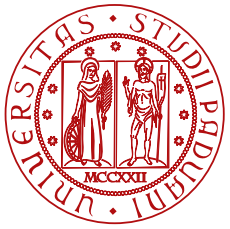

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

MASTER THESIS IN CONTROL SYSTEM ENGINEERING

Bike-sharing system rebalancing by greedy algorithm

MASTER CANDIDATE

Hao Peng

Student ID 2018603

SUPERVISOR

Prof. Federico Chiariotti

University of Padova

CO-SUPERVISOR

Alberto Testolin

University of Padova

ACADEMIC YEAR
2020/2021

To my parents
and friends

Abstract

Bike-sharing systems are becoming increasingly popular in cities around the world, providing a sustainable and convenient mode of transportation. However, one of the biggest challenges in operating a bike-sharing system is rebalancing, which refers to the redistribution of bikes from high-density to low-density areas. In this study, we propose a greedy algorithm to rebalance bike-sharing systems. The algorithm considers the current demand for bikes at each station and dynamically adjusts the number of bikes that are being transported to ensure that all stations have a sufficient number of bikes. The results of our simulation show that our greedy algorithm has a relatively worse than the one in the ideal scenario, w. These results have important implications for the design and operation of bike sharing systems in cities around the world.

Sommario

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

CSV Comma Separated Values

1 Introduction

The bicycle sharing system is a concept that has gained widespread popularity in recent years as a convenient and sustainable mode of transportation in urban areas. With Paris' successful smart bike-sharing system, these systems are bringing mobility into everyday European cities with rapid adoption in European cities [22]. The basic idea is to provide a network of bike stations throughout the city, where people can rent bikes for short trips and return them to any station. This system has proven to be a success in many cities, helping to reduce traffic congestion and air pollution while providing a convenient and affordable alternative to traditional modes of transportation [25, 28].

However, one of the biggest challenges in operating a bike-sharing system is rebalancing [30], which refers to the redistribution of bikes from high-density areas (where a large number of bikes are located) to low-density areas (where a small number of bikes are located). Rebalancing is an effective technique to increase efficiency and reduce the number of bikes to meet the demand for bicycle use. It is about moving the bike from crowded stations to stations where users need bikes. For example, an operator can send trucks to collect bicycles from crowded stations and determine routes to unload bikes and leave at stations where bicycles are needed. To reduce the cost, it is reasonable to decide on the shortest route to cover all stations. Although there are different approaches under different circumstances, there is a way to solve the static problem [4] and also to solve the dynamic problem [7]. Static rebalancing refers to the practice of regularly moving bicycles between docking stations to ensure that each station has a sufficient supply of bicycles for users. Dynamic rebalancing, on the other hand,

is a more real-time approach that involves using data and technology to automatically adjust the distribution of bicycles based on current demand. This is necessary to ensure that bikes are available when and where people need them. The rebalancing process can be time-consuming and costly, especially in large cities with complex bike-sharing networks [13].

In response to this challenge, researchers and engineers have developed various methods to optimize the rebalancing process [10, 6]. One such method is the use of greedy algorithms, which can automatically and dynamically adjust the bike distribution to meet changing demands. The use of algorithms can significantly improve the efficiency of the rebalancing process, reducing the number of bikes that must be transported and increasing the number of users served [31].

This research topic is of great importance for the design and operation of bike-sharing systems, as well as for the overall sustainability of urban transportation systems. By improving the efficiency of the rebalancing process, bike sharing systems can become an even more attractive alternative to traditional modes of transportation, helping to reduce traffic congestion and air pollution while promoting sustainable transportation.

Bike-sharing systems can be categorized into two main types: station-based and free-floating. The primary difference between these two types of systems is the way in which users can access and return bicycles. In a station-based system, bicycles are stored in fixed locations, known as docking stations. Users must take and return bicycles to these stations, and payment is typically based on the duration of use. Docking stations can be found throughout the service area and users can use a mobile app or a membership card to check out and return bicycles at any station. Station-based systems are typically larger and more complex than free-floating systems and require more infrastructure, such as dedicated parking spaces and bike racks. In a free-floating system, bicycles are not anchored at a specific location and can be parked anywhere within a designated service area. Users can use a mobile app to locate and reserve bicycles, and payment is based on the duration of use. Free-floating systems offer more flexibility and convenience for users as they do not need to worry about finding a docking station to pick up or return a bike. However, this system requires a more advanced technology for bike tracking and monitoring, and bike parking and vandalism become more difficult issues. Citi Bike is the largest bimodal system in New York City. which was launched in May 2013 and has become a key element in the transportation network, was launched in May 2013 and has

become part of the network. It is possible to unlock a bike at any station and return it to any other station in the system, making it ideal for one-way trips. People ride a bike to work or school, run errands, get to appointments or social activities, and many more. Citi Bike is open to users 24 hours a day, 7 days a week, 365 days a year. The bike is available for use for 365 days, and riders can access thousands of bikes at hundreds of stations in Manhattan, Brooklyn, Queens, and Jersey City.

We face two major challenges in handling the rebalancing issue. One is how to predict which docks need bikes and which do not. Another is to decide the shortest route to minimize the cost of this service. The proposal to answer the first challenge is to model the occupancy of each station as a Birth-Death Process (BDP) with time-varying birth and death rates (i.e., arrival and departure). In such a manner, we can estimate how long the station is self-sufficient before running out of bikes or available docks, as long as its original state is known. Intuitively, the rebalancing cannot be performed every moment with respect to the monetary cost and the traffic. Therefore, a threshold should be set to trigger rebalancing. The solution to the second challenge can be done with the greedy algorithm. The greedy algorithm works by selecting the best option available at the moment, without considering the impact it may have on future options. This approach can lead to the finding of an optimal solution in a relatively efficient manner and is often used in optimization problems such as scheduling [26], minimum spanning trees [23], and Huffman coding [5]. In this study, the optimization problem is quite close to the scheduling problem. When considering that the problem is NP hard, only a suboptimal solution can be found. In this study, we are considering a scenario in reality where the truck has a capacity when performing the rebalancing operation. In the ideal situation, there is no such constraint on the rebalancing vehicle.

To assess the performance of our schemes, we use three different metrics: the percentage of system outage (empty or full station), the number of daily rebalancing operations, and the daily distance covered by the rebalancing trucks. The final results are compared under different scenarios; one is static rebalancing twice a day for a fixed period of time. Another is rebalancing using the greedy algorithm. All analyzes are based on data from Citi Bikes in New York City. In particular, data collected from July 2013 to June 2016 is used as training data sets, while data from July 2016 to June 2017 is used as a test data set. Observation should be taken into account that unusual events happen to affect our

final analysis results. For example, a sudden increase or decrease in the number of bike rentals as a result of a change in the weather, the number of bike malfunctions. Although there are some anomalies in the patterns, the proposed framework should handle them properly under a bearable loss. An analysis of the bike sharing system would give us a better understanding of urban mobility. Provides valuable information on urban mobility patterns and behaviors, which can inform planning and decision making for transportation at the city level. It also increases user satisfaction by understanding the usage patterns and behaviors of bike sharing users; operators can make improvements to the system that improve the user experience, such as improving the availability of bikes, reducing wait times, and improving the overall quality of bikes.

The work is schemed as follows, 2 gives us insight about the latest work related bike-sharing system, optimization problems that have been conducted. 3 is about the modeling of the bike-sharing system and the methodologies to analyze the overall problem. 4 includes the parameters and data processing. 5 concludes all the results we get and what further work may be needed to solve them.

2 Literature Review

2.1 DOCKING SYSTEM

Bike sharing systems have evolved significantly in recent years, incorporating various technological advances to improve their functionality and usability. A notable development is the emergence of dockless systems, which allow riders to locate and rent bicycles from various locations within a designated service area without the need for dedicated docking stations. From [21], we can get the following pros and cons of dock-based and dockless systems.

Dock-based systems:

Dock-based bike-sharing systems require riders to pick up and return bikes to designated docking stations. This model is often associated with more established bike sharing programs, where bikes are rented for a specified time period, and fees are charged based on the duration of the rental. Some advantages of dock-based systems include:

• Organized system: Dock-based systems provide designated bike stations where bikes can be easily located and returned. This makes the system organized and easy to manage. Predictable availability: Riders can be sure to find a bike at a docking station as long as they are within the service area and the bike is not already rented out. Reduced bike clutter: Since bikes must be returned to docking stations, bike clutter is minimized, and public space is left clean.

However, dock-based systems also have some limitations, including a low resolvability.

• Limited flexibility: The requirement of returning bikes to designated dock-

ing stations can make the system less flexible than dockless systems. Limited coverage: Dock-based systems require significant infrastructure investment, which can limit the coverage area of the system. Higher capital costs: Dock-based systems require significant capital investment, including the cost of bike parking and maintenance.

Dockless Systems:

On the other hand, dockless bike sharing systems allow riders to pick up and drop off bikes at any location within a specified service area. Dockless systems are often associated with more recent bike sharing programs, where riders use mobile apps to locate and rent bikes on demand. Some advantages of dockless systems include:

• Greater flexibility: Dockless systems offer more flexibility than dock-based systems, allowing riders to pick up and drop off bikes at any location within the service area. Lower Infrastructure Costs: Dockless systems require fewer docking stations, which can significantly reduce infrastructure costs. Increased coverage: Dockless systems can cover a larger area than dock-based systems, since there is no need to invest in expensive docking station infrastructure.

However, dockless systems also have some limitations, including:

• Bike clutter: Dockless bikes can be parked anywhere, leading to cluttered sidewalks and public spaces and complaints from some communities. Lower predictability: Because bikes can be parked anywhere, riders may not be able to locate a bike when they need one, or may find one too far away from their starting point. Vandalism and theft: Without the security of a fixed docking station, dockless bikes are more susceptible to theft and vandalism.

[21] It suggests four key approaches to reduce carbon footprints with respect to bike sharing systems. (1) using more sustainable approaches to optimize bicycle distribution and rebalancing, (2) encouraging more private car users to switch to using bike-sharing systems by incentives, (3) extending the life of docking infrastructure could significantly reduce the entire normalized environmental impact of station-based systems, and (4) increasing the efficiency of bike utilization to improve the environmental performance of dockless systems.

2.2 BIKE REBALANCING PROBLEM

Bike sharing system rebalancing is the process of redistributing bikes between stations to ensure that there are enough bikes and docks available to users

at different times and locations. Rebalancing is a challenging and important task for bike sharing operators, as it affects the quality of service, user satisfaction, operational cost, and environmental impact of the system.

There are two main types of rebalancing strategies: station reallocation and bike relocation. Station reallocation refers to changing the number and location of stations in the system, usually based on long-term demand patterns and spatial analysis. Bike relocation refers to the move of bikes from one station to another, usually based on short-term demand fluctuations and optimization models.

Station reallocation can be performed periodically or dynamically. Periodic reallocation involves planning the station layout in advance for a certain period, such as a season or a year, based on historical data and forecasts. Dynamic reallocation involves adjusting the station layout in real time or near real time, based on current or predicted demand and supply. Dynamic reallocation can be more responsive to changing conditions but can also be more complex and costly to implement.

Relocation of bikes can be done manually or automatically. Manual relocation involves using vehicles or staff to transport bikes between stations, usually following a predefined schedule or route. Automatic relocation involves using self-driving vehicles or smart bikes that can move autonomously between stations, usually following an online optimization algorithm. Automatic relocation can be more efficient and flexible, but also more technologically demanding and risky.

The literature on rebalancing bicycle sharing systems is vast and diverse, covering different aspects such as problem formulation, solution methods, performance evaluation, and case studies. Some of the main research topics include:

•Demand analysis and prediction: This involves studying the spatiotemporal patterns of bike share demand and supply and developing models or methods to forecast future demand and supply at different stations or regions.

•Optimization models and algorithms: This involves formulating the rebalancing problem as a mathematical program or a combinatorial optimization problem and developing exact or heuristic algorithms to solve it efficiently and effectively.

•Simulation models and tools: This involves developing simulation models or tools to mimic the behavior and dynamics of bike sharing systems and using them to test and compare different rebalancing strategies or scenarios.

•Performance metrics and indicators: This involves defining and measuring different performance metrics or indicators to evaluate the effectiveness, efficiency, robustness, and sustainability of rebalancing strategies or systems.

•Case studies and applications: This involves applying rebalancing strategies or systems to real-world bike sharing systems and analyzing their impacts on service quality, user behavior, operational cost, and environmental footprint.

Vallez et al. [27] provide a systematic review of the challenges and opportunities in rebalancing dock-based bike sharing, focusing on the algorithmic aspects of the problem. They performed a keyword analysis in the literature and a timeline that shows the evolution of those keywords throughout the last decade. They also include an exhaustive table that summarizes the main characteristics of 114 papers on bike sharing rebalancing published between 2010 and 2020. Beigi et al. [2] provide a comprehensive analysis of the usage pattern of Capital Bikeshare in the Washington DC metropolitan area, one of the prominent bike sharing systems in the United States. They also propose an optimization strategy formulated as deterministic integer programming to reallocating bike stations daily and rebalancing the bike supply system. They tested their strategy in a case study in Washington, DC, using historical data from 2019. Wang et al. [29] investigate an extended bike sharing rebalancing problem (BRP) that considers the influence of the number of bikes distributed by the operator on the demand of users. They proposed a mixed-integer nonlinear programming model for the problem and linearized it into a mixed-integer linear programming model. The model aims to maximize the profit of bike-sharing operators by making transportation vehicle route plans and determining the target number of bikes at each station after the redistribution operation. The model is tested on real Beijing Mobike data and shows that the rebalancing behavior significantly impacts user demand, thus guiding the operator.

The following content will focus mainly on the topic of bike rebalancing. The bike relocation problem is a type of optimization problem in which the goal is to minimize the cost of moving bikes from surplus to shortage stations while also ensuring that the bikes are available when needed. This problem is particularly challenging because it involves real-time decision making in response to changing user demand patterns and requires a detailed understanding of the underlying network of stations and how they are connected. The most widely used approaches are *static* and *dynamic*.

Static Approach: In the static approach, the bike rebalancing problem is

viewed as a single optimization problem that can be solved using mathematical modeling and optimization techniques. The goal is to find a single optimal solution that minimizes the cost of moving bikes from surplus to shortage stations, based on a snapshot of the system at a particular point in time. This approach assumes that demand patterns and bike availability remain constant over time and does not take into account the real-time dynamics of the system. Static rebalancing is usually performed by mixed integer programming (MIP). Branch-and-cut algorithms are proposed to solve a relaxation problem. The upper bound of the optimal solution for a problem is obtained by a tabu search based on some theoretical properties of the solution [4]. [12] It proposed a mathematical programming-based three-step heuristic for static repositioning problems. In the first step, the stations are organized based on considerations of geographic location and bike inventory. The second step is to route the repositioning of vehicles through clusters while tentative inventory decisions are made for each station individually. Third, the original repositioning problem is solved with the restriction that the traversal of repositioning vehicles is allowed only between stations belonging to consecutive clusters following the routes determined in the previous step or between stations belonging to the same cluster. Pal et al. [24] propose a novel bike sharing model called free-floating bike sharing (FFBS), where bikes can be parked and locked at any location within a predetermined area, thus eliminating the need for docking stations. They address the static rebalancing problem for FFBS, which consists of adjusting the number of bikes at each location to match the expected demand by using a fleet of vehicles that can transport and relocate the bikes. They formulate the static complete rebalancing problem as a mixed-integer linear program (MILP), which can accommodate single and multiple vehicles and multiple visits to a location by the same vehicle. The paper develops a hybrid nested large neighborhood search with variable neighborhood descent (LNS-VND) algorithm, which is capable of solving large-scale instances of the problem efficiently and effectively. They evaluated the algorithm on some benchmark instances from the literature and on some new instances based on real-life FFBS programs in Tampa and Chicago. Lahoorpoor et al. [18] propose a bottom-up spatial cluster-based model to solve the static rebalancing problem in bike sharing systems. The static rebalancing problem aims to adjust the number of bikes at each station to a predetermined level by using vehicles that can transport and relocate the bikes. They first investigate the spatial and temporal patterns of bike-sharing trips in the network.

They then define a similarity measure based on trips between stations and use a hierarchical agglomerative clustering method to discover groups of correlated stations. With assumption that there are two levels of rebalancing: intra-cluster and inter-cluster. The intra-cluster level balances the bike distribution inside each cluster, and the inter-cluster level balances the bike distribution between different clusters. And optimizing the rebalancing tours according to the positive or negative balance at both levels using a single-objective genetic algorithm. The rebalancing problem is modeled as an optimization problem that minimizes the length of the tour.

Dynamic Approach: In the dynamic approach, the bike rebalancing problem is viewed as a continuous optimization problem that requires real-time decision making and adaptation to changing user demand patterns. The goal is to dynamically allocate bikes between stations based on current and predicted future demand patterns, with the aim of minimizing the overall cost of cycling rebalancing over time. This approach requires the use of machine learning and optimization techniques that can be adapted to change user behavior and traffic patterns. [8] It presents a modeling approach by taking advance of the Dantzig-Wolfe and Benders decompositions to derive the lower and upper bound for the solution of the pick-and-delivery problem. Chiariotti et al. [7] propose a dynamic rebalancing strategy for bike sharing systems that uses historical data to predict network conditions and decide when and how to redistribute bikes between stations. The paper models the occupancy of the stations as birth-and-death processes and uses graph theory to select the rebalancing path and the stations involved. They validate the proposed strategy based on data provided by the New York City bike sharing system and show that it outperforms static rebalancing schemes based on a fixed schedule. Hu et al. [14] propose a dynamic optimization rebalancing model for docked bike-sharing systems that aims to minimize the operating cost of rebalancing while maximizing user satisfaction. They evaluate the demand for rebalancing using historical and predicted data to avoid unnecessary service for each station within a rebalancing horizon. They use a time window satisfaction modeling to evaluate user satisfaction and adopt a multiobjective evolutionary algorithm based on decomposition (MOEA/D) under the rolling horizon strategy to solve the model. They improve algorithmic performance by applying a local search based on station priority and perform numerical experiments using real-world data to demonstrate the proposed model and the advantage of the improved algorithm.

However, from a practical point of view, only the static approach to operating rebalancing is far from sufficient. A more practical method would be adopting the dynamic method. Obviously, the dynamic approach is difficult to implement when considering user activities and traffic during the day.

2.3 MARKOV DECISION PROCESS(MDP)

A Markov Decision Process (MDP) is a mathematical framework used to model decision-making problems in which an agent interacts with an environment over a sequence of discrete time steps.

In an MDP, the agent takes actions that affect the state of the environment and receives rewards based on the state transitions and actions taken. The state of the environment at each time step depends only on the previous state and action taken by the agent, and not on any earlier history.

Formally, an MDP is defined by a set of states S, a set of actions A, a transition function $T(s,a,s')$ that specifies the probability of transition from state s to state s' under action a, and a reward function $R(s,a,s')$ that specifies the reward received by the agent for transition from state s to state s' under action a.

The agent's goal is to learn a policy $\pi(s)$ that maps each state to an action, to maximize the cumulative reward expected over time. This is typically done using reinforcement learning algorithms that iteratively update the policy based on the observed rewards and state transitions.

[20] It proposed MDP to help decide which station to prioritize and how many bikes should be taken or added to each station. Its goal is to minimize the arrival rate of unsatisfactory users who cannot deposit bikes or rent bikes. In [3], their method also adopts MDP to prioritize stations according to the urgency that must be operated.

2.4 GREEDY ALGORITHM

Greedy algorithm, this algorithm is based on the principle of moving bikes from stations with a surplus to stations with a deficit. Select the station with the largest surplus and the station with the largest deficit and move bikes from the former to the latter until the surplus is eliminated or the deficit is filled. The process is repeated until no further improvements can be made.

Returning to the context of the relancing problem, it is similar to the Traveling Salesman Problem(TSP). To solve the TSP, the Nearest-Neighbor (NN) algorithm is a heuristic approach used to find a suboptimal solution to the problem. The TSP is a classic combinatorial optimization problem that asks for the shortest possible route that visits a given set of cities and returns to the starting city.

The NN algorithm starts by selecting an arbitrary city as the starting point and then repeatedly selects the nearest unvisited city as the next destination until all cities have been visited. Finally, the algorithm returns to the starting city to complete the tour. The resulting path is a suboptimal solution to the TSP.

One of the first greedy algorithms to solve the TSP was proposed by Duan et al. [9], who adapted a Hamiltonian path algorithm to find a route that covers all unbalanced stations (that is, stations that have more or less bikes than their target level). The algorithm starts from an arbitrary unbalanced station and iteratively adds the nearest unbalanced station to the route until all unbalanced stations are visited. Then, the algorithm greedily adjusts the route by inserting or deleting stations if the vehicle capacity is violated. The algorithm can be easily extended to a parallel version that assigns multiple vehicles to different subsets of unbalanced stations. The main advantage of this algorithm is that it has a flexible trade-off between running time and solution quality, as it can adjust the number of vehicles and the size of subsets. The main limitation is that a feasible solution may not be found if there are too many unbalanced stations or if the vehicle capacity is too small.

Another example of a greedy algorithm for the TSP is the one proposed by Kadri et al. [16], which is based on an insertion heuristic. The algorithm starts with an empty route for each vehicle and then iteratively inserts an unbalanced station into an existing route that minimizes the increase in distance and does not violate the capacity constraint. The algorithm stops when all unbalanced stations are inserted or when no feasible insertion is possible. The authors compared their algorithm with several other heuristics and exact methods in real-world datasets and showed that their algorithm can find near-optimal solutions in reasonable time.

3

System Model

The basic modeling works thanks to the work of Chiariotti et al. [7]. While to make the model more realistic, the capacity of the rebalancing vehicle has been taken into consideration. The network of bicycle sharing system stations can be defined as a fully connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. where $\mathcal{V} = \{1, \cdots, V\}$ is the group of stations and $\mathcal{E} \subseteq V \times V$ denotes the edges, which are the distances between each two stations, on the graph $\mathcal{G}$. For each station $v \in V$ has a capacity of bikes $m_v(t) \in \mathcal{M}_v(t) = \{0, 1, \cdots, M_v\}$ at time t. M_v stands for the maximum capacity of the station. The distance between each node can be denoted as $d(v_i, v_j)$ The goal of this work is to determine the route to rebalancing with an operating vehicle, with capacity C, minimize the cost and maximize user satisfaction. Here, we define the cost related to the number of vehicles, X, and the total distance covered. We set $N = 1$ in this study. For the satisfaction aspect of users, we try to meet the demand of users when they need bike service. To meet this requirement, we can increase the number of bikes at each station at the time t. To process the data, the data was sampled every time length T_r. In time frames T_r, for each time frame k, we decide the ideal state $m_v^*(kT_r)$ for $v \in V$. $m_v^*(kT_r)$ is the value that is least likely to be 0 or M_v over a period of time. Maximizing *survival time* is the first step in deciding on our rebalancing operation. Analyzing historical data on the bike sharing system could achieve this goal. Second, though we have the optimal survival time of each station, there is a trade-off we are facing, which is minimizing the total distance and not visiting a station twice simultaneously. The *network-wide optimization* was solved by introducing the greedy algorithm.

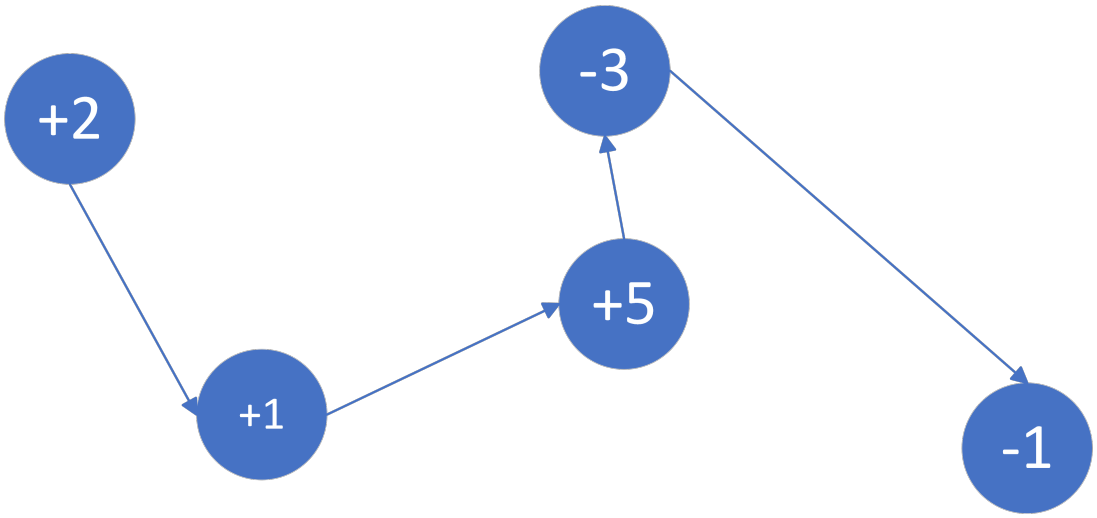

Figure 3.1: Illustration of bike rebalancing route

3.1 Survival Time

We can model the occupancy of each station as the Markov-modulated Poisson process (MMPP) [11]: with the time-varying Poisson-distributed birth and death rates $\lambda_v(t)$ and $\mu_v(t)$ (ie, arrival and departure rates), respectively. And we can make a reasonable assumption that the use of bikes is irrelevant to the current state of the stations, that is, the rates $\lambda_v(t)$ and $\mu_v(t)$ are independent of the state of the stations $m_v(t)$.

According to the Birth-Death Process(BDP) [17] to model the occupancy of the stations. We adopt the discrete time $\lambda_v(t)$ and $\mu_v(t)$ in the time frame T_r and the BDP slot T_p. Therefore, the mapping is as follows:

$$t \longrightarrow nT_r + kT_p \quad with \quad T_r = \lfloor t/T_p \rfloor, k = \lfloor (t - nT_r)/T_p \rfloor \tag{3.1}$$

In 3.1, n in the index of T_r(the time frame) and k is the index of T_p(the BDP slot). So far, we have made two assumptions, BDP and discrete time, which make it possible to estimate arrivals and departures accurately. After determining T_r and T_p, the MMPP parameters can be calculated with the available data set.

Intuitively, when the stations are empty or fully docked, users cannot rent or park bikes. Therefore, it may affect or change users' decision making. As a result, the true demand for bikes would not be accurately reflected in the historical data set. To reduce the impact of this scenario, we take these two states 0 and M_v as

absorbing states. Based on the formulations above, the survival time $S_v(t, m)$ of station v at time t with state m can be formally defined as the shortest time period τ after the probability $P_{m,a}(\tau; t)$ of reaching state $a \in \{0, M_v\}$ where $m \in M_v$ is the starting state at time t, until the probability $P_{m,a}(\tau; t)$ exceeds the preset threshold p_{th}, the expression as shown in 3.2

$$S_v(t, m) = inf\{\tau : P_{m,0}(\tau; t) + P_{m,M_v}(\tau; t) \geq p_{th}\} \tag{3.2}$$

It should be noted that $S_v(t, m)$ varies over time, it shows the variance of the birth rate $\lambda_v(t)$ and the death rate $\mu_v(t)$. The hyperparameter P_{th} controls the frequency of rebalancing operation, and a lower value results in a shorter survival time, as a consequence of more frequent rebalancing. Vice versa, a higher value of P_{th} produces a less frequent rebalancing. Here, we are facing a trade off; more frequent rebalancing can increase the overall satisfaction of the users, and it also increases the cost of the system.

The transition probability $P_{i,j}(t)$ can be calculated on the basis of the state distribution at each end of the previous time frame, since the birth and death rates are different according to different time frames. After using the *Markov chain* property, we get the following.

$$P_{i,j}(t) = P_{i,j}(nT_r + kT_p) = \sum_{l \in M_v} P_{i,l}(nT_r) P_{l,j}(kT_p), \quad i, j \in M_v, \tag{3.3}$$

as k and n are already given in 3.1. The transition matrix $P_{i,j}(nT_r + kT_p)$ after the k steps is the $(k+1) - th$ power of the one-step transition matrix. for the system in time frame n and slot k, we use 3.3 to compute transition matrices.

Calculating $P_{m,0}(t)$ and $P_{m,M_v}(t)$ at station v can be done by evaluating transition probabilities starting from state m for each time step. Then the length of the slot T_p will be our only focus, with birth and death rates $\lambda_v(t)$ and $\mu_v(t)$. And A_v and D_v denote the arrival and departure numbers at station v during the design slot length. For empty docks (birth) and bike demand (death), they are modeled as a Poisson process, with mean $\lambda_v T_p$ and $\mu_v T_p$, respectively. After a sequence of events has occurred, the coming state is the same regardless of the order of the events. Therefore, the sum of the number of arrivals and departures is what we really care about. Technically, this statement is not strictly accurate. The state may reach the absorbing state in the process, which can affect decision-making over time. However, such a probability is relatively low, and the impact on the

simplified model is neglected. The formulation of the probability $P_{i,j}$ from state i to state j in slot length T_p can be written as follows:

$$\begin{aligned} P_{i,j} &= Pr[A_v - D_v = j - i] \\ &= \sum_{l=-\infty}^{\infty} Pr[A_v = j - i + l] Pr[D_v = l] \\ &= \sum_{l=max\{0,j-i\}}^{+\infty} \frac{(\lambda_v T_p)^{-(j-i+l)} (\mu_v T_p)^{-l}}{l!(j-i+l)!} \end{aligned} \tag{3.4}$$

A_v and D_v follow the Poisson distribution in 3.4. According to [15], 3.4 can be written in the form of a truncated Skellam distribution:

$$P_{SK}(l; \mu_1, \mu_2) = e^{-(\mu_1+\mu_2)} \left(\frac{\mu_1}{\mu_2}\right)^{(l/2)} I_{|l|}(2\sqrt{\mu_1\mu_2}) \tag{3.5}$$

where $l = j - i$, $\mu_1 = \lambda_v T_p$(the mean of A_v) and $\mu_2 = \mu_v T_p$(the mean of D_v). $I_k(z)$ is the modified Bessel function of the first kind and of order k [1]:

$$\begin{aligned} I_\alpha(x) &= \lim_{N \to -\infty} \sum_{n=0}^{N} \frac{1}{n!\Gamma(n+\alpha+1)} \left(\frac{x}{2}\right)^{2n+\alpha} \\ \Gamma(z) &= \int_0^\infty x^{z-1} e^{-x} dx \end{aligned} \tag{3.6}$$

Given 3.4, we need to consider it in the finite range $\{0, \ldots, M_v\}$. The probability transition function of the absorbing states $\{0, M_v\}$ has the following form:

$$\begin{aligned} P_{i,0} &= \sum_{k=\infty}^{0} p_{SK}(k - i; \mu_1, \mu_2), \quad i \in M_v \backslash \{0, M_v\}, \\ P_{i,M_v} &= \sum_{k=M_v}^{+\infty} p_{SK}(k - i; \mu_1, \mu_2), \quad i \in M_v \backslash \{0, M_v\}. \end{aligned} \tag{3.7}$$

Intuitively, we know that when the state reaches the absorbing states 0 and M_v. Probability is unlikely to change. In other words, $P_{0,0} = P_{M_v,M_v} = 1$, while $P_{0,j} = 0, \forall j \neq 0$ and $P_{M_v,j} = 0, \forall j \neq M_v$.

To actually compute the probabilities on the time stamp t, we divide the process into three steps, since we show that time t has equivalence in 3.1:

1. Separate the problem in time frame T_r by exploiting the Markov property;

for the probabilities in $n > 0$, it is essential to compute the probabilities at the end of the frame $n' < n$

2. For computing the probabilities in slots length T_p within a single time frame n. We obtain it at the power of $k + 1$(indexing from 0), in the slot $k > 0$ within the time frame n.
3. Probabilities of transition at the beginning of the frame or slot($n = 0, k = 0$). Then we take advantage of equations 3.4 and 3.7.

For each time stamp t and state m, we are capable of computing the probability in the absorbing states 0 and M_v, without even mentioning the survival time $S_v(t, m)$ at the station v. Taking into account $S_v(t, m)$, the optimal state of each station should be around the mid value intuitively, and . Therefore, there exists an optimal value $m_v^\star(t)$ makes the optimal survival time$S_v^\star(t)$ meets such requirement:

$$m_v^\star(t) = \arg\max_{m \in M_v} S_v(t, m) \quad S_v^\star(t) = S_v(t, m_v^\star(t)), \quad v \in V. \tag{3.8}$$

Consequently, we would like to stay in the state $m_v^\star(t)$ with the time stamp t to maintain the longest survival, under which circumstances the rebalancing operation is not requested. Next, we will discuss rebalancing with the help of the optimal survival time.

3.2 Network wide optimization

Previously, we have introduced that the bike sharing system can be modeled as a graph $\mathcal{G}$. The number of vehicles that are rebalancing is expressed as X, and under a reasonable assumption that the subsets of the stations visited by each vehicle are disjoint. Each rebalancing vehicle has a capacity C, and the total number of rebalancing bikes cannot exceed this number. After that, we define a routing graph $\mathcal{H} = (\mathcal{V}', \mathcal{F})$, where $\mathcal{V}' \subseteq \mathcal{V} \cup \{0\}$ is the sum of the subsets of the stations that the rebalancing vehicles need to cover and $\{0\}$ is the vehicle storage station. $\mathcal{F}$ denotes the edges that need to be solved in the Vehicle Routing Problem (VRP) [19] on $(\mathcal{V}', \mathcal{E})$.

Regarding station occupancy, we use the vector to express it in the vector of the form $\mathbf{m}^{(\varnothing)}(t) = [m_1^{(\varnothing)}(t), \ldots, m_{\mathcal{V}}^{(\varnothing)}(t)]$. The occupancy vector after rebalancing is denoted as $\mathbf{m}^{(\mathcal{H})}(t)$. The reward function can be designed on the basis of the time before the rebalancing operation is performed. For example, the difference between the minimum survival times among all stations after and before

rebalancing operations. For every rebalancing operation, there is a fixed cost to deploy a truck to carry out the operation, and the other cost is related to the distance $D_x(\mathcal{H})$ of each truck $x \in [1, \ldots, X]$. Based on all of the above assumptions, the optimization function $\tilde{f}(\mathcal{H}, t)$ can be designed as

$$\tilde{f}(\mathcal{H}, t) = \left(\min_{v \in V} S_v(t, m_v^{\mathcal{H}}(t)) - \min_{v \in V} S_v(t, m_v^{\varnothing}(t)) \right) - \alpha X - \beta \sum_{x=1}^{X} D_x(\mathcal{H}) \tag{3.9}$$

where α and β are the cost of service on the number of rebalancing vehicles and the total distance, respectively. Survival times $S_v(\cdot, \cdot)$ are expressed in 3.2.

Furthermore, we would like to introduce another parameter γ to set a threshold on survival time to eliminate longer survival times, which may have unbearable estimation errors resulting in meaningless rebalancing operations. The updated optimization function would be the following.

$$\begin{aligned} f(\mathcal{H}, t) = & \left(\min \left\{ \min_{v \in V} S_v(t, m^{\mathcal{H}_v(t)}), \gamma \right\} - \min \left\{ \min_{v \in V} S_v(t, m_v^{\varnothing}(t)), \gamma \right\} \right) \\ & - \alpha X - \beta \sum_{x=1}^{X} D_x(\mathcal{H}) \end{aligned} \tag{3.10}$$

As mentioned above, the operating day is discretized into frames T_r. Problems can be solved periodically for every T_r in the routing graph $\mathcal{H} = (\mathcal{V}', \mathcal{F})$. The results we will obtain from solving the problem will be to determine the number of bikes that need rebalancing operations at each station and to plot the route for each rebalancing vehicle. It should be noted that if the optimization function $f(\mathcal{H}, t)$ is negative at time t and station $\mathcal{V}' \neq 0$, no rebalancing operation is required (the reward is less than the cost).

The set of stations to visit, $\mathcal{V}'$, and the route of the vehicle, $\mathcal{F}$, should be jointly determined because the optimization function depends on the distance covered by the truck. To simplify the calculation of $\mathcal{V}'$, we first assume that we know the minimum path length of the rebalancing route for a given set of nodes. We can then derive $\mathcal{V}'$ and describe how to compute this distance. The calculation is simplified by a theorem 3.2.1 taken from [7] that states that $\mathcal{V}'$ contains the nodes with the shortest survival times. All subsets that do not satisfy this theorem 3.2.1 can be safely discarded as suboptimal.

Theorem 3.2.1. *If* $v \in \mathcal{V}'$, *then* $S_v(t, m_v^{\varnothing}(t)) < S_u(t, m_u^{\varnothing}(t)) \quad \forall u \in \mathcal{V} \backslash \mathcal{V}'$

The procedure for estimating $\mathcal{V}'$ is described in Algorithm 1. First, $\mathcal{V}'$ contains the deposit point(Line 1); then identifying the node $v(i) \in \mathcal{V} \backslash \mathcal{V}'$ in each iteration i (Line 9). Calculation of the reward after adding $v(i)$ to $\mathcal{V}'$ when the survival time is optimal(Line 10), plus the cost to cover the route(Line 14). In particular, the reward is designed as the difference between the smallest survival time in $\mathcal{V}'$ and the smallest survival time before rebalancing (Line 4 and Line 11), as for the cost which is decided by distance (Line 14). When the optimization function(the difference between reward and cost) increases, we add node $v(i)$ to $\mathcal{V}'$ (Line 15). The termination condition of the iteration is either that the station $v \in \mathcal{V}$ has been added to $\mathcal{V}'$, that is, the iteraion condition is met, or that the optimization function has converged because the minimum optimal survival time for all $\omega \in \mathcal{V}'$ after rebalancing is less than the current optimal survival time of each node(Line 19).

The idea of solving the dynamic rebalancing problem can be divided into two parts: (i) plot the route to connect the stations that need rebalancing operation plus consideration of the vehicle capacity and the traveling time, and (ii) work out which stations have to be visited. For the first part, we calculate the function $path$ to connect all nodes with the shortest distance under the constraint of vehicle capacity. The latter part has been widely discussed in 2. However, route plotting can adopt other algorithms or strategies depending on different factors.

As for the *distance* function, the sum of requested bikes of rebalancing is not zero. Reflecting on the real world is that there are always some bikes that should be taken away or added to the stations from some other place. Here we make the assumption that there is a warehouse at the base with infinite bikes to take and space to store the bikes. As a result, when the truck does not have enough bikes to add to the next station or space to take the bikes back. The truck needs to go back to the base to reload the truck to the maximum capacity or unload all bikes to the empty.

The route plotting strategy can be solved by the famous VRP, which is a problem that has been solved before and possibly at the lowest cost. As mentioned before, VRP is an NP-hard problem; there are only existing suboptimal solutions. We can make the following assumptions to reduce the computational complexity of VRP:

1. For the $distance$ function in Line 13, we use the Euclidean distance to represent the distance metric for the edges in $\mathcal{E}$. In other words, $d(v_i, v_j) = d(v_j, v_i)$ for any two stations i and j.

Algorithm 1 Rebalancing routing plotting

$i = 0, \mathcal{V}' = \{0\}, f(\mathcal{H}, t) = 0, done = 0$ ▷ Initialization
α, β, γ, X ▷ Hyperparameter initialization
$\mathbf{S}(t) = [S_1(t, m_1^{\varnothing}), \ldots, S_{|\mathcal{V}|}|(t, m_{|\mathcal{V}|}^{\varnothing})]$ ▷ Survival time vector before rebalancing for each station $v \in \mathcal{V}$
$\sigma = \min_{\omega \in \mathcal{V}} \mathbf{S}(t)$ ▷ Smallest survival time before rebalancing
$\sigma = \min\{\sigma, \gamma\}$ ▷ Put a threshold on the survival time
$\mathbf{S}^{\star}(t) = [S_1^{\star}(t), \ldots, S_{|\mathcal{V}|}^{\star}(t)]$ ▷ Optimal survival time vector for each station $v \in \mathcal{V}$
while $(i < \mathcal{V}) \& (done = 0)$ **do**: ▷ Until the visited nodes can improve the overall reward
    $i \leftarrow i + 1$ ▷ The iteration
    $v(i) \leftarrow \arg\min_{\omega \in \mathcal{V} \setminus \mathcal{V}'} \mathbf{S}(t)$ ▷ Choose unvisited node with smallest survival time
    $[\mathbf{S}]_{v(i)}(t) \leftarrow [\mathbf{S}^{\star}]_{v(i)}(t)$ ▷ Update the survival time of each node $v(i)$
    $reward \leftarrow \min\{\min \mathbf{S}(t), \gamma\} - \sigma$ ▷ Update the reward
    $\mathcal{F} \leftarrow path\{\mathcal{V}' \cup v(i)\}$ ▷ Determining the rebalancing path $\mathcal{F}$
    $D \leftarrow distance\{\mathcal{F}\}$ ▷ The distance of the path $\mathcal{F}$
    $cost \leftarrow \alpha X + \beta D$ ▷ Update the cost
    if $reward - cost > f(\mathcal{H}, t)$ **then**:
        $\mathcal{V}' \leftarrow \mathcal{V}' \cup v(i)$ ▷ Include node $v(i)$ in the rebalancing
        $f(\mathcal{H}, t) \leftarrow reward - cost$
    end if
    if $\min_{\omega \in \mathcal{V}'}\{[\mathbf{S}^{\star}(t)]_{\omega}\} < \min_{\omega \in \mathcal{V} \setminus \mathcal{V}'}\{[\mathbf{S}(t)]_{\omega}\}$ **then**
        $done \leftarrow 1$ ▷ The optimization function $f(\mathcal{H}, t)$ converge
    end if
end while

Algorithm 2 Distance calculation

```
bike_numbers ▷ the number of bikes need rebalancing operation at each station
capacity ▷ the capacity of the truck
carrying = 0 ▷ initialize the the load of truck at the base
the_next_station ▷ the next station is the closest station to the current station
while visited_stations ≠ all_stations do ▷ until all stations have been visited
    if sum(bike_numbers) > 0 then
        carrying = capacity
    else
        carrying = 0
    end if
    if 0 ≤ carrying + bike_numbers[the_next_station] ≤ capacity then
        carrying = carrying + bike_numbers[the_next_station] ▷ update the load on the truck
    else
        the_next_station ▷ update the next station by selecting the next closest station
    end if
end while
```

2. Only one vehicle will be taken into consideration, that is, $X = 1$, with capacity C.

For the $path$ function is designed with the idea of a greedy algorithm. especially the *Nearest Neighborhood* algorithm, which chooses the closest station heuristically at each step. Set $U_i \subseteq \mathcal{V}'$ as the nodes visited up to iteration i. The starting point of the vehicle can be defined as $U_0 = u(0) = \{0\}$, while $i \in \{1, \ldots, |\mathcal{V}'| - 1\}$. It is noted that $\{0\}$ is included in $\mathcal{V}'$, so iteration i stops until $|\mathcal{V}'|-1$. And we have $U_i = U_{i-1} \cup \{u_i\}$, where u_i denotes the station visited in the $i-th$ iteration. Therefore,

$$u_i = \arg \min_{\omega \in \mathcal{V}' \setminus U_{i-1}} d(\omega, u_(i-1)) \tag{3.11}$$

For the function $distance$, Euclidean distance $d(x, y)$ between node x and y. The implementation of $NearestNeighborhood$ on the total travel path can be expressed as

$$D(\mathcal{H}) = \sum_{i-1}^{|\mathcal{V}'|-1} d(u_i, u_{i-1}) + d(u_{|U|-1,0}) \tag{3.12}$$

4

Analysis

In order to compare the optimization in how it works under different scenarios, we have developed the following circumstances to design the simulation and record the corresponding key data. Here is how the design works and the data collected.

1. Optimization is not performed to get the overall failure rates that include the empty and maximum states.
2. Dynamic rebalancing optimization is performed and the rebalancing vehicle has unlimited capacity. The failure rate and the overall cover distance have been compared.
3. Dynamic rebalancing optimization performed and a single rebalancing vehicle with limited capacity. The failure rate and distance are calculated.

while conducting the simulation, we were facing a few problems, In order to make the modeling more generalizable to the real problem and computationally affordable. The following assumptions need be made,

1. The data from Citibike can stand for the demand of the bikes' use. The true demand is not yet known due to the censoring effect. However, the lower bound of the demand can be represented by the data.
2. The bikes that users use are not included in the optimization framework; the reason is that part of the data is not available.
3. If the user goes to a station but cannot find a bike there or the user with the bike goes to park, the bike cannot find a stall available to lock the bike. Both scenarios are considered as they exit the system.
4. At the beginning of the month, the state of all stations is set as half of the entire capacity of the station. As the initial states of the stations are not known, this premise would be a necessity to start the simulation.

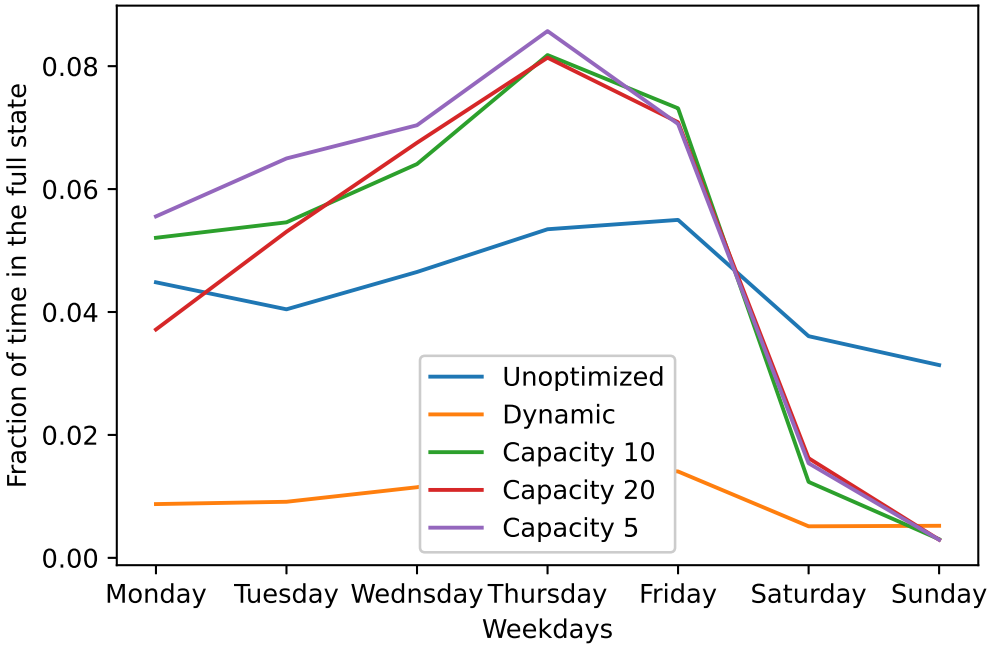

Figure 4.1: The full state failure rate in June 2017

All the above assumptions have zero effect on the optimization framework, while the final numerical results may change slightly, but the difference between different scenarios should be maintained. Simulations are carried out only in June 2017, and numerical results are presented on weekdays. We take the cost of performing a single rebalancing operation, α, to 1800 s (30 min), and the parameter β represents the cost of deploying the truck per meter, set at 0.02 s / m. As for the truck capacity, C is simulated with 5, 10, 20.

As shown in 4.1, after the rebalancing operation the full state failure rates are even worse than in the unoptimized scenario. However, the failure rates of the empty state are much improved in Fig.4.2. Overall, total failure rates have improved considerably according to Fig.4.3. Even though the empty state failure rates are not as expected, the total failure rates are significantly improved. The full state failure rates are even worse than the unoptimized scenario. However, the full state failure rates are relatively smaller than the empty state failure rates. As a result, empty state failure rates are the main factor in total failure rates by comparing Fig.4.2 and Fig.4.3. Obviously, the empty state matters more to the overall failure state by comparing Fig.4.2 and Fig.4.3. Without any limits on the capacity of the truck, which is the dynamic scenario in Fig.4.3, the failure rate is

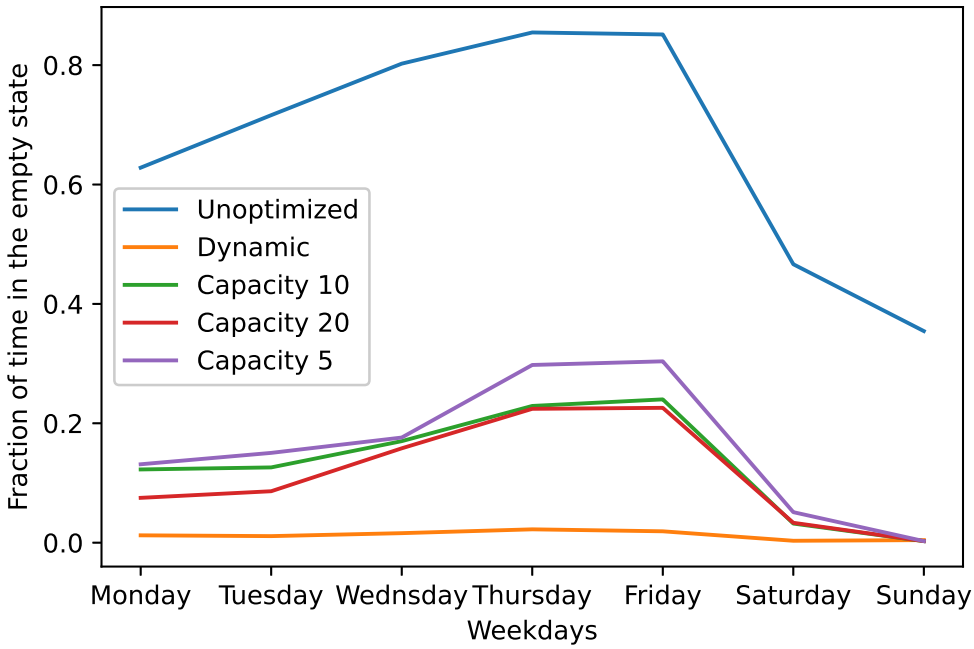

Figure 4.2: The empty state failure rate in June 2017

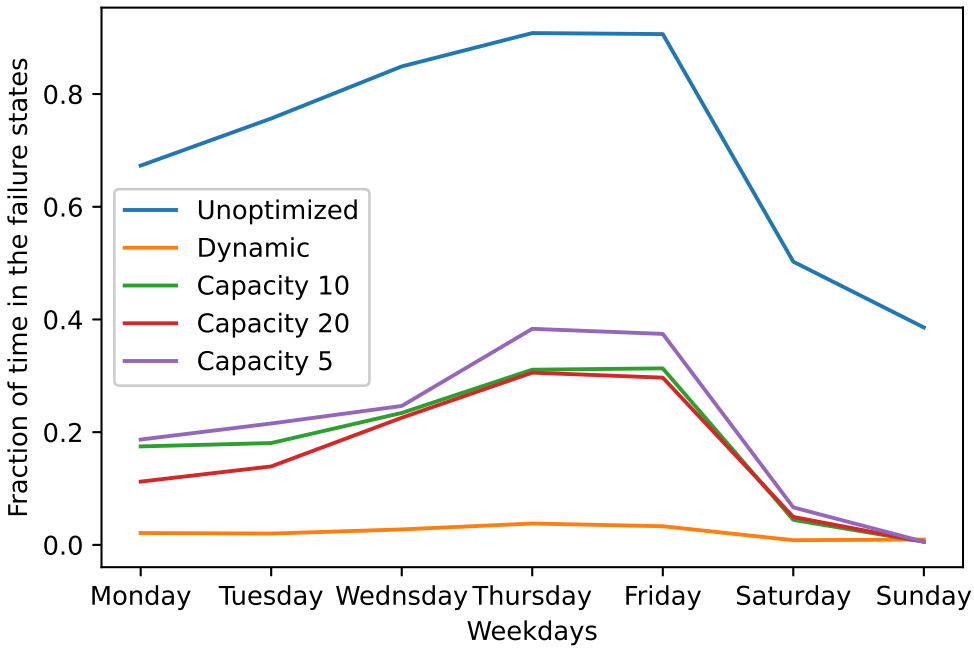

Figure 4.3: The total failure rate in June 2017

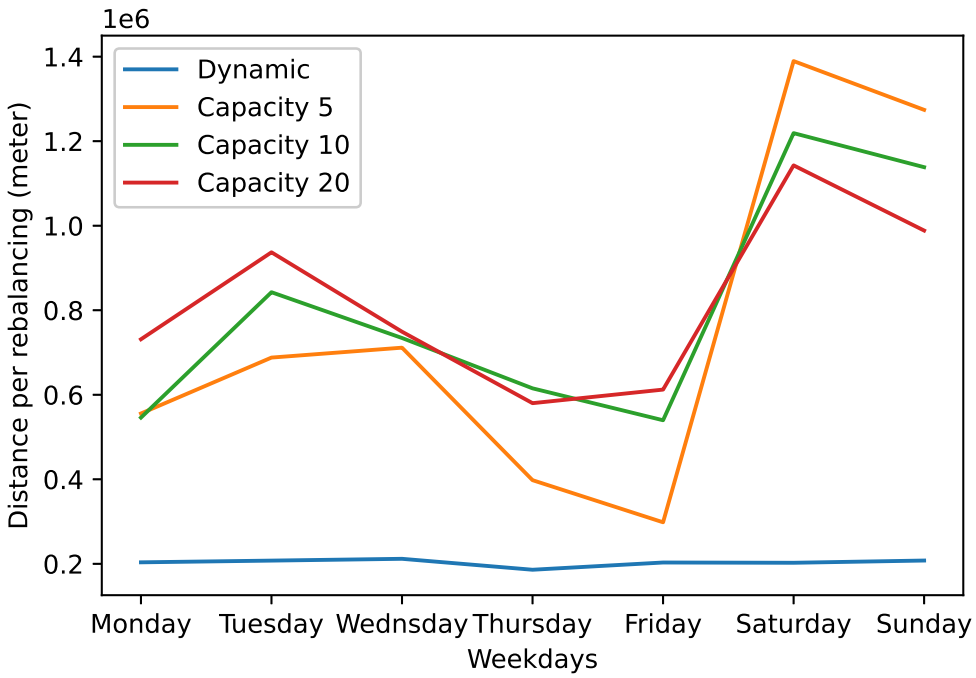

Figure 4.4: The distance covered by the truck by different capacities in June 2017

roughly four times better than the situation with the limit of the capacity of the truck. Furthermore, with a larger truck capacity, failure rates could be relatively higher than those with smaller trucks if we observe Fig.4.3.

It is clear to see in Fig.4.4, the 20-capacity truck covered the most distances under a low failure rate during most of the week. On the contrary, the distance on weekends is lower as the capacity of the truck decreases, the reason is that the failure rates are close to each other from 4.3. Additionally, with higher capacity, the truck does not need to go back to the base to retrieve the bike or remove the truck load. We move to Fig.4.5, it shows how many trucks it needs to do in a single rebalancing on average a day. Apparently, with larger capacity, it requires fewer trucks to do the rebalancing.

Looking at Fig.4.6, red dots represent stations that need to be rebalanced, while the purple dots represent stations without rebalancing operation at all. It shows the pattern that the stations in Upper Manhattan and Williamsburg require less rebalancing compared to Lower Manhattan and Downtown Brooklyn. Some of the guessing we have is that there are more commercial activities conducted in Lower Manhattan and Downtown Brooklyn. Therefore, commuters are the main users of the bike sharing system. In addition, as many visitors

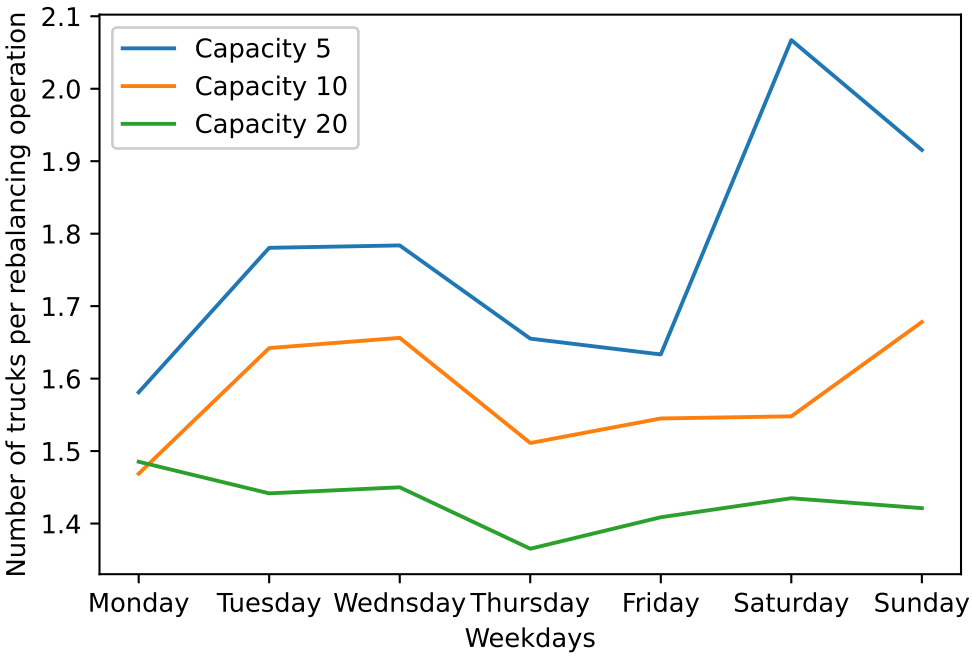

Figure 4.5: The trips one truck takes from base in June 2017

want to see the sights of downtown, such as the Brooklyn Bridge, the World Trade Center, Battery Park, and the Seaport. Another, Upper Manhattan has more hills, which discourages cyclists from going there, and it has more metro lines that pass through.

After removing non-rebalancing stations, Fig.4.7 represents the number of visits the truck visited a month. It shows that the most visited station is Henry & Poplar Station. The station is located around the subway station in downtown Brooklyn. Where it is a transition point for metro users and bike users considering that downtown Brooklyn has commercial and residential neighborhoods. AS for the second most rebalanced station is in upper Manhattan, a place near the Lexington Avenue / 59 Street metro station; it is also a transition point where people change the metro to bicycles.

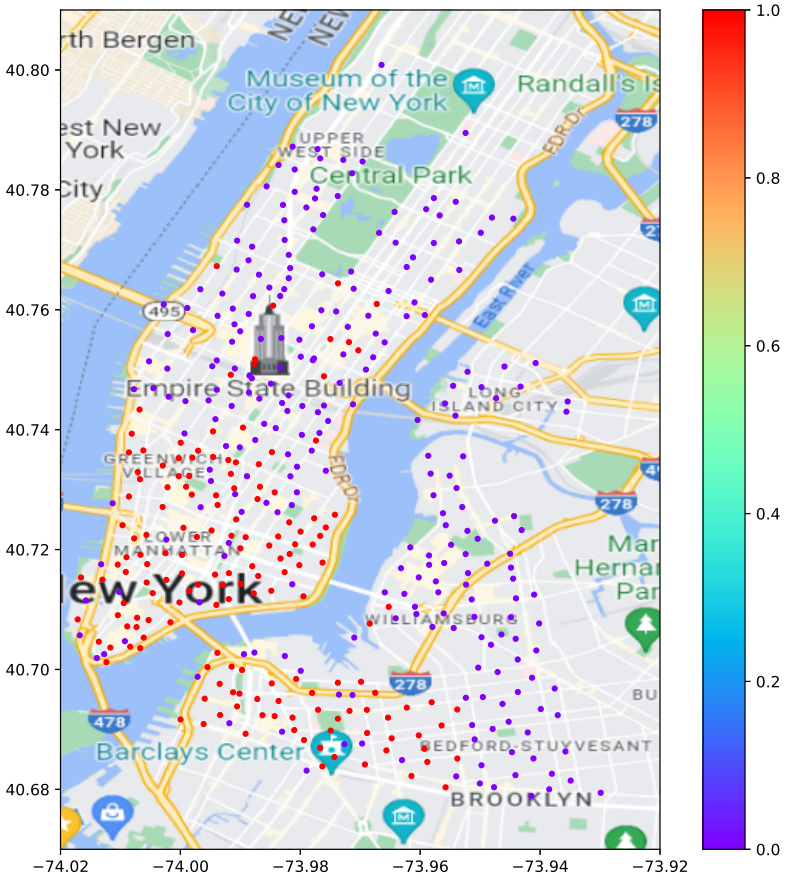

Figure 4.6: The stations need to be rebalanced in June 2017

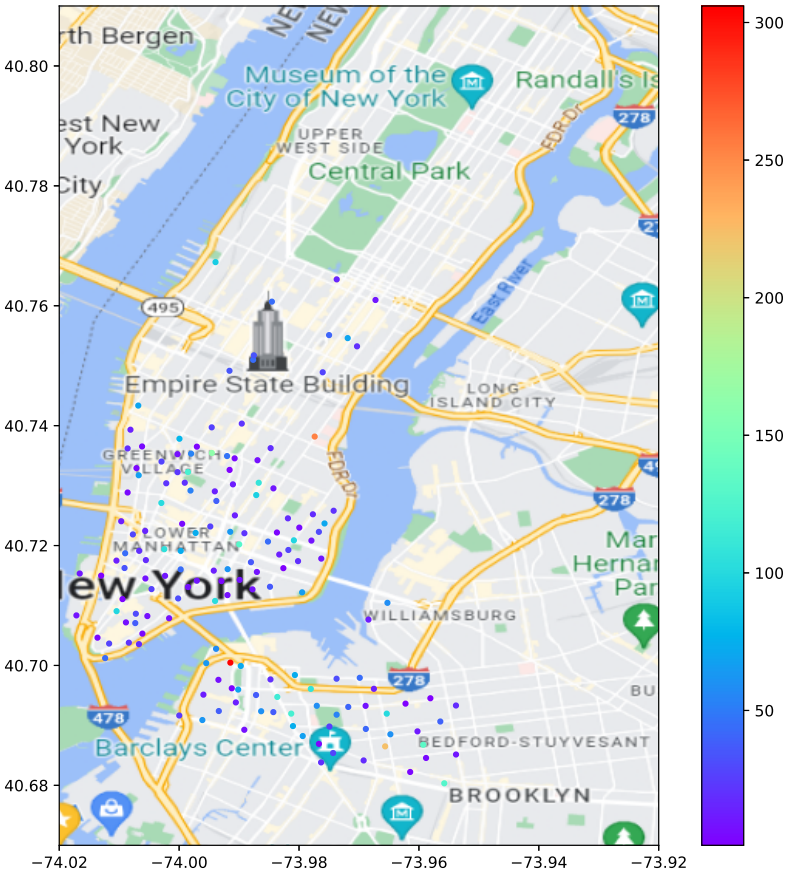

Figure 4.7: The stations need to be visited in June 2017

5

Conclusions

In this work, we have adopted a dynamic rebalancing technique in the bike sharing system with considering a more realistic scenario, which is the rebalancing vehicle has a limited capacity and when there a no space or bikes available on the truck, it returns to the base to unload the bikes or reload the bikes depending on the situations. To identify the stations where a rebalancing operation was required, we chose to analyze the previous data under the assumption without any human intervention. Specifically, MMPP with BDP helps to decide the optimal survival time, where the birth and death rates are depending on the time frame and time slot. After getting the survival of each station, we designed an incentive to decide which stations are worth including into the route by the greedy algorithm. More realistically, the truck has a capacity that would force the truck to go back to the base to clear the truck load or full load the truck depending on the situation.

The results were obtained based on the assumptions made and the models designed in 3, which are reasonable. With a larger capacity, the truck needs more trips from the base to complete the rebalancing. The distance it covers would also decrease; as a result, it is more likely to be incentivized to add more stations to the rebalancing routes. However, failure rates and distance will not change once the truck capacity reaches some point. Intuitively, the truck with the more capacity would cost more money and have failure rates compared to other trucks of a lower capacity would make another trade-off.

It should be noted that the most visited stations are *Henry and Poplar Station* and *Lexington Av/59 St Station*. An option to help us is to arrange some special

trucks only responsible for them in order to meet the demand to achieve the maximum satisfactory of the users. Another is that the visited stations are in the area of Lower Manhattan and downtown Brooklyn. It could be better to set the base near that neighborhood.

References

[1] Milton Abramowitz, Irene A Stegun, and Robert H Romer. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. 1988.

[2] Pedram Beigi et al. "Station Reallocation and Rebalancing Strategy for Bike-Sharing Systems: A Case Study of Washington DC". In: *arXiv preprint arXiv:2204.07875* (2022).

[3] Jan Brinkmann, Marlin W Ulmer, and Dirk C Mattfeld. "Short-term strategies for stochastic inventory routing in bike sharing systems". In: *Transportation Research Procedia* 10 (2015), pp. 364–373.

[4] Daniel Chemla, Frédéric Meunier, and Roberto Wolfler Calvo. "Bike sharing systems: Solving the static rebalancing problem". In: *Discrete Optimization* 10.2 (2013), pp. 120–146.

[5] Dan Chen et al. "Alphabet partitioning techniques for semiadaptive huffman coding of large alphabets". In: *IEEE Transactions on Communications* 55.3 (2007), pp. 436–443.

[6] Federico Chiariotti et al. "A bike-sharing optimization framework combining dynamic rebalancing and user incentives". In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 14.3 (2020), pp. 1–30.

[7] Federico Chiariotti et al. "A dynamic approach to rebalancing bike-sharing systems". In: *Sensors* 18.2 (2018), p. 512.

[8] Claudio Contardo, Catherine Morency, and Louis-Martin Rousseau. *Balancing a dynamic public bike-sharing system*. Vol. 4. Cirrelt Montreal, 2012.

[9] Yubin Duan, Jie Wu, and Huanyang Zheng. "A greedy approach for vehicle routing when rebalancing bike sharing systems". In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, pp. 1–7.

[10] Ahmadreza Faghih-Imani et al. "An empirical analysis of bike sharing usage and rebalancing: Evidence from Barcelona and Seville". In: *Transportation Research Part A: Policy and Practice* 97 (2017), pp. 177–191.

[11] Wolfgang Fischer and Kathleen Meier-Hellstern. "The Markov-modulated Poisson process (MMPP) cookbook". In: *Performance evaluation* 18.2 (1993), pp. 149–171.

[12] Iris A Forma, Tal Raviv, and Michal Tzur. "A 3-step math heuristic for the static repositioning problem in bike-sharing systems". In: *Transportation research part B: methodological* 71 (2015), pp. 230–247.

[13] Zulqarnain Haider et al. "Inventory rebalancing through pricing in public bike sharing systems". In: *European Journal of Operational Research* 270.1 (2018), pp. 103–117.

[14] Runqiu Hu et al. "Dynamic rebalancing optimization for bike-sharing system using priority-based MOEA/D algorithm". In: *IEEE Access* 9 (2021), pp. 27067–27084.

[15] SKELLAM JG. "The frequency distribution of the difference between two Poisson variates belonging to different populations." In: *Journal of the Royal Statistical Society. Series A (General)* 109.Pt 3 (1946), pp. 296–296.

[16] Ahmed Abdelmoumene Kadri, Imed Kacem, and Karim Labadi. "A branch-and-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems". In: *Computers & Industrial Engineering* 95 (2016), pp. 41–52.

[17] Alan Krinik and Carrie Mortensen. "Transient probability functions of finite birth–death processes with catastrophes". In: *Journal of statistical planning and inference* 137.5 (2007), pp. 1530–1543.

[18] Bahman Lahoorpoor et al. "Spatial cluster-based model for static rebalancing bike sharing problem". In: *Sustainability* 11.11 (2019), p. 3205.

[19] Gilbert Laporte. "The vehicle routing problem: An overview of exact and approximate algorithms". In: *European journal of operational research* 59.3 (1992), pp. 345–358.

[20] Benjamin Legros. "Dynamic repositioning strategy in a bike-sharing system; how to prioritize and how to rebalance a bike station". In: *European Journal of Operational Research* 272.2 (2019), pp. 740–753.

[21] Hao Luo et al. "Comparative life cycle assessment of station-based and dock-less bike sharing systems". In: *Resources, Conservation and Recycling* 146 (2019), pp. 180–189.

[22] Peter Midgley. "The role of smart bike-sharing systems in urban mobility". In: *Journeys* 2.1 (2009), pp. 23–31.

[23] Bernard ME Moret and Henry D Shapiro. "An empirical analysis of algorithms for constructing a minimum spanning tree". In: *Algorithms and Data Structures: 2nd Workshop, WADS'91 Ottawa, Canada, August 14–16, 1991 Proceedings 2*. Springer. 1991, pp. 400–411.

[24] Aritra Pal and Yu Zhang. "Free-floating bike sharing: Solving real-life large-scale static rebalancing problems". In: *Transportation Research Part C: Emerging Technologies* 80 (2017), pp. 92–116.

[25] Lu-Yi Qiu and Ling-Yun He. "Bike sharing and the economy, the environment, and health-related externalities". In: *Sustainability* 10.4 (2018), p. 1145.

[26] Rubén Ruiz and Thomas Stützle. "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem". In: *European journal of operational research* 177.3 (2007), pp. 2033–2049.

[27] Carlos M Vallez, Mario Castro, and David Contreras. "Challenges and opportunities in dock-based bike-sharing rebalancing: A systematic review". In: *Sustainability* 13.4 (2021), p. 1829.

[28] Mingshu Wang and Xiaolu Zhou. "Bike-sharing systems and congestion: Evidence from US cities". In: *Journal of transport geography* 65 (2017), pp. 147–154.

[29] Xu Wang et al. "Bike sharing rebalancing problem with variable demand". In: *Physica A: Statistical Mechanics and its Applications* 591 (2022), p. 126766.

[30] Jie Wu. "Challenges and opportunities in algorithmic solutions for re-balancing in bike sharing systems". In: *Tsinghua Science and Technology* 25.6 (2020), pp. 721–733.

[31] Peiyu Yi, Feihu Huang, and Jian Peng. "A rebalancing strategy for the imbalance problem in bike-sharing systems". In: *Energies* 12.13 (2019), p. 2578.

Acknowledgments