Levels

Another deterministic method involves sorting stations by their raw fill levels and using the top and bottom "classes" of stations. Given a particular portion of the stations (e.g. 0.1), the stations that fall in that bottom portion will be selected as demand stations. For this demand algorithm, demand stations are returned with the number of bikes necessary to fill them up. The supply algorithm selects stations in the same way, by taking the top portion given as supply stations. The number of bikes taken from each station, however, is determined by the number of bikes in the supply station with the lowest fill level. That station has all but one bike taken, and each of the other supply stations provides the same number of bikes. Through this process, bikes are redistributed from highly-populated stations to those with the fewest.

2.4 Path Algorithms

The information of which stations have excess supply or excess demand is not sufficient to implement a rebalancing strategy. There needs to be a route between them for an agent to follow. In this final section, the algorithms responsible for organizing those demand and supply station lists are explained and analyzed.

2.4.1 Motivation

Once the stations to visit and the number of bikes to take or leave at each have been determined, rebalancing implementations need to be able to determine a path between all of these stations such that the bikes can actually be moved. While there is much need and availability for optimization here, finding an optimal pathfinding algorithm is not the focus of this problem. The challenge of finding the best path to do all necessary rebalancing is a type of Traveling Salesman Problem with weights and constraints. Unfortunately, a standard Traveling Salesman Problem algorithm would not work in this context. Limitations such as rebalancing truck capacity and ensuring that agents do not arrive at demand stations with no bikes to deliver place constraints on optimal paths. There were not obvious implementations of efficient solutions that fulfilled all of the necessary conditions in this specific problem, so two simple path algorithms were designed to function in the simulation. There is one small difference between them: one algorithm allows for stations to be revisited if insufficient bikes were moved on the first visit, whereas the other does not.

2.4.2 Station Ordering

Both algorithms use Prim's algorithm to iteratively build a minimum spanning tree. Prim's algorithm works by adding the smallest edge that would connect a new node to the tree. For this problem, nodes are the stations to be visited and edge weights are the distances between them. Ignoring constraints, when edge weights satisfy the triangle equality (as station distances do), in-order traversal of the minimum spanning tree well-approximates optimal solutions for Traveling Salesman Problems (Cormen et al. 2009). Due to the constraints, however, the algorithm must be modified. For each node considered, the algorithm checks if traveling to this node next would overflow or underflow the rebalancing truck. If that is the case, the edge weight receives a penalty of an additional 50% of its original value. This encourages, but does not require, the algorithm to select stations that can be handled fully at the first stop.

Depending on which algorithm is used, the consequences of selecting a station that cannot be fully handled differ. In the algorithm that allows for revisiting stations, the rebalancing agent processes whatever bike moving is possible (e.g. dropping off only 2 out of 5 bikes needed), but does not remove the station from future consideration. It instead adjusts the number of bikes needed for or from that station (in the example, 3 bikes are needed now), and continues normally. Eventually, the algorithm will again choose this particular station and either satisfy or continue to make progress on the supply or demand present. The algorithm that does not allow for revisits bypasses this process by truncating any excess supply or demand that cannot be handled on a station's first visit. The station is removed from future consideration. In practice, the algorithm without multiple visits completes a path more quickly than the algorithm with repetition, but it incurs some decrease in quality from ignoring unsatisfied supply or demand.

2.4.3 Unequal Supply and Demand

Some approaches for selecting stations aim to create exactly equal bikes needed at demand stations and provided from supply stations. Some, especially if using non-paired algorithms, may not always follow this rule. If the path algorithm receives a list of supply and demand stations with demand and supply totals that are not equal, each station's demand is scaled by the ratio of total supply to total demand. For example, if the total supply was 75% of the total demand, then every demand station's number of bikes would be scaled by 75% and rounded to an integer. Stations with demand reduced to zero are removed.

Additionally, sometimes demand or supply algorithms will not return any stations. In these scenarios, the algorithm will return an empty path as either rebalancing is not needed (zero demand stations) or it is not possible (zero supply stations). Both instances are considered normal use cases.

2.5 Summary

To reiterate, there are several algorithmic components at work within the system overall. The demand distribution is handled by a collection of Poisson distributions covering the variation between stations and time of trips. Demand prediction is done by an LSTM neural network trained using samples from those Poisson distributions. Three pairs of rebalancing algorithms are proposed, the first of which uses the demand prediction network. The other two pairs are based on station fill level heuristics. Finally, to organize rebalancing station information into a usable route, two approaches are described as path-finding algorithms: one with station revisits and one without.

3.0 Simulation Structure

One of the main contributions of this thesis is the program for dynamically and flexibly simulating station-based bike-sharing systems. The simulation works by processing bikes moving between stations due to customer rentals or the selected rebalancing methodology. Once the simulation terminates, it can return various metrics as well as a table of all of the customer rentals that occurred within the simulation. The following sections explain the stages of the simulation and how they interact. They also describe how algorithms are applied and what parameters can be changed.

3.1 True Demand and Initialization

The true customer demand for a particular run of the simulation is determined by sampling from the Poisson distribution described in Section 2.1. For a given source station, destination, Tslice, and day, a value t is sampled implying that on that day and in that Tslice, t people will arrive intending to ride a bike from the source station to the given destination. The sum of all of these t values gives the maximum number of trips that could happen.

Healthy Ride does not provide historical data of station fills and so the initial fill level of stations were manually estimated. These values are fixed and were estimated by looking at historical data and the real-time fill levels available on the Healthy Ride website. The total number of these fills, and thus the total number of bikes available to the system, is 528. The system does not rely on this number and so initial fills could be changed if necessary.

3.2 Processing Trips

Using the demand samples, trips are generated in order by day, Tslice, destination, and source. All of the possible trips given by the sample are initialized. Using the stored historical durations of trips from the given source to its destination, a duration is sampled for each trip. Because durations are in seconds, trips with the same duration occurring at the beginning and the end of a twenty-minute Tslice could be completed in different Tslices. Demand samples only provide time information at the granularity of Tslices, so to handle this variation each trip begins at a random uniformly distributed offset within its Tslice.

The system then checks if each trip would occur based on the current state. This state is up to date for and includes trips completed before the latest trip initialized, offset included. The sole determining factor for if a trip can occur is if there are sufficient bikes for that trip individually. If the source station has a bike to use, then the trip can proceed. The source station's fill level is reduced by one and the bike is outside of the system for the trip duration. When the system has progressed in time far enough, the trip will end and the destination's fill level is increased by one, making the bike available again. Trips that are processed successfully will have their start and end timestamps, duration, source station, and destination added to the total trips output by the simulator.

If the station is empty, this counts as missed demand. The system records some information about the trip for overall metrics, but it is not processed. It does not affect the system state at all and no information will be added to the final trips output.

3.3 Rebalancing

The most complex and time-consuming stage of the simulation is processing rebalancing. This is likely due to the other stages performing simpler, often binary checks of the system state, whereas during rebalancing, the system must not only rely on a larger portion of the system state, but, by using that information, must dynamically plan, construct, and process rebalancing events using the provided algorithms. For ease of discussion, rebalancing is broken into two stages: a rebalancing day and a rebalancing session. A rebalancing day is, intuitively, the entire day during which all agents in the system will do rebalancing work. Rebalancing days are broken up into distinct rebalancing sessions. Each rebalancing session is a planned route that one truck will follow to collect and redistribute bikes among certain stations.

3.3.1 Rebalancing Days

Rebalancing days are defined primarily by their frequency and their active hours. The system is built to schedule rebalancing days at consistent intervals (e.g. every day, every other day, weekly, etc.). Therefore, when processing trips on non-rebalancing days, the entire rebalancing stage is bypassed. This includes any demand calculation that would have been done as part of rebalancing algorithms. Active hours refers to the time window in a rebalancing day during which rebalancing agents will engage in rebalancing work.

When scheduling individual rebalancing sessions, the simulation does not strictly adhere to the active hours' endpoint. As each session is processed, the system keeps track of the average time needed to complete a session and will only schedule a new session if the time remaining within active hours is greater than the average session time. If there is not enough time, the

rebalancing day simply ends early. In practice, this does not result in a significant number of lost rebalancing time from pessimistic, large average session averages. This feature was primarily implemented to prevent long rebalancing sessions from consistently or severely extending active hours in a day. Additionally, if a truck would otherwise be idle (as detailed in the next subsection) past the end of active hours, the truck's time idle is capped at the active time remaining.

3.3.2 Rebalancing Sessions

During the active hours of a rebalancing day, any time a rebalancing truck is available the system will create a rebalancing session plan to process. Session plans are created independently between trucks and are based on the current state and estimates of the system. Once the plan is set, the system will run more calculations to determine how long the rebalancing agent will be unavailable and when each session stop should be processed. It is important to note that the session plans are guides for how many bikes to withdraw or deposit from each station, but station and truck fill levels and capacities when each stop is processed may affect actual bikes moved.

3.3.2.1 Planning

When conditions are met to build a rebalancing session plan, the system works through three main components to get the following core pieces of a session plan: the demand stations, the supply stations, and the session path. As mentioned elsewhere, demand stations are stations where bikes will be deposited, supply stations are stations from which bikes will be withdrawn, and the session path gives the best-case instructions for the rebalancing agent to follow.

The system constructs a list of demand stations first. Using the given demand station algorithm, the system determines the stations that need bikes and how many ought to be provided

to each. The total for all demand stations is always stored, in case it is needed by an algorithm later in the process.

Next, the system constructs a list of supply stations to provide the bikes for rebalancing. Same as with the demand stations, the list also includes how many bikes are needed from each station. Some of these algorithms use the total number of bikes needed as a cap, but some will take the extra supply and use it for further rebalancing. It is important to note that while using paired algorithms will always result in disjoint supply and demand station lists, utilizing the pairs is not required. Therefore, in these scenarios, a station might meet the criteria of both the demand and the supply algorithm. Upon reflection, taking bikes from a demand station seemed more likely to cause missed demand than giving bikes to a supply station, so if a station is on both the demand and the supply list, the demand list takes precedence. That station will be removed from the supply list and remain on the demand station list unchanged.

Finally, the system feeds both demand and supply station lists to the path algorithm which constructs a course to move bikes accordingly. Using information about the current system state and individual truck capacity, the path algorithm will aim to create an ordering that interleaves supply and demand stations such that issues surrounding truck over- or underflow are less likely. This plan, however, as mentioned before, may not be perfect and may not exactly reflect the true number of bikes moved.

Additionally, if one of either the demand or the supply station lists is empty, rebalancing cannot occur. In this case, the rebalancing agent will idle for one hour or until the end of active hours. After this idle time has ended, the session planning process will restart normally to determine if the agent will continue to be idle or if the state has changed enough to cause rebalancing.

3.3.2.2 Processing

The actual number of bikes moved and where exactly those bikes end up are determined when the session stops are processed. The session stops are processed in temporal order in the same pool as normal customer trips. Therefore, the exact number of bikes moved will not be calculated until the session stop actually occurs in the system. Here, depending on whether the truck is depositing or withdrawing bikes, the simulation will check if allowing that stop's full transaction is possible given the number of bikes in the truck at that point and the stations fill level. For example, if the plan was to pick up four out of five bikes at station 1, but station 1 only has four bikes, then the rebalancing agent will only pick up three of those four.

Allowing for this deviation from the plan introduces some issues. Changes to the number of bikes in the truck can have ripple effects for the duration of the rebalancing session. To mitigate this persisting issue and try to get back on plan, the rebalancing agent stores its deviation from its planned number of bikes. At each stop, if the truck has too few or too many bikes, there will be an attempt to correct this deviation by dropping off fewer bikes or picking up fewer bikes, respectively. This corrective action has a limit such that, if possible, at least one bike will be moved at a station stop regardless of deviation. Throughout all of these checks and corrections, station fills and bikes in a truck are updated by the stored value of the actual number of bikes moved.

3.4 Parameters

One of the points of emphasis in this simulation is its modularity and flexibility. It is capable of experimenting with different configurations for rebalancing strategies and resources. This is achieved by feeding the system multiple parameters that can be changed between

simulations of the same or different true demand distributions. When using the same true distribution, the system samples from the actual distribution only once through. The system can then reuse that collection of samples under different conditions. The parameters function mostly intuitively and are only briefly described here.

The demand scale parameter is a positive real factor that uniformly scales demand across all days and all stations. Therefore, a demand scale of 2 would result in exactly twice the overall demand, proportionally distributed the same as normally. The default value is, intuitively, 1.

The rebalance rate parameter is one of those which defines rebalancing days. It is a positive integer rate at which rebalancing days occur. For example, a rate of 1 is every day, a rate of 7 is every week, and so on. The default value is 1, or every day rebalancing.

The number of trucks parameter functions as its name implies. It defines the number of identical trucks available for concurrent rebalancing, each of which incurs its own costs. The default value is 1.

The truck capacity parameter also functions intuitively. This is the maximum number of bikes an individual truck can hold at one time, which may limit individual rebalancing agents' overall speed and capability. The default value is 20.

The rebalance bounds parameter is the starting and ending Tslices of active hours for rebalancing days. Therefore, the system is flexible enough to define active hours in twenty-minute chunks, but it is more sensible to rebalance for somewhat normal working hours. The default values make active hours 9:00 a.m. to 7:00 p.m.

The demand method parameter defines the algorithm for selecting demand stations. For more information about the available choices, see Section 2.3. The default value is to use the predictor network to find deficit stations.

The supply method parameter defines the method for selecting supply stations and relevant information about these algorithms can also be found in Section 2.3. The default is to use the pair of the default demand method: use the predictor network to find stations expected to have a surplus.

The path method parameter determines the session-path-building algorithm. This parameter currently has only has two options: to allow station revisiting or not. The default option is to allow for revisiting with a penalty. For a more in-depth explanation, see Section 2.4.

Each of these parameters can be modified individually. There are, however, parameters that interact more closely than others. One example is that, as mentioned elsewhere in this thesis, demand and supply station methods were designed in pairs. Another would be the number of trucks and truck capacity. Having multiple trucks, in some ways, multiplies the capacity. Enabling individual parameterization like this gives significant flexibility for experimentation.

3.5 Summary

To recap, the overall simulator flow begins with sampling the true demand for the simulation's run and initializing stations. Trips are processed individually in order of their start time with rebalancing days and relevant calculations interleaved when necessary. Each of these rebalancing days is made up of zero or more rebalancing sessions, depending on the system state. The end result is a table of trips that occurred in the simulation, as well as a collection of metrics for analysis or experimentation. Finally, all of these processes are shaped by the values of system parameters, all of which possess a default value and may be individually modified.

4.0 Experimental Setup

The primary use for the simulation system described in the previous chapter is to perform experiments to compare the performance of different ways of rebalancing. Therefore, the best way to demonstrate the system's capability is to run experiments on a particular bike-sharing environment and evaluate performance under different configurations. Changing different parameters will affect the system in various ways, as some parameters affect the frequency of rebalancing done or the resources available to do it. Experimenting with individual parameters should demonstrate this. To design a methodology that is efficient and effective, however, it is necessary to explore approaches with different combinations of parameters. Experiments of both types were performed and their results are discussed in the next chapter.

4.1 Data Used

The real world, historical data used for all of the experiments is the publically available Healthy Ride Pittsburgh trip records for the first three quarters of 2019. Each data point contains information such as a bike identification number and customer type, but the columns used by the simulation are the source station ID, destination station ID, start time, duration, and stop time. The experiments were limited to only three quarters of 2019 for a few reasons. Primarily, computing time and resources were somewhat limited, so the reduced window of data processing and generation enabled experiments to be run more quickly and frequently. During the earlier stages of this work, the data for the final quarter of 2019 was not yet available. Because of this, initial

analysis of the data was performed on just the available quarters, showing noticeable fluctuation of overall demand based on the time of the year. The three quarters captured most of this fluctuation, as the third quarter ends in late September when demand has begun to diminish for winter. As this subset of the year's data seemed representative enough to capture seasonal variance in demand, the limited dataset could still be used without too heavily sacrificing the significance of results.

4.2 Individual Parameters

Intuitively, insight into some parameters and even the simulation itself could be gained by experimenting with changes to individual parameters. The parameters selected for these individual experiments were those expected to cause the greatest change to the system while still being easily and intuitively comparable. Therefore, comparisons between algorithms (such as demand station selection) are saved for the next section. The parameters chosen using these considerations were demand scale, rebalance rate, number of trucks, and truck capacity. For each, the difference between two possible values' main effects on the system is immediately clear (e.g. rebalancing is done more often for a rebalance rate of 2 than for a rate of 3). Other parameters were not included for different reasons. For example, the parameter for active hours' bounds has a significantly larger number of possible values, and their relationship is not always intuitive or predictable. More specifically, it is not clear what effect keeping total hours consistent but shifting the window some hours earlier or later is expected to have, making results from these experiments more difficult to interpret for rebalancing insight or system debugging.

A summary of the specific values used for these individual experiments can be seen in Table 1. For each parameter value, the list of metrics recorded was gross revenue, cost to rebalance, total lost revenue from missed demand, average minutes spent rebalancing per truck per day, average minutes spent idle per truck per day, and total bikes moved.

Table 1 Individual parameter experiment values

	Default	Values tested
Demand Scale	1.0	{0.5, 1, … 3}
Rebalance Rate	1	{1, 2, … 10}
Number of Trucks	1	{1, 2, 3}
Truck Capacity	20	{10, 20, 30, 40}

4.3 Parameter Combinations

Some of the simulation system's full potential comes from utilizing parameter combinations for experiments. Unlike those using only a single parameter, experiments with combinations of parameters require more decisions and more specific comparisons. This section gives insight into the process of choosing configurations for experiments as well as more detail on how the experiments were conducted.

4.3.1 Choosing Combinations

Mixing and matching different parameter values and algorithms enables the customization of a rebalancing strategy and allows for some fine-tuning to take advantage of resources most efficiently. For these experiments, the metrics recorded were gross revenue, cost to rebalance, total bikes moved, and total revenue lost from missed demand. These metrics seemed to represent the performance of a rebalancing strategy comprehensively enough for multifaceted analysis. Additionally, to determine whether some configurations were better suited to low overall demand or scaled particularly well, experiments were run under normal demand scales, doubled demand, and tripled demand (demand scale = {1, 2, 3}).

Eleven configurations were designed to be compared under these workloads using the aforementioned metrics. These configurations were designed based on practical experience with some of the algorithms' performance as well as combinations of parameters expected to be beneficial. The final configuration is a baseline configuration meant to demonstrate how meaningful an algorithm's metric value is. This baseline uses random demand and supply station selection. The demand station algorithm causes, on average, 10% of stations to be marked for filling, while the supply algorithm randomly selects as many non-empty stations as necessary to provide enough bikes. Each configuration was simulated five times at a given demand scale and the results of each metric were averaged to reduce the influence of expected simulation variance.

The specific parameter values used for each of the configurations are summarized in Table 2. For readability, rebalance bounds are specified with time instead of Tslices. "Deficit" and "Supply" refer to algorithms using the prediction network, and algorithms denoted "Class" are those defined on relative fill levels. An asterisk (*) next to a parameter value signifies a non-default value for that parameter. A dash (-) signifies that the parameter is irrelevant for that configuration.

4.3.2 Metrics and Environment

Gross revenue is the measure of how much money was generated by trips in the system throughout the entire period simulated. When a customer wishes to use the Healthy Ride bike-sharing system, they may either pay for an individual trip or pay for a subscription service allowing them unlimited rides in up to a max duration in a given period for a flat rate. Because the public data does not include any information about subscriber counts, revenue from this source cannot be known. Additionally, bike trips taken by subscribers would not generate any additional revenue. In the historical data, approximately half of the trips recorded were using one-time payments, and the Healthy Ride Pittsburgh's rate is $2.00 per 30 minutes. Therefore, the gross revenue was calculated by randomly sampling half of the final simulated tips without replacement. Based on duration, the prices of these sampled trips were summed to give the total gross revenue.

The cost to rebalance is calculated using three main components. The first is based on the total distance traveled by all trucks across all rebalancing sessions. This distance is used to calculate the cost of gas for the trucks (assuming the trucks get 11 miles-per-gallon and gas costs $2.75). The second component is an estimated maintenance cost for each employed driver (to cover expenses like insurance, etc. which are not included in a driver's wages). This maintenance cost is assumed to be 35% of the driver's total earnings through the simulated period, and the number of drivers is assumed to be equal to the number of trucks available. The final and most significant contributor to the rebalancing cost is wages for the driver(s). Healthy Ride drivers reportedly earn $9.00 an hour, so the total active rebalancing hours across the simulation is multiplied by 9 and by the number of trucks. The three components are summed to give the total cost.

The total number of bikes moved is simply calculated by counting every bike that is deposited in a station. Because session path plans do not necessarily reflect the actual number of bikes moved, the counting must be done when the bike is actually moved. To avoid double-counting bikes as they go in and out of a truck, bikes are only counted after they are removed from the truck and placed in a station.

The total revenue lost from missed demand is calculated using all of the recorded missed trip information. As mentioned in Section 3.2, when a trip cannot occur, part of the information saved is the sampled duration the trip would have had. Using this, the revenue that would have been gained from each trip can be calculated and summed to give the total.

Experiments were written in MATLAB and executed using version 2019a. The code was run on a personal laptop running a 64-bit Windows 10 OS with a single 2.4GHz processor and 8 GB of RAM. This information is relevant as time was a bottleneck when conducting experiments.

4.4 Summary

Overall, the experiments can be split into two categories: those using individual parameters and those using parameter combinations. Both utilize data from the Pittsburgh Healthy Ride system and are meant to compare some key metrics surrounding cost and rebalancing activity. Values were chosen to give a comprehensive range when possible, but were otherwise selected based on system experience or intuition. The computing environment was a bottleneck in experimentation by causing time limitations and restrictions.

Table 2 Exact values for each parameter in all experimental configurations. An asterisk (*) signifies a non-default value for that parameter. A dash (-) signifies that the parameter is irrelevant for that configuration.

Name	Rate	Num. Trucks	Capacity	Rebalance Bounds	Demand Method	Demand Ratio	Supply Method	Supply Ratio
Default Predictor	1	1	20	9:00am – 7:00pm	Deficit	-	Surplus	-
Default Threshold	1	1	20	9:00am – 7:00pm	Threshold*	0.5	Threshold*	1.0
Default Class	1	1	20	9:00am – 7:00pm	Class*	0.1*	Class*	0.1*
5 Hour Predictor	1	1	20	9:00am – 2:00pm*	Deficit	-	Surplus	-
3 Trucks / 3 Days Predictor	3*	3*	20	9:00am – 7:00pm	Deficit	-	Surplus	-
5 Trucks / 5 Days Predictor	5*	5*	20	9:00am – 7:00pm	Deficit	-	Surplus	-
[0.2 0.8] Threshold	1	1	20	9:00am – 7:00pm	Threshold*	0.2*	Threshold*	0.8*
2 Truck Threshold	1	2*	20	9:00am – 7:00pm	Threshold*	0.2*	Threshold*	0.8*
5% Class	1	1	20	9:00am – 7:00pm	Class*	0.05*	Class*	0.05*
2 Truck 5% Class	1	2*	20	9:00am – 7:00pm	Class*	0.05*	Class*	0.05*
Random Predictor	1	1	20	9:00am – 7:00pm	Random*	-	Random*	-

5.0 Experimental Results

Analysis of the experimental results is split into the independent parameter experiments and parameter combination experiments. The independent parameter experiments were less time- and resource-consuming than the combination experiments, mainly due to a smaller number of trials. Additionally, for most of the independent trials, the demand scale was kept at the default value of 1, which gave the system fewer possible trips to process overall.

5.1 Independent Parameters

Among all of the independent parameter experiments, the most interesting and insightful results came from the rebalancing rate experiments. Intuitively, one would expect the cost of rebalancing to go down as rebalancing days became increasingly spread out, but one would also expect the gross revenue to decrease in tandem. With less frequent rebalancing, predictions could be less accurate as the model is required to plan farther into the future. Under-filled stations could remain in that state for longer, missing significant demand and losing possible revenue. As seen in Figure 2, this is mostly the case. However, the plot does not show a perfectly linear descent in net revenue, and so further experimentation could show some rebalancing rates that can be decreased to reduce cost without dramatically impacting gross revenue.

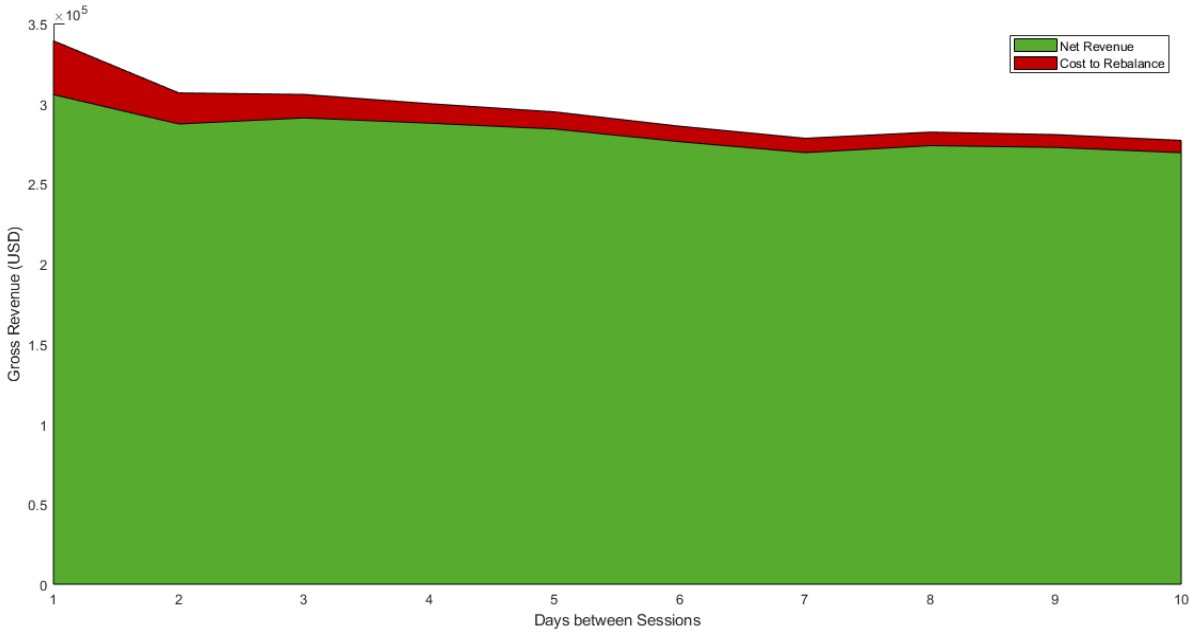

Figure 2 Net revenue and cost to rebalance as portions of gross revenue

The other parameters' experiments (demand scale, number of trucks, and truck capacity) either functioned exactly as expected or did not appear to independently affect the system with the default configuration. More specifically, increasing the demand scale while keeping rebalancing strategies and resources constant primarily increased missed demand and time spent rebalancing, as expected. The parameters affecting the truck's resources (number and capacity) did not independently give any benefit. Increasing the number of trucks would increase the cost to rebalance as well as each truck's average idle time, and increasing tuck capacity resulted in little to no change at all. This is likely because the single truck was able to perform rebalancing efficiently enough that increasing the capacity did not significantly reduce the number of session stops and adding trucks only spread the already achievable workload. These parameters would likely have more of an effect if the truck was a bottleneck for rebalancing (e.g. running out of time in the day or needing to revisit the same stations many times to fulfill requirements).

5.2 Parameter Combinations

Analysis of the combined parameter experiments is separated by individual metrics recorded. Subsection 5.2.1 discusses results for gross revenue, Subsection 5.2.2, results for the cost to rebalance, Subsection 5.2.3, results for total bikes moved, and finally, Subsection 5.2.4, results for missed demand. Additionally, some remarks about other analysis that was not performed are found in Subsection 5.2.5. These experiments were time-consuming to run, taking five days in total, but this runtime could be drastically reduced through parallelization (simulation runs are independent) and superior computing hardware.

5.2.1 Gross Revenue

The gross revenue results of the experiments are summarized in Figure 3. The X-axis is the scale of the demand workload, and for each value, the results are grouped. Overall, the gross revenue grows as the demand scale increases, which is expected as, even without rebalancing, an increase in demand for trips will increase the actual number of trips. Some trips do not require rebalancing to happen.

On a normal demand scale of 1.0, most approaches result in similar amounts of gross revenue. The exceptions are the default threshold, 3 trucks per 3 days, and 5 trucks 5 days, which failed to significantly outperform the random baseline. The reason for the latter two's poor performance may be due to compounding error in the demand predictor. Small errors in predictions may not derail frequent rebalancing, but in cases where rebalancing is less frequent, the model must be able to accurately plan for the future. Additionally, perhaps some long term fluctuations cannot be properly handled by rebalancing every 3 or 5 days.

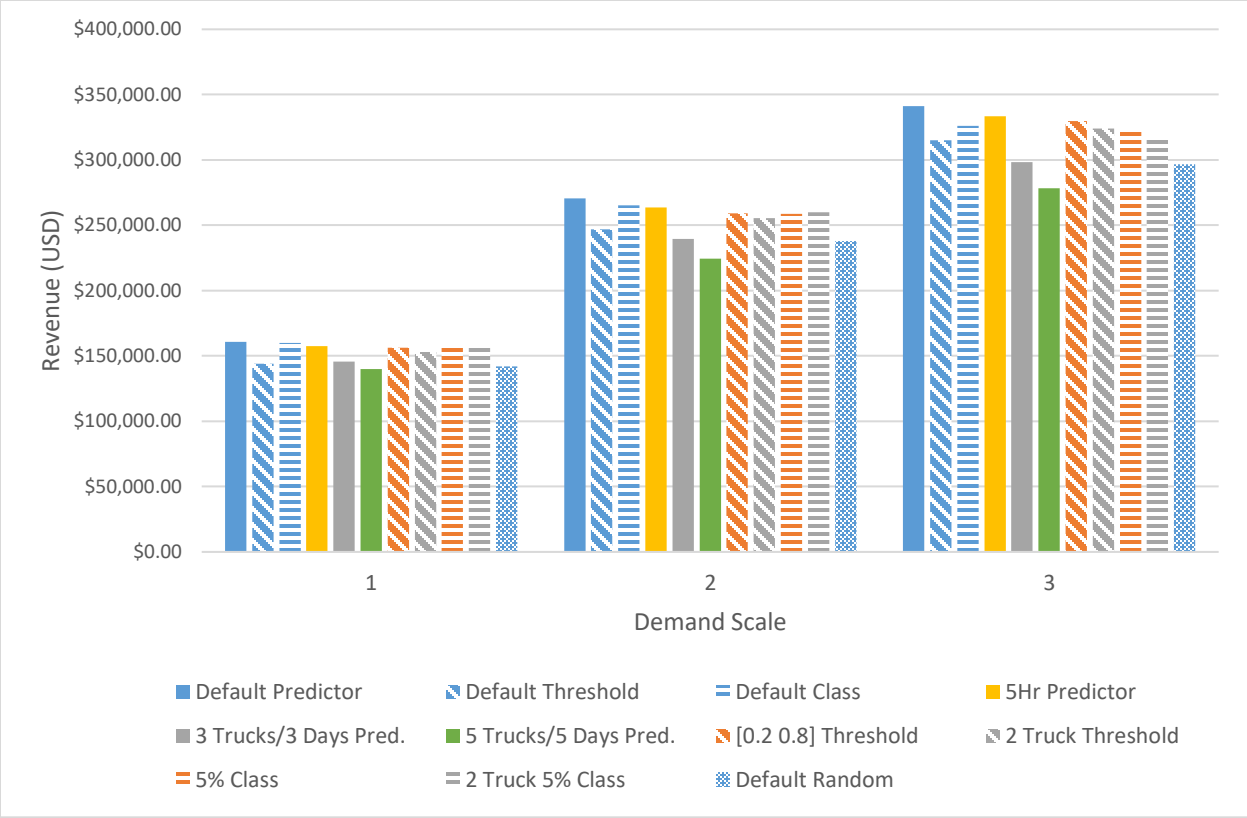

Figure 3 Gross revenue across rebalancing strategies and demand scales

5.2.2 Cost to Rebalance

Using the same format as the gross revenue graphs, the rebalancing cost results are displayed in Figure 4. As described in Subsection 4.3.2, the primary contributor to rebalancing cost is total driver wages, which rely only on the number of rebalancing agents and the total active hours. Therefore, between demand scales, there is not much variation in terms of rebalancing cost for a given strategy. Some approaches see a small increase (likely due to extra distance traveled and thus more gasoline), but proportionally the cost barely changes.

Comparing individual approaches, the costs can be roughly split into categories. The approaches using more than one truck, even when rebalancing less frequently, have more employees and therefore consistently incur a higher cost. With the exception of the five-hour per day rebalancing strategy, the rest of the rebalancing strategies have the same active hours and one driver, so the only variation in cost comes from the total rebalance distance. In this group, the default class method has the highest cost, demonstrating that it covered the most distance. This is not surprising, as the relative fill level algorithms always travel to the same number of stations per session. Other methods might have short sessions on which little distance is covered.

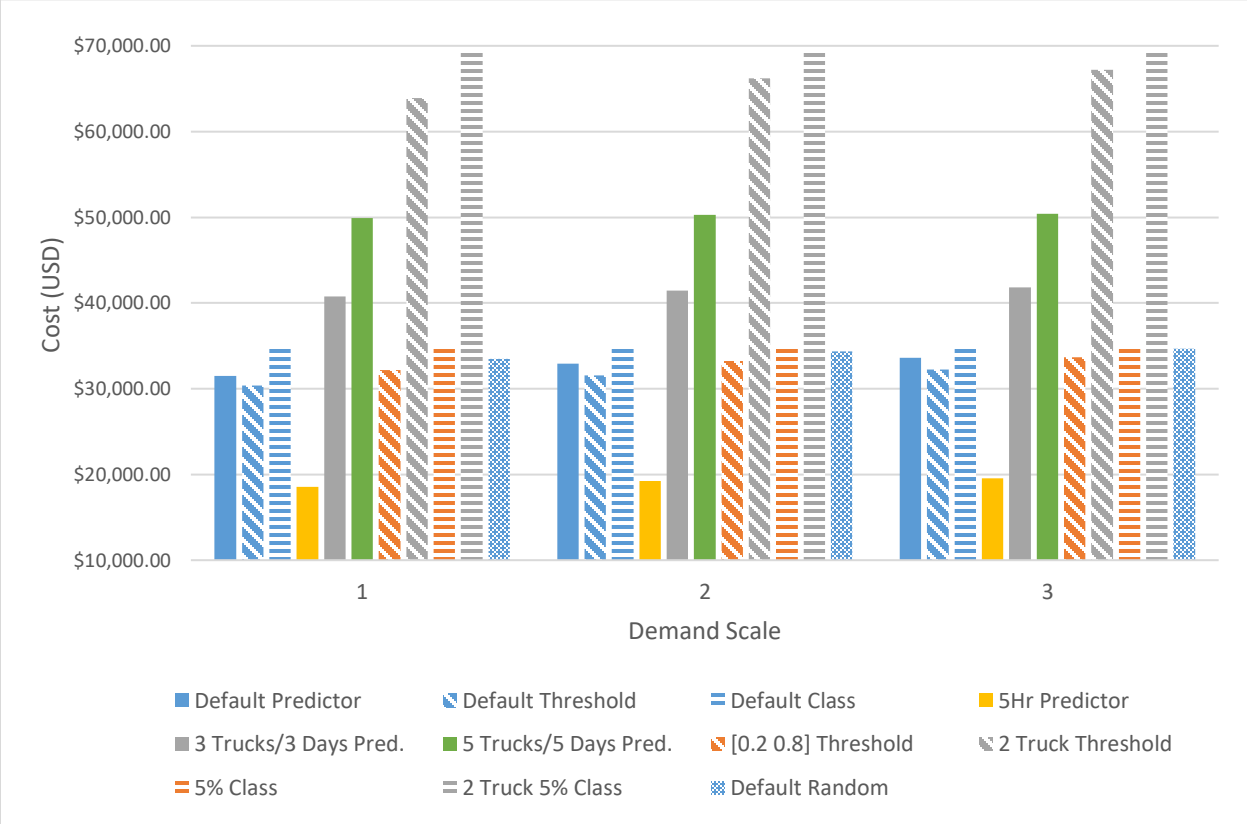

Figure 4 Cost to rebalance across strategies and scales

5.2.3 Total Number of Bikes Moved

Figure 5 shows the results for bikes moved during the experiments. This metric provides a lot of insight into how much work the rebalancing method is actually doing, but it can be very difficult to interpret as many different factors can affect the final bike total.

First looking at the approaches which utilize the demand predictor, the number of bikes moved always increases as demand scale is increased. This is desirable behavior because as the system's workload increases, the demand predictor should anticipate more demand and plan to move more bikes accordingly.

Moving on to the threshold-based approaches, the same trend is visible. A possible explanation for this trend is that with increased overall demand, the system is more active and station fills are more fluid. If station fills are constantly changing by relatively large amounts, stations may fluctuate above and below thresholds at a higher rate than a more stable system with less overall demand. If the fill levels are regularly being brought within the safe threshold through rebalancing and then pulled out of it again (either by large amounts of demand or a large influx of bikes), then the algorithms will select these stations more frequently and move more bikes.

One noteworthy aspect of the default threshold approach is its very low number of bikes moved relative to similar approaches. This is almost certainly due to inadequate bikes from supply stations. With an upper threshold of 1.0, only stations that are overfilled can provide bikes for rebalancing. Overflowing stations are not exceptionally rare, but they are not the norm. There may be an abundance of demand stations with not enough bikes to fulfill their requests. Another interesting part of these results is that at normal demand scales, the single truck [0.2 0.8] thresholding approach moved more bikes than the two-truck equivalent. Because each agent’s rebalancing session plan is designed independently of the other agent’s, it is possible that when

the demand workload is light, the agents' attempts to not idle cause them to interfere with each other's rebalancing. This could be done by performing some part of rebalancing at a station that another agent will visit later. For example, truck A and truck B want to go to the same station and pick up 5 and 10 bikes, respectively. Assume the station cannot fulfill both requests (e.g. the station has only 10 bikes). Whichever truck shows up first will proceed as normal, but the second truck, if no other bikes arrive, will be short five bikes. This shortage can continue for the duration of the session, so fewer bikes are deposited. With higher levels of demand, however, it is more likely that there is enough necessary work to keep the agents separate or for the station fill to change and remedy the problem between truck arrivals.

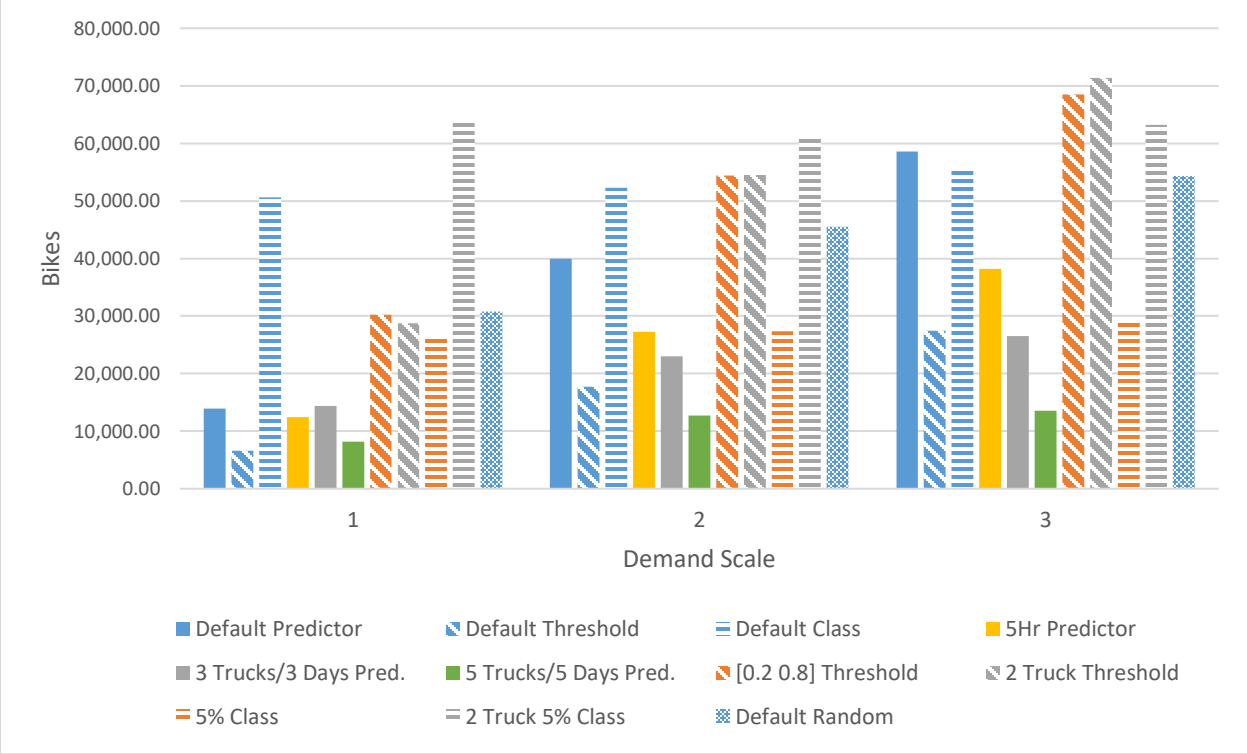

Figure 5 Bikes moved across rebalancing strategies and scales

The number of bikes moved in the relative-fill-level based "class" approaches does not change much with the demand scale increase. This is because the number of bikes in the whole system stays the same, so bikes in the most full and most empty stations should not have their fill levels significantly affected.

The random baseline seems to follow the trend of demand scale increasing bikes moved. There a few possible theories to explain this. One is that since candidate selection for random demand stations eliminates filled or overfilled stations, the increased demand may keep more bikes in transit, which would reduce the number of bikes in stations and reduce the number of full stations. Another theory is that having a more active system increases the need for rebalancing at every station, random selections will not include as many bad station choices where bikes cannot be deposited or picked up.

5.2.4 Total Lost Revenue from Missed Demand

The lost revenue from missed demand metric, shown in Figure 6, is closely tied to the gross revenue metric, and most results seem to show opposite trends to their gross revenue counterparts, but directly observing missed demand in combination with the other metrics (especially bikes moved) provides more well-rounded performance analysis. First and foremost, these results reveal a need for scalability in rebalancing resources. Relative to other approaches in the same demand scale, performance is mostly consistent. In terms of actual revenue lost, scaling demand results in more of an increase than the scale increase itself. Notice how when moving from normal demand to doubled, the default configuration missed four to five times more revenue. Most of these algorithms are using only one truck with default capacity to perform all rebalancing. They may run into issues of running out of active hours for rebalancing or spend too long on each rebalancing

session going back and forth to the same stations. Augmenting rebalancing resources could help. The problem may not be so easily avoided, however. A station misses trip demand when empty. If the rebalancing strategy has not changed, it may make mistakes in the same stations on scaled-up demand versus normal. These mistakes, however, may be much more costly. Being unable to satisfy up to three customers at a station is a small issue, but when the demand scale is 3.0, up to nine customers could miss their trips, inflating the lost revenue totals.

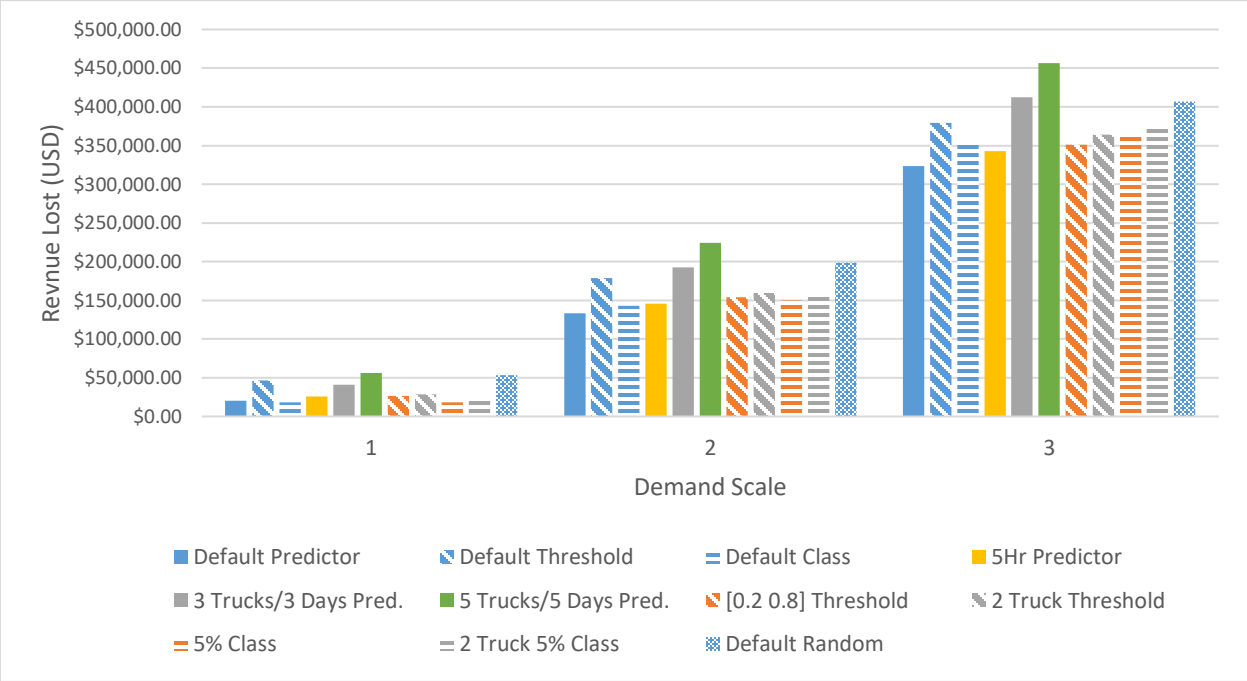

Figure 6 Missed demand across rebalancing strategies and scales

Looking at individual results would yield much the same insights as in Subsection 5.2.1. One small detail, however, is slightly more apparent in the missed demand graph. At normal demand scales, the class rebalancing methods achieve performance comparable to the successful default configuration. As overall demand increases, however, the demand predictor default more significantly outperforms the relative-fill-based algorithms. These types of algorithms likely do

not automatically scale in the way the demand predictor does. Working with the top and bottom 5-10% of stations places a flexible cap on the number of bikes that can be used and a strict upper bound on the number of stations involved. The demand predictor can look at every station and determine the rebalancing needs and capabilities of each, and include or exclude stations from the process as necessary. Increasing the percentage of stations at the top, bottom, or both may help the class approaches perform better under heavy workloads.

5.2.5 Cost-Benefit Analysis

When interpreting and comparing results on the system scale, some lower-level insights and tradeoffs in the rebalancing process are obscured. Including some sort of cost-benefit analysis to smaller samples of rebalancing work could help describe different rebalancing approaches' relationships between gross revenue and rebalancing cost. Assume two approaches with very similar overall costs, approaches A and B. In a particular session, approach A included demand station 1 and approach B did not. To fill the need at station 1, approach A will have to incur some cost that approach B will avoid. A, however, expects some benefit or reward from taking on this cost. Comparing the expected cost and benefit of small-scale rebalancing decisions could encourage a better understanding of them. In addition, though, this comparison could be used in other stages, too.

Adding a cost-benefit analysis to the rebalancing session construction process could provide more varied and efficient rebalancing strategies. This could take the form of weighing the expected revenue gain from demand at a station against the well-approximated cost of doing the rebalancing work needed to fill that demand. From this, portions of demand or whole stations could be left out of rebalancing to reduce costs. This enhancement, however, would likely affect most

rebalancing methodologies similarly. Therefore, comparisons without this component may represent the same comparisons with cost-benefit analysis well.

5.3 Summary

Summarizing the results reveals that while insights from individual parameter experiments were limited, parameter combinations have a lot of potential for comparing the efficacy and efficiency of various rebalancing strategies. Individually, modifying rebalancing rate showed some possible cost-revenue optimizations, but these results are difficult to analyze on their own. Through combinations, advantages and shortcomings in strategies emerge. Methodologies with demand prediction remain most effective, but some heuristic algorithms show promise. Through tuning of other parameters to use resources more effectively, they may show more potential. Finally, these experiments and results could gain from including cost-benefit analysis at the rebalancing session level, but overall trends would probably remain the same.

6.0 Related Work

This chapter will provide more in-depth descriptions of the related work on bike-sharing systems. Here, this work is separated into two broad categories: the more general analysis of these systems overall, and the work featuring a significant demand prediction component or contribution.

6.1 Bike-Sharing System Analyses

The analysis of bike-sharing systems is not a novel area of research, specifically in the areas of visualizing and predicting bike usage. Often, however, modeling usage, modeling demand, and attempts to create an efficient rebalancing strategy are somewhat separated. Work on temporal and spatial analysis of bike and station usage is an example of modeling demand and supply without addressing specific logistic rebalancing concerns in a bike-sharing system without fixed stations (Chandra, Liu, and Fu 2018). (Cazabet, Jensen, and Borgnat 2018) also analyzed bike usage over time using matrix factorization on different moving time windows to gain insight into the system as well as into the city in which it was found. Work in this area is useful for understanding how a system is used and providing insight that may, down the line, help to improve the system. In the short term, however, how to make practical use of this work is not as apparent.

Some inspiration for the system simulation came from work by (Au 2019), as that author built a similar simulator to implement and demonstrate a numerically-verified multi-agent distributed rebalancing algorithm. This algorithm is capable of giving an optimal rebalancing

strategy but requires significant computing power for large systems. More importantly, the simulation built to demonstrate these capabilities did not provide many of the metrics desired for comparative analysis on strategies. Furthermore, this simulation did not possess any parameters to modify system behavior beyond input data, therefore comparing solutions to the rebalancing problem was not feasible.

6.2 Work on Demand Prediction

Extensive work has been done and numerous papers have been published on demand prediction in station-based bike-sharing systems. Station-based systems are both an easier demand-prediction problem (compared to station-less) and more necessary to solve. Having fixed station locations increases customer expectation that bikes should be available at specific locations and the likelihood that many bikes will be concentrated in certain locations.

In the last few years, good performance has been achieved on this problem by utilizing GCNs to incorporate geographical relationships between stations into the prediction process (Chai, Wang, and Yang 2018; Saxena and Cao 2019). In addition to their state-of-the-art performance, these networks predict demand per hour, are well tested on large systems with a plethora of historical data available, and do not suffer a severe performance drop off when predicting more future time steps. Despite these favorable results, these models were not copied or recreated for the demand predictor. Instead, a vanilla LSTM approach was chosen, primarily for simplicity. Examining and evaluating rebalancing algorithms using the simpler network and a coarser time granularity made it easier to analyze the network's predictions and how the algorithm utilized them. Using more state-of-the-art networks may have necessitated more data or functioned as a black

box within the algorithm. It is possible, however, that the relative performance of non-predictive, heuristic algorithms could be inflated by intentionally using a worse predictive model.

One work especially worth mentioning is the statistical demand prediction approach and rebalancing analysis of (Hulot, Aloise, and Jena 2018). This work shares one of the major goals of this thesis: finding an efficient, effective, and realistic rebalancing strategy. This author took a more mathematical approach, using reduction methods to simplify the computation and non-neural machine learning models to predict demand. Using these predictions, the model uses similar mechanisms to thresholding for rebalancing, aiming to keep all stations in a certain fill range. Some areas not covered by the author that were explored in this thesis are, first, the use of models that are not primarily data-driven (e.g. the relative-fill-based algorithms). Additionally, the rebalancing strategies proposed by this author require an operator to monitor and manage station rebalancing, whereas this thesis aims to further automate the process.

6.3 Summary

Research on bike-sharing systems (station-based or otherwise) does exist, but still possesses some gaps in detailing how to compare and apply solutions. There are various papers on the movement of bikes through the systems and predicting this flow. Some recent papers, especially those utilizing GCNs, achieve good performance, but the papers do not expound on how this performance may be applied in the real world. Additionally, it is not clear how these networks compare to other methods (using a GCN or other approaches) on different bike systems or the same bike system under a different demand workload. The simulation detailed in this thesis could fill these gaps.

7.0 Conclusions and Future Work

The last chapter of this thesis is divided into three sections. First, the contributions will be reiterated and summarized. Next, possible extensions and continuations of the work will be presented. Finally, the thesis closes with some comments on the significance of this problem overall and, more specifically, this thesis's contributions towards solutions.

7.1 Summary of Contributions

Reviewing the goals of this thesis confirms- that the simulator is making progress on the problem of rebalancing in bike-sharing systems. To reiterate, the current simulator is capable of performing side-by-side comparisons between rebalancing methods and across different configurations. While no experimental results are provided here, the system is capable of accepting data from different cities' systems and has been shown to operate using several different parameter configurations.

Additionally, despite experimental results demonstrating that heuristic-based algorithms have room to improve, these approaches were simulated using feasible real-world resource allocations. These experiments demonstrated not only the potential of alternate algorithms, but they also provided configurations for rebalancing resources that can be directly implemented into the real world. Each proposed rebalancing solution contains such configurations.

Finally, the experimental results detailed in this thesis show the efficacy and promise of the simulator as a comparative tool, capable of acting as an objective evaluator of multiple

methodologies or as a benchmark to demonstrate one approach's efficiency. Furthermore, some of the system's features are such that the experiments in this thesis need not be the only results generated.

7.2 Future Work

One of the contributions of this thesis is a simulation system designed to test and compare rebalancing strategies. A key feature is its modularity, so an obvious area for future work involves using the system to conduct more experiments. There are too many parameter combinations to meaningfully test them all, but some significant avenues have not been explored in this thesis. First, conducting more focused experiments around parameters that affect rebalancing resources (number of trucks, active hours, and truck capacity) could give insight into certain approaches' potentials. Methodologies limited by resources above certain demand levels may be able to achieve meaningful reductions in missed demand.

Additionally, experimenting outside of the paired demand and supply methods could provide more information about individual algorithms and might also improve performance. For example, the supply algorithm using the prediction network may be a limiting factor in demand predictor configurations. Hypothetically, changing the supply method to, for instance, thresholding could outperform configurations using either set of paired algorithms. Furthermore, no formal experiments were conducted using the path planning algorithm that does not allow station revisits. Perhaps avoiding this backtracking could reduce costs without significantly adding to missed demand.

Another avenue for experimentation is the use of different cities' bike systems and different data. All of the experiments conducted were simulations of Pittsburgh's Healthy Ride system and relied on the same data. In other cities, different methodologies could possess advantages or interact differently due to different geographical station relationships, demand concentrations, etc.

As discussed previously, including some cost-benefit analysis while building session paths would add another dimension to session planning. Weighing the expected gain of a rebalancing move versus the cost of carrying it out, the system could rely on a decision threshold to determine if the move is worth the cost. This change would not be trivial, however, as the process leans heavily on the accuracy of not only demand prediction, but trip duration prediction as well. If the system is not able to reliably predict demand and duration, the expected gain would be inaccurate. Comparing it to the cost of that rebalancing move becomes less meaningful. Because the demand predictor being used is not state of the art and predicting trip durations accurately adds another dimension to the problem, incorporating this is would be a significant undertaking and must be left to future work.

7.3 Concluding Remarks

Looking at all of the experimental metrics together, the demand predictor consistently achieves the best results at a reasonable cost. Other methodologies, such as the relative-fill-based algorithms, provide some competition but would almost certainly fall short of predictive algorithms with a state-of-the-art predictive model backing them. That is not to say that the problem is solved, as there is still a lot of experimentation possible that may reveal parameter combinations to sometimes or consistently outperform the predictor. Also, it is important to note

that the predictor requires existing historical data for training before it can be effective. Heuristic approaches have no such barrier. Additionally, these results cannot be applied to other bike systems as they are; different cities and bike systems may yield different performance.

These open-ended possibilities, however, are perfectly in line with some of the goals of this research. Because the simulation system is flexible, it fits with other research in this space and can be used to demonstrate the efficacy and efficiency of other works. Bike-sharing systems continue to evolve and spread around the country and the world, and so the need for innovation in this space continues. Tools like this simulator and consistent experimental comparisons between methodologies will provide continuous insight into the successes and shortcomings of rebalancing strategies, enabling system administrators to properly manage this transportation infrastructure and improving the experience of those who wish to use it.

Bibliography

Au, Eden. 2019. "A Generic Model and a Distributed Algorithm for Optimization of Station Based Bike Sharing Systems." Commit 2648342. Accessed at https://github.com/edenau/Bike-Sharing-Systems-Optimization.

Cazabet, Remy, Pablo Jensen, and Pierre Borgnat. 2018 "Tracking the Evolution of Temporal Patterns of Usage in Bicycle-Sharing Systems Using Nonnegative Matrix Factorization on Multiple Sliding Windows." International Journal of Urban Sciences 22, no. 2. (2018): 1-15. https://doi.org/10.1080/12265934.2017.1336468.

Chai, Di, Leye Wang, and Yang Qiang. 2018. "Bike Flow Prediction with Multi-Graph Convolutional Networks." Paper presented at SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, November 2018. https://doi.org/10.1145/3274895.3274896.

Chandra, Dakshak Keerthi, Kunpeng Liu, and Yanjie Fu. 2018. "How unbalanced are bicycle dynamics? Demand-supply shortage detection with spatiotemporal tensor factorization in station-less bike systems." Paper presented at SDM '18: 18th SIAM International Conference on Data Mining, Sand Diego, May 2018. https://illidanlab.github.io/big_traffic/2018/papers/chandra2018how.pdf.

Cormen, Thomas H., Charles E. Leiserson, Ronald Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. 3rd Edition. Cambridge: The MIT Press.

Healthy Ride Pittsburgh. *Historical Trip Data*. Q1-Q3. (2019). distributed by Healthy Ride Pittsburgh. https://healthyridepgh.com/data/.

Hulot, Pierre, Daniel Aloise, and Sanjay Jena. 2018. "Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems." Paper presented at KDD '18: 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, July 2018. https://doi.org/10.1145/3219819.3219873.

Lin, Lei, Zhenbing He, and Srinivas Peeta. 2018. "Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach." Transportation Research Part C: Emerging Technologies 97. (December 2018): 258-276. https://doi.org/10.1016/j.trc.2018.10.011.

MATLAB and Deep Learning Toolbox Release 2019a, The MathWorks, Inc., Natick, Massachusetts, United States.

Saxena, Divya, and Jiannong Cao. 2019. "D-GAN: Deep Generative Adversarial Nets for Spatio-Temporal Prediction." ArXiv: https://arxiv.org/abs/1907.08556.