A Model Predictive Control approach on Dynamic Bike Reposition

Guilherme Duarte Tavares Saraiva

Thesis to obtain the Master of Science Degree in Aerospace Engineering

Supervisor(s): Professor Cláudia Alexandra Magalhães Soares
Doctor Manuel Ricardo de Almeida Rodrigues Marques

Examination Committee
Chairperson: Prof. José Fernando Alves da Silva
Supervisor: Prof. Cláudia Alexandra Magalhães Soares
Member of the Committee: Prof. Alexandre José Malheiro Bernardino

November 2019
Acknowledgments

I would like to thank my Supervisors, Professors Manuel Marques and Cláudia Soares, as the work developed throughout this thesis would not have been possible without them. Their constant guidance and interest in my work made me grateful for the opportunity to have had them as supervisors.

I am extremely grateful to my parents and my sister, for the unconditional love and for always being patient and supportive of all my decisions.

Finally, I also take this opportunity to thank all my friends that always been there for me during these 5 years independently of the situation, who will surely never be forgotten. To my friends at ISR, who kept me company during the long hours of work in the last months, in this second home of mine.
Abstract

Over the last few years, Bike Sharing Systems (BSSs) have emerged as a solution to tackle the “last mile” problem associated with public transportation, as well as to improve the sustainability and decarbonization of cities. One major challenge associated with BSSs is the need to respond to the fluctuating demands by the users in order to provide a good customer service. The key solutions to this problem include a good prediction algorithm and an efficient rebalancing method.

With the introduction of the General Data Protection Regulation law in Europe, current datasets of European bike sharing systems are restricted in terms of individual movements due to privacy concerns. This constraint restricts our data model to a minimum, having access to nothing but the number of bicycles in each station.

Building on prior research, a Meteorology Similarity Weighted K-nearest-neighbors algorithm is adapted to be used to predict both the pick up and drop off demands, on a station level basis, considering the most similar days in terms of weather features.

We propose a convex Model Predictive Control approach to tackle the dynamic repositioning problem, which considers the station level demands and outputs a system based solution while trying to minimize the customer loss and at the same time reducing the reposition cost associated. The MPC reposition method was evaluated using a simulator to model the system dynamics with real-world datasets, where it was compared against existent reposition methods in order to evaluate its performance.

Keywords: Demand prediction, Rebalancing, Dynamic Reposition, Model Predictive Control, Bike Sharing System
Resumo

Durante os últimos anos, os sistemas de partilha de bicicletas têm surgido como uma solução para combater o problema do “última milha” associado aos transportes públicos, bem como para aumentar a sustentabilidade e descarbonização das cidades. Um dos principais desafios associados aos sistemas de bicicletas partilhadas passa pela necessidade de responder à procura flutuante dos utilizadores do sistema, passando pelo uso de um algoritmo de previsão bem como de um método de reposição eficiente. Com a introdução da lei de proteção de dados na União Europeia, os conjuntos de dados disponíveis dos sistemas de bicicletas partilhadas na Europa têm limitações devido as questões de privacidade. Deste modo, o nosso modelo está constrangido a um uso de dados agregados, não permitindo identificar viagens individuais, tendo apenas acesso ao número de inventário em cada estação individual.

Com base em trabalho anteriormente desenvolvido, utiliza-se um algoritmo “Meteorology Similarity Weighted K-Nearest-Neighbors” para prever a procura de bicicletas e de docas. Esta previsão é feita a nível individual por estação, considerando os dias mais semelhantes em termos de meteorologia.

Propomos uma abordagem ao problema da reposição dinâmica com recurso ao Modelo de Controlo Preditivo, que considera a procura em cada estação individualmente e cria uma solução global para todo o sistema, enquanto minimiza a perda para o consumidor e ao mesmo tempo o custo de reposição associado. O modelo é avaliado usando um simulador que modela as dinâmicas do sistema com dados reais, onde é comparado com outros métodos de reposição existentes, avaliando assim a sua performance.

**Keywords:** Previsão da Procura, Rebalanceamento, Reposição dinâmica, Controlo Preditivo, Bicicletas partilhadas
## Contents

<table>
<thead>
<tr>
<th>List of Figures</th>
<th>xi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nomenclature</td>
<td>xiii</td>
</tr>
<tr>
<td>Acronyms</td>
<td>xv</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Industry Parallelism</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Objectives and Contributions</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Thesis Outline</td>
<td>2</td>
</tr>
<tr>
<td>2 State of the Art</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Bike Sharing Systems</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Related Work</td>
<td>3</td>
</tr>
<tr>
<td>2.2.1 Demand Prediction</td>
<td>4</td>
</tr>
<tr>
<td>2.2.2 Rebalancing</td>
<td>5</td>
</tr>
<tr>
<td>User-Based Rebalancing</td>
<td>6</td>
</tr>
<tr>
<td>Vehicle-based Repositioning</td>
<td>6</td>
</tr>
<tr>
<td>3 Exploratory Data Analysis and Pre Processing</td>
<td>9</td>
</tr>
<tr>
<td>3.1 Concepts and Notation</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Meteorology Data Extraction</td>
<td>10</td>
</tr>
<tr>
<td>3.3 New York</td>
<td>11</td>
</tr>
<tr>
<td>3.4 Lisbon</td>
<td>12</td>
</tr>
<tr>
<td>3.5 Data Visualization</td>
<td>13</td>
</tr>
<tr>
<td>3.6 Discussion</td>
<td>16</td>
</tr>
<tr>
<td>4 MPC for Rebalancing</td>
<td>17</td>
</tr>
<tr>
<td>4.1 Data Model</td>
<td>17</td>
</tr>
<tr>
<td>4.2 MPC Background</td>
<td>18</td>
</tr>
<tr>
<td>4.3 Problem Statement and Notation</td>
<td>19</td>
</tr>
<tr>
<td>4.4 User movements Prediction</td>
<td>21</td>
</tr>
<tr>
<td>Strategy 2: Inventory K Nearest Neighbours(IKNN)</td>
<td>22</td>
</tr>
<tr>
<td>Strategy 3: Meteorology Similarity Weighted K-Nearest-Neighbors (MSWK)</td>
<td>23</td>
</tr>
<tr>
<td>Performance Comparison</td>
<td>24</td>
</tr>
<tr>
<td>Regularizing the estimated user movements</td>
<td>24</td>
</tr>
<tr>
<td>Matrix B</td>
<td>25</td>
</tr>
<tr>
<td>4.5 Optimizing User Service</td>
<td>25</td>
</tr>
</tbody>
</table>
5 Results

5.1 Simulator .......................................................... 29
5.2 Outline ............................................................... 29
5.3 Metrics ................................................................. 29
5.4 Inputs to the Simulator ............................................. 30
5.5 Users creation and Trip Generator .............................. 31
5.6 Repositioning Functions ............................................ 34
  5.6.1 Fixed Routes ...................................................... 34
  5.6.2 Pool System ...................................................... 34
5.7 MPC for Rebalancing ............................................... 34
  5.7.1 Metrics Evaluation ............................................... 35
5.8 MPC Reposition Cost Reduction ................................. 37
  5.8.1 MPC regularization ............................................. 37
5.9 Test case: Lisbon ..................................................... 41
5.10 Performance review ............................................... 42

6 Discussion and Future Work ....................................... 43

Bibliography .............................................................. 45

A Proof of convexity of our MPC ....................................... 49
  A.1 Notation ............................................................ 49
  A.2 Convex Functions ................................................... 49
  A.3 MPC ................................................................. 50
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Different episodes over the day</td>
<td>5</td>
</tr>
<tr>
<td>3.1</td>
<td>Median Bike Rents for each hour of the day</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>Mean Hourly Bike Rental over relative humidity</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>Mean Hourly Bike Rental over visibility</td>
<td>14</td>
</tr>
<tr>
<td>3.4</td>
<td>Mean Hourly Bike Rental over Wind Speed</td>
<td>15</td>
</tr>
<tr>
<td>3.5</td>
<td>Influence of Weather Conditions on the Mean Hourly Bike rent demand</td>
<td>15</td>
</tr>
<tr>
<td>3.6</td>
<td>Mean Hourly Bike Rental over temperature</td>
<td>16</td>
</tr>
<tr>
<td>4.1</td>
<td>Block Diagram of a MPC controller</td>
<td>19</td>
</tr>
<tr>
<td>4.2</td>
<td>Station graph example with 4 stations</td>
<td>20</td>
</tr>
<tr>
<td>4.3</td>
<td>Monthly histograms of user movements for the Citibike system, from where data will be extracted for the Average Value Method (AV), Most Probable Value (MP) and Monthly Hourly Average (MHA) methods</td>
<td>22</td>
</tr>
<tr>
<td>4.4</td>
<td>MAE for the prediction of pick up demand and drop off demand</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>Service Region</td>
<td>32</td>
</tr>
<tr>
<td>5.2</td>
<td>Demand for each system</td>
<td>33</td>
</tr>
<tr>
<td>5.3</td>
<td>Alarms definition scheme for continuous reposition</td>
<td>34</td>
</tr>
<tr>
<td>5.4</td>
<td>Hourly Reposition Metric, for each reposition method</td>
<td>35</td>
</tr>
<tr>
<td>5.5</td>
<td>Hourly Bad Service Metric, for each reposition method</td>
<td>36</td>
</tr>
<tr>
<td>5.6</td>
<td>Death Penalty Metric comparison</td>
<td>36</td>
</tr>
<tr>
<td>5.7</td>
<td>Users that leave the system for Citibike and Bicing</td>
<td>37</td>
</tr>
<tr>
<td>5.8</td>
<td>Reposition metric for different values of the parameter $\lambda$</td>
<td>38</td>
</tr>
<tr>
<td>5.9</td>
<td>Bad Service metric for different values of the parameter $\lambda$</td>
<td>38</td>
</tr>
<tr>
<td>5.10</td>
<td>Death metric for different values of $\lambda$</td>
<td>39</td>
</tr>
<tr>
<td>5.11</td>
<td>Trade-off between Reposition metric and Bad Service+Death metrics in New York</td>
<td>39</td>
</tr>
<tr>
<td>5.12</td>
<td>Trade-off between Reposition metric and Bad Service in Barcelona</td>
<td>40</td>
</tr>
<tr>
<td>5.13</td>
<td>Different metrics throughout the day in the city of Lisbon</td>
<td>41</td>
</tr>
<tr>
<td>5.14</td>
<td>Comparison of Metrics in Lisbon</td>
<td>42</td>
</tr>
</tbody>
</table>
Nomenclature

\( b \) Current Number of Bikes
\( C \) Cluster
\( r \) Drop Off (Dock) Demand
\( D_{\text{max}} \) Maximum Distance between Stations to be Rebalanced
\( \eta_r \) Drop Off (Dock) Availability
\( \psi_r \) Drop Off (Dock) Frequency
\( e \) Selection of a Path
\( \lambda \) Regularization Parameter
\( y \) Net Flow demand
\( O \) Pick Up (Rent) Demand
\( \eta_O \) Pick Up (Rent) Availability
\( \psi_O \) Pick Up (Rent) Frequency
\( c \) Station Capacity
\( T_D \) Travel Distance
\( \theta \) Number of Bikes to Reposition
\( V_C \) Vehicle Capacity
\( x \) Number of Bikes in a Station
\( \hat{x} \) Target State
\( w \) Number of Bicycles Loaded in the Truck
Acronyms

**BSS** Bike Sharing System.

**GDPR** General Data Protection Regulation.

**KNN** K-Nearest-Neighbors.

**KPI** Key Performance Indicator.

**LDD** Lagrange Dual Decomposition.

**MAE** Mean Absolute Error.

**METAR** Meteorological Terminal Air Report.

**MILP** Mixed Integer linear programming.

**MINLP** Mixed Integer Non Linear Programming.

**MPC** Model Predictive Control.

**MSWK** Meteorology Similarity Weighted K-Nearest-Neighbors.

**POI** Point of Interest.

**QP** Quadratic Programming.

**SEE** Sum of Squared Errors.

**STRL** Spatio-Temporal Reinforcement Learning.

**TSP** Travelling Salesman Problem.
Chapter 1

Introduction

1.1 Motivation

Cycling causes virtually no environmental damage, promotes health through physical activity, takes up little space and is economical both in direct user costs and public infrastructures [2].

With the rapid growth of cities, increase in traffic congestion and consequently concentrated pollution, several measures need to be adopted. An investment in more and better public transportation is a solution, but brings up the “last mile” problem, which represents the cost to the user of traveling from the last public transport station to its final destination.

Today, bicycles address traffic congestion as they form a valid substitution for cars on short trips, have a low carbon footprint, and contribute to the use of public transport by providing an effective last-mile connectivity.

Bike sharing systems allow the inhabitants to use a bicycle without owning one nor having to carry it around. Most systems are composed of stations (docks) placed throughout the city, from which the users can rent a bike and return it in another station. The essence of an efficient bike sharing system relies on having a balanced number of bikes in every station at all times. An unbalanced system would have empty stations, from which the users cannot rent bikes, and full stations, which do not allow docking. Since there is an asymmetric pendular flow of users at different times of the day from residential areas to working areas and vice-versa, the stations tend to get jammed or empty and penalize the quality of the service provided to the user. This bicycle imbalance problem makes it necessary for bikeshare cities to employ costly bike redistribution, which is typically performed by trucks, moving bikes among stations. Furthermore, this problem is aggravated when there is real-time monitoring instead of predicting the user movements, as it takes too much time to reposition after an imbalance has occurred. An example of the repositioning benefits is the case of Citibike, in which only after a few years since the inauguration of the Citibike system, did they start using predicting algorithms which improved the service to the user greatly [3].

Rebalancing is a challenging problem due to its intractability. In this thesis we seek to tackle this problem, by having a first approach to demand prediction and to the control of the system while minimizing the costs to the system operator and providing a good service to the user.

1.2 Industry Parallelism

Even though the main motivation behind this thesis was to improve a bike sharing system, the developed work has a vast number of applications. The airline industry faces the problem of aircraft routing, which
is an analogy to the single vehicle static repositioning. Moreover, our proposed model can be applied to any urban mobility system, including the futuristic concept of an autonomous car sharing system, rerouting the vehicles to answer to the users demand. A more current application to be approached is the airport logistics, in which airport personnel and ground support equipment are in "stations" and must be repositioned to answer to the demand of the airport users.

1.3 Objectives and Contributions

Our first goal is to find a global control algorithm that fulfills cyclists demand with a limited number of bicycles and a fixed infrastructure of stations. Respecting the [General Data Protection Regulation](GDPR) [4], which imposes hard limits on personal data access, namely on digital traces of human movements, our data model is restricted to a minimum: we seldom require samples in time of the number of bicycles on each station, with no knowledge of user trips. As such, we take a data driven approach to estimate the station demand, which takes into account several factors, such as time, location, weather condition, and traffic situations.

We propose a Model Predictive Control dynamic approach to tackle the repositioning problem, which considers the station level demands and outputs a system wide solution.

1.4 Thesis Outline

This dissertation project is structured as follows:

In Chapter 2 we introduce previous work done relevant to our thesis. We start by introducing an overview over Bike Sharing systems and how our work can help improve these. Furthermore, we present previous work on the demand prediction, which is one of the main challenges associated with rebalancing. Finally, we briefly review the existent rebalancing methods.

In Chapter 3 we do an Exploratory Data analysis of the available datasets, namely weather information and the where we present a visual comparison of the variables and their relationship with the demand.

After interpreting the data in the previous chapter, in Chapter 4 we attest the efficiency of some prediction models, which will be used for the proposed rebalancing method using MPC.

In Chapter 5 we compare our model against two benchmark approaches. These approaches are evaluated using a simulation where the movements of customers are generated from real-world datasets.

Finally, Chapter 6 discusses the results of this thesis and provide possible areas of improvement for further work.
Chapter 2

State of the Art

2.1 Bike Sharing Systems

Despite rapid global motorization, worldwide bicycle use has increased over the years [5]. The first bike sharing system was introduced in Amsterdam in 1965 and it was a solution to solve the traffic problems in the city center. Fifty bikes were placed throughout the city and were free to use without any previous registration. Unfortunately, the bikes were stolen or damaged and the project was discontinued.

The second generation bike system was based on a coin-deposit system, where users paid a deposit that was refunded upon return. This generation introduced the concept of docks which is still used in most of the existent bike sharing systems. Unfortunately, that system still presented the same theft problems as no identification was provided by the users.

The third generation tackled the previous problem by introducing IT-based systems that required smart technology for the renting process and the members had to provide an ID or bank card. This generation (present time) is demand responsive and considers multimodal systems, collecting data about the state of the system. With the growth of the bike sharing systems, smart reposition is necessary, and due to the advancements in computing power for data mining, it is possible.

The concept of dockless bikes was introduced in this last generation. Its users have the freedom to leave the bike wherever they want. This is advantageous as the last mile problem is further reduced. Moreover, bikes are equipped with an innovative locking system embedding specific electronics that also enables the possibility to define some restricted parking areas (geo-fencing).

As of the time of the writing of this work, there are over 2500 cities throughout the world which have implemented a bike sharing system, of which only 16% were discontinued [6]. Annual operating costs can reach as high as 1600 € per bike [7], hence, response to the demand must take into account a limited budget.

2.2 Related Work

Due to constant changes in the system, a good prediction algorithm is the key to an efficient rebalancing minimizing the resources used. Our work focuses on the problems of demand prediction and efficient vehicle routing for rebalancing.
2.2.1 Demand Prediction

Early work by Froehlich et al. [8] uses Bayesian Networks to predict the demand per station. However, they only compare it with simple baseline algorithms such as Historic Mean values. In addition, it is also a one factor method, which does not consider the weather features nor seasons. Most literature on prediction proves that, among other features, weather plays an important role in demand prediction.

Yutaka et al. [9] analyze the effects of not only weather (temperature, rain and snow) on cycling decisions, but also other geographical factors like cycling time, slope, cycling facilities (bike lanes) and traffic.

Several previous works have studied several predicting models and compared their accuracy, such as Random Forests, decision trees, gradient boosting machines and linear regression [10, 11, 12, 13]. There is a consensus between Yin et al. [12] and Feng et al. [14], which found Random Forest to work best. Among the benchmark algorithms used are Basic Linear and Ridge Regression, Models with Elastic Net Regularization, Generalized Boosted Model, Principal Component Regression, Support Vector Regression, Inference Trees.

Li et al. [15] proposed a bike-sharing demand prediction framework that introduces a Bipartite Station Clustering algorithm to group individual stations. The whole city demand is predicted based on the Gradient Boosting Regression Tree and later split across clusters based on a Multi-similarity-based Inference model.

Further work is developed by Liu et al. [16] which consider a weighted multi similarity model using K-nearest Neighbors to predict pick up demand according to the most similar days in weather variables. Drop off demand is calculated by designing a transition network between stations and attribute a destination based on the pick up demand.

Liu et al. [17] proposed a hierarchical bike traffic prediction model with the objective of studying a Bike System expansion. They perform a clustering of stations according to distance and POI and predict the user demand using a Random Forest algorithm. The station level demand is then calculated using ridge regression model, which is trained using the historical transition records of stations within the same functional zone. The model is evaluated using baseline algorithms as a comparison, such as Random Forest, K-nearest-neighbor, neural network and gradient Boosting Regressor and performed significantly better in a Root Mean Squared Logarithmic error (RMLSE) metric.

Cluster Based Prediction

There is extensive literature on predicting the movements of users both on a station level basis and on a region level, which is composed of a set of stations close to each other and with similar patterns and urban functions. This clustering of stations reduces computation time significantly [18]. Additionally, these methods present better results in the demand prediction [17, 19], but neglect the cost of the user to move to a station within the region, which may not the closest.

Bao et al. [19] investigated the bike sharing travel patterns and trip purposes by combining smart card data and online point of interests (POIs) using Latent Dirichlet Allocation (LDA). Firstly, clustering analysis was applied to divide bike sharing stations into several different types according to the ratios of each POI category surrounding the stations, using K-means. The results of LDA model showed that the most prevalent travel purpose in New York City is renting bikes for eating, followed by shopping and transferring to other public transit systems. In addition, the result also suggested that people living around bike sharing stations are more likely to transfer to other commuting tools on the morning peak and ride for home after work. Liu et al. [17] presented similar work applied to the expansion of Bike Sharing Systems.

Li et al. [11] generated clusters according to distance and similar patterns. Firstly, regions are gen-
erated, which are small groups of stations close to each other and have similar origin and destination regions. Li et al. [1] claim the rent demand at a region is more stable and regular than at an individual station. Besides, the transition between two regions is more frequent than between a pair of stations. Furthermore, instead of guaranteeing bike and dock availability at each station, there is only the need to guarantee it in the region. Instead of repositioning among regions in the entire system, there is a further grouping of these regions into clusters, where there is only conducted inner-cluster reposition, which further reduces the problem complexity.

These clusters must obey two rules:

- **Inner-Balanced**: The total rent and return demands within the cluster should be almost equal.
- **Inter-Independent**: There are not frequent transitions between stations of different clusters.

Furthermore, Li et al. [1] conclude the dynamics of the system can be reduced to 5 episodes in a day. In each episode, it is assumed the inner-balance and inter-independence properties of clusters do not change.

<table>
<thead>
<tr>
<th>Episode</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morning rush hours</td>
<td>7:00 am – 11:00 am</td>
</tr>
<tr>
<td>Day time</td>
<td>12:00 pm – 16:00 pm</td>
</tr>
<tr>
<td>Evening rush hours</td>
<td>17:00 pm – 22:00 pm</td>
</tr>
<tr>
<td>Travel hours</td>
<td>9:00 am – 17:00 pm</td>
</tr>
<tr>
<td>Evening hours</td>
<td>18:00 pm – 23:00 pm</td>
</tr>
</tbody>
</table>

Figure 2.1: Different episodes over the day [1]

Lin et al. [20] and Zhang et al. [21] applied Convolutional Neural Network (CNN) for large-scale bike-sharing flow prediction. Zhang et al. [21] split the whole city into grids with a pre-defined grid size and calculated the bike-sharing demand for each grid. The demand data, represented using grid maps, was converted into images by defining a color scale. These images were then taken as the input to the CNN model and the correlations among grids were learned through localized filters. Note that this approach is still not applicable for station-level bike-sharing demand prediction because if the grid size is set too large, multiple stations will be covered by the same grid and fail to satisfy the required granularity. Conversely, if the grid size is as small as one station, the huge image matrix with redundant zero elements will increase the computational burden heavily. On the other hand, Lin et al. [20] predicted station-level hourly demand by proposing a novel Graph Convolutional Neural Network with Data-driven Graph Filter. It captures heterogeneous pairwise correlations between stations and also implements the recurrent block from the Long Short-term Memory (LSTM) neural network architecture to capture the temporal dependencies in the bike-sharing demand series.

### 2.2.2 Rebalancing

Rebalancing is a necessary process as a result of bike flows that cluster in certain areas of the city, following a pattern in which residential areas receive the most bikes in the evening whereas commercial and working areas receive the most bikes in the morning of a workday.
User-Based Rebalancing

There is a considerable set of literature regarding vehicle based repositioning, which will be explored in the next section. The vehicle-based approach employs multiple trucks/bike-trailers to achieve balancing by loading or unloading bikes in different stations or areas. This implies traveling costs of trucks, as well as labor costs. Consequently, the truck-based approach can deplete the limited budget rapidly.

In contrast, the user-based approach offers a more economical and flexible way to rebalance the system, by offering users monetary incentives and alternative bike pick-up or drop-off locations. In this way, users are motivated to pick up or return bikes in neighboring regions rather than regions suffering from bike or dock shortage. Singla et al. [22] propose a pricing mechanism that dynamically calculates the incentive values for each neighbor station and offers the corresponding incentive amounts to the users through a mobile application.

Our approach to the problem falls into the category of vehicle-based repositioning.

Vehicle-based Repositioning

Static Repositioning

In this method, the movements of bikes during the repositioning problem are negligible. Static repositioning is ideally done during night time, when the system is off hours and there is little or no bike movements.

It focuses on the problem of finding the optimal routes for a fixed set of vehicles to reposition bikes to achieve the desired configuration of bikes across the base stations. The single capacitated vehicle is a NP-hard algorithm and several propositions were developed to reduce computing time.

Chemla et al. [23] proposed a heuristic method for the single-vehicle static rebalancing problem, where each station could be visited more than once and could be used as a buffer in which bicycles are stored for a later visit. A branch-and-cut algorithm is proposed for solving a relaxation of the problem.

Forma et al. [24] firstly cluster stations according to geographic as well as inventory (of bicycles) considerations. In the second step, the repositioning vehicles are routed through the clusters while tentative inventory decisions are made for each station individually. Finally, the original repositioning problem is solved with the restriction that the vehicles can only do repositioning in stations that belong to neighbor or same clusters, according to the routes determined in the previous step, or between stations of the same cluster. In the first step, the clusters are formed using a specialized saving heuristic. The last two steps are formulated as Mixed Integer Linear Programs.

Rebalancing Bike Sharing Systems: A Multi-source Data Smart Optimization

After predicting the demand per station, Liu at al. [16] first attributes an optimal repositioning value to each station, while maximizing the time it will be balanced.

We propose a modification to the original algorithm, with the following formulation:

$$\begin{align*}
\text{maximize} & \quad T - \lambda \theta^2 \\
\text{subject to} & \quad -V_C \leq \theta \leq V_C \\
& \quad 0 < T < 24 \\
& \quad 0.15 * c < b + \theta + \sum_{t=0}^{i} w(t) < 0.85 * c \\
& \quad \text{where } i = 1, \ldots, 24
\end{align*}$$
where \( T \) is the maximum future horizon the station remains balanced, \( V_C \) is the vehicle capacity, \( b \) is the current number of bicycles on the station at the moment \( t = 0 \), \( c \) is the station capacity, \( \theta \) is the number of bikes to be repositioned and \( w \) is the netflow at time \( t \).

This algorithm differs from the original as it includes the \( \lambda \) parameter in the objective function, which represents the trade-off between the number of bikes to be repositioned and the time slots the station will be balanced. For instance, in case a station requires a high number of bikes for a minimal gain in \( T \), it may be more advantageous for the system to have those bikes distributed between other stations, as such a higher lambda, should be chosen.

Furthermore, if multiple values of \( \theta \) have no effect on \( T \), the parameter \( \lambda \) enforces the choice of the lowest \( \theta \).

After finding the optimal inventory per station, Liu et al. [16] cluster the stations in need of rebalancing using a Constrained K-nearest neighbors algorithm, in which the sum of the repositioning values(\( \theta \)) of the station in the cluster cannot be higher than the vehicle capacity. The choice of the number of \( K \)-centers lies not only on the available truck fleet and budget, but also on a maximum of the total repositioning time, chosen by the system administrator. Afterwards, a [Mixed Integer Non Linear Programming (MINLP)] model with lazy constraints is conducted to optimize the inner cluster rebalancing routes with the vehicle. Due to its’ non linearity, this is a challenging problem to solve.

**Dynamic Repositioning**

Opposite to Static Reposition, a Dynamic Repositioning Model considers the system changes, which affect the result of the repositioning. It is useful to consider demand surges and dips during the day, where a dynamic model can produce stronger results than a static one [18].

Lowalekar [25] extend the work by Ghosh et al. [26] with an online approach where they employ a Lagrangian decomposition problem (decouples the global problem into routing and repositioning slaves and employs a novel DP) and a greedy online anticipatory heuristic to solve large scale problems effectively and efficiently.

Ghosh et al. [18] further improved their previous work from [26] where they also apply a [Lagrangian Dual Decomposition (LDD)] problem and create an abstract formulation of the [Mixed Integer linear programming (MILP)] which speeds up the process by 5000% compared to a global [MILP] and [LDD] problem with a loss profit to the system of 0.3%.

Unlike the static reposition methods, the [Spatio-Temporal Reinforcement Learning (STRL)] model by Li et al. [1] does not generate a sequence of repositions in advance. A reinforcement learning model is used to maximize the long-term reward of a sequence of decisions, in which the reward to be given when transitioning to the next state is the negative customer loss. The problem is formulated in a multi-agent way, such that a new reposition is generated to a trike that has just completed its last reposition, without waiting for the completion of the others. For each cluster of stations, the model uses a deep Q learning network to generate the optimal policy. Each time a trike requires for a new reposition, it has a real-time observation of the current environment. As the action and state space are large, a Spatio-temporal pruning is designed to discard some actions which are known to be bad. The rules to prune the possible bad actions are:

1. Always unload at a region with few bikes and load at a full region.
2. The reposition target region should be chosen from the top-k nearest neighborhoods of the current region; besides, how many bikes to load or unload is generated with a step size.

Our approach of the rebalancing problem is dynamical, in which we consider the dynamic changes of the system each time step through continuous feedback to the controller.
Chapter 3

Exploratory Data Analysis and Preprocessing

In this chapter we cover the data collection process, describing the multiple data sources, and present an exploratory data analysis, where we analyze the data statistically and visually with the aid of histograms and graphic plots. After preprocessing and interpreting the data, we formulate hypotheses and check assumptions required to fit a data model.

For this Exploratory Data Analysis, we will analyze the datasets from 2 Bike Sharing Systems: Citibike, in New York, and GIRA, in Lisbon. Our goal is to fit a data model which can be applied not only these systems, but also any Bike Sharing System.

3.1 Concepts and Notation

Based on previous work by Liu et al. [16], we adopted some of the concepts to measure pick up (rent) and drop off (return) demands:

Pick Up (rent) Demand and Frequency

For a time slot \( t \), the pick-up frequency \( \psi_O \) is the number of bikes that were rented. The pick up availability \( \eta_O \in [0, 1] \) is a normalized value of the amount of time the station had bicycles for the users to rent, during time slot \( t \). The formula for the pick up demand \( O_i \), where the subscript indicates station \( i \), can be seen in Equation 3.1.

\[
O_i(t) = \frac{\psi_O}{\eta_O}
\]  

For instance, in case there were 20 bikes rented in time slot \( t \), but the station was empty half of the time slot, the corresponding pick up demand is:

\[
O(t) = \frac{20}{0.5} = 40
\]

The station availability is important as we would be losing pick up frequency information if the stations were empty or full. In order to not penalize the prediction algorithm due to poor rebalancing during the data collection period, the author is considering that the pick up/drop off activity is the same for the whole time slot. This is why in the example above, the demand considered during the station unavailable time is the same as observed during the rest of the time slot.
Drop Off (return) Demand and Frequency:

Similarly, the drop off frequency($\psi_r$) is the number of bikes docked during $t$, where $\eta_r$ is the normalized time for which the station was not full. The formula for the drop off demand is:

$$r_i(t) = \frac{\psi_r}{\eta_r}$$  \hspace{1cm} (3.2)

Netflow Demand

The netflow demand at station $i$, or total demand in a station corresponds to the total number of bikes being docked minus the number of bikes rented.

$$y_i(t) = r_i(t) - O_i(t)$$  \hspace{1cm} (3.3)

Mean Absolute Error (MAE)

MAE is the metric used to measure the performance of various algorithms. Its equation is:

$$MAE = \frac{\sum_{m=1}^{M} |\hat{d}_m(t) - d_m(t)|}{M}$$  \hspace{1cm} (3.4)

where $\hat{d}$ is an estimated value, $d$ is the ground truth. This could be applied to calculate the average error of the pick up demand $\hat{O}$ prediction. For a certain time slot, the MAE would be the sum of the absolute difference between the predicted demand $\hat{O}$ and the historical demand $O$. In this case, $M$ is the total number of stations.

3.2 Meteorology Data Extraction

Most open data weather services provide a real-time feed of weather reports and historical data of up to a few months free of charge. For our problem, a large dataset makes our data less sensible to extraordinary episodes, ie. a football game where the demand is unusually high.

With the proximity of the airports to the city center, the airport’s weather reports are a source of free historical meteorology data. The [METARs](https://en.wikipedia.org/wiki/Meteorological_Terminal_Air_Report), short for Meteorological Terminal Air Report[27], is the most common format in the world for the transmission of observational weather data. They are issued every 30 minutes and are highly standardized for the aviation industry.

There are many variations of METAR reports to communicate different weather phenomena and cloud coverage. An example of a METAR report is as follows[28]:

KJAX 020256Z 02003KT 10SM TSRA SCT100 BKN130 18/17 A2996

where:

- KJAX : 4 letter ICAO code to identify the airport
- 020256Z : the first 2 numbers correspond the day of the month and the last 4 digits indicate the time zone.
- 02003KT : the first 3 digits represent the heading that the wind is blowing from. The last 2 numbers represent the wind velocity, while the last 2 letters give the unit
- 10SM : Visibility of 10 miles
• TSRA : Thunderstorms and Moderate Rain in the area
• SCT100 : Scattered clouds at 10000 feet
• BKN130 : Cloud coverage, Broken clouds at 13000 feet
• 18/17 : Temperature (18°C) and Dew Point (17°C)
• A2996 : Standard Altimeter setting in inches of mercury

The METAR dataset was parsed and the units were converted to SI, extracting the following variables:

• Wind Speed and Direction
• Clouds and Sky Status
• Pressure
• Weather Phenomena
• Visibility
• Precipitation
• Temperature and Dew Point, which is used to calculate the relative humidity

Due to the proximity of LaGuardia (NY) and Portela (Lisbon) airports to the city center, which are less than 9km and 7km far from the city center, respectively, we will consider the airport weather reports to be similar to what is observed in the city center.

The METARs are provided in UTC time zone, thus it was converted to the time zones of each city.

### 3.3 New York

The Bike Sharing System from New York is called Citibike. It started in 2013 with 332 stations and around 6000 bikes. As of July 2017, there were 130,000 subscribers. Currently, there are over 750 stations. The annual pass costs 169$ year and allows a free ride for the first 45 minutes, with an extra charge of 2.5$ per additional 15 minutes[29].

It is a dock-based system although the network was expanded to Bronx where the bikes are dockless. However, these bikes were not considered to our work as no data is provided.

The dataset we used for the Exploratory Data Analysis is comprised between 2017/05 and 2019/06. Citibike provides monthly datasets on their website with all the trips performed by the users, with a duration of over 1 minute[30]. The information included is:

• Start time and end time
• Origin and Destination Stations IDs and names
• Station Location
• User Birth Year
• User Gender
• User Type (Subscriber/ Customer)
A subscriber is a regular user while a customer is an occasional/tourist rider.

Additionally, Citibike provides a real-time ARCGIS feed with information regarding each station:

- Station ID and name
- Station Capacity
- Number of Bikes in the station
- Station Location
- Update date

There were some inconsistencies in the datasets provided. Some locations were invalid, not even existing at the Citibike website. Additionally, there were multiple IDs for the same station in the different datasets. These stations were ignored and not considered for our work.

3.4 Lisbon

The Bike Sharing System in Lisbon is called GIRA. It is a dock-based system and there are both electric and non-electric/classic bikes. As of October 2019, there are 81 stations, 18000 active annual passes and around 600 bikes. The system is still in development and the last phase of the program predicts a total of 1410 bikes and 140 stations. Currently the prices for a regular use cost 25 €/month and, similarly to Citibike, the first 45 minutes are free, but with an extra charge of 1 € per additional hour [31].

Gira system has an open data platform1 where they provide a real-time ARCGIS feed of station information with a sample time of 5 minutes:

- Station ID
- Location
- Status of the station (active/inactive)
- Number of docks (capacity)
- Number of bicycles

Due to the European General Data Protection Regulation [4], Lisbon open data platform is not allowed to share individual trip records, having only available the number of bikes on each station. Hence, we are lacking the trip records with origin and destination stations of each trip, which limits our choice of demand prediction algorithms as we cannot build a network of stations according to past trips [16, 1].

Additionally, the dataset provided has an update frequency of 5 minutes, which leads to some data loss as multiple rents and returns can happen in a time slot and we only have information on the total balance. As such, we can not extract the precise pick up and drop off frequencies. Instead, we make an approximation that during the 5 minutes time slot, we assume there is only a kind of event happening: either pick up events or drop off events. This way we extract the rent and return demands by analyzing the difference in the number of bikes.

After extracting the data, some pre-processing was required:

- There are stations which have invalid locations or a capacity of 0. These stations were ignored.

1https://emel.city-platform.com/opendata/
• There is missing data due to malfunction of the feed. In case it corresponds to a time frame of a few minutes, the data is ignored and the demand during that day is unknown. If it corresponds to a time frame of a few minutes, it is considered the inventory at the stations remains the same, with a demand of 0.

The dataset used for the GIRA system is comprised between 2018/03 and 2018/12. Opposite to Citibike, GIRA is still in development. The user base keeps increasing, as well as the bike fleet. However, due to problems with suppliers in 2019, the number of bikes in the system decreased due to lack of maintenance, and the real-time feed stopped casting due to a problem with a contractor. The total number of stations during this time period is 54, with an estimated fleet of 300 bikes.

3.5 Data Visualization

Our goal is to predict the rent and return demands. As presented in Chapter 2, there is extensive literature on demand prediction, which considered multiple factors such as weather, social and geographical conditions.

After gathering all the data, we take into account the available common data in both BSS, as we want to design a model that fits both systems.

We create a new variable, the Weather Conditions, which conveys information about weather phenomena clustered into four categories, according to their suitability for outdoor bicycling: \( U_1 = \{ \text{Cloudy, Sunny} \} \), \( U_2 = \{ \text{fog, mist, haze} \} \), \( U_3 = \{ \text{snow, rain, light snow, light rain} \} \), \( U_4 = \{ \text{heavy snow, heavy rain} \} \). The values corresponding to each category are: 0.25, 0.5, 0.75, 1 respectively.

(a) New York
(b) Lisbon

Figure 3.1: Median Bike Rents for each hour of the day

Just like Citibike system has around 30 times more bikes than GIRA, the average demand in New York is observed to be 25 times higher than in Lisbon, as observed in Figure 3.1a and 3.1b. The dynamics of the system are similar. New York has two daily peaks of demand, in the morning and in the afternoon. Lisbon additionally has a high peak at 13h during working days. As expected, the weekend demand is generally lower than during weekdays. The demand from 2 til 6 is almost null since the system is closed during that time, as seen in 3.1b. The movements observed correspond to repositions done by the system operator.

In the next set of Figures, the relationship between the rent demand and the weather variables is visualized with histograms. Each data point represents the hourly average bike rental for a specific
weather variable. For example, the data point (30, 3200) in Figure 3.6, the value 3200 corresponds to the average hourly bike rentals in the hours in which the temperature was observed to be 30°C.

In Figure 3.2, the relationship between humidity and the pick up demand is not well defined. In New York there is a lower demand with the increase in relative humidity and their relation is approximately linear. It can be hypothesized that added temperature feel due to high humidity works as a deterrent to riding a bicycle outdoors. However, this is not confirmed in the case of Lisbon, where the humidity appears to have no correlation with the demand, possibly due to the existence of electric bikes.

In a low visibility day, there might be a presence of fog or intense rain. As such, the higher the visibility, the better the conditions to ride. There seems to be a small relationship between the demand and visibility as seen in Figure 3.3. With the exception of a few outliers, the tendency is for the demand to increase linearly with visibility.
In the case of New York, in Figure 3.4a, the demand drops greatly with higher wind speeds than 50km/h. According to the Beaufort scale\(^2\), wind speeds of up to 50 km/h are considered a strong breeze holding the number 6 in the Beaufort scale, while winds of 60km/h (7 in the Beaufort scale) are considered intense winds. In Lisbon, there were no observations of intense winds, as such, the influence of the wind speeds is inconclusive as the demand does not seem to be affected by the wind.

---

\(^2\)https://www.spc.noaa.gov/faq/tornado/beaufort.html

---

The weather conditions present a clear relationship with the demand. An environment with thunderstorms and snow (weather conditions = 1) is not ideal for cycling, and users are demotivated to do so, whereas a sunny and clear skies day (weather conditions = 0.25) is more appealing for the users to go cycling. As such, the demand decreases with extreme weather conditions.
Analyzing Figure 3.6a, there is a clear linear relationship between temperature and demand in New York, in which the bike usage increases with temperature, since with colder it is, more people prefer to use other means of transport. In Lisbon, there is an increase in the demand with temperature until 20°C, where it reaches a peak. For higher temperatures than 20°C during weekdays, the demand is not affected, but for weekends, it decreases with temperature. This should make sense as users may also be discouraged to bike when it is too hot outside.

### 3.6 Discussion

Through the analysis of the relationship between the weather variables and demand, it is seen the system in development does not display the same patterns as a matured system like Citibike. This is due to the fact that the demand is increasing over time due to a rise in the user base. This eliminates the effects of seasons as a larger user base during less cycling prone conditions might present a higher demand than a smaller one in good riding conditions.

To conclude, a more defined weather dependent pattern of user movements should be more visible when the system reaches maturity.
Chapter 4

MPC for Rebalancing

Taking into account the data analysis performed previously, in this chapter we detail the implementation of Model Predictive Control (MPC) to solve the problem of dynamic rebalancing. We also compare multiple demand prediction algorithms, to attest their validity to use in a real world situation.

4.1 Data Model

As the available data of our system is the station inventory, our system state can be represented by the variable $x$:

$$x_n = \text{#number of bikes on station } n$$

The system is modeled linearly, where the difference in the number of bikes in the next time slot $t$ is the inflow of bikes, minus the outflow:

$$x_n(t + 1) = x_n(t) + F_{in}(t) - F_{out}(t) \quad (4.1)$$

where $F_{in}$ corresponds to the inflow and $F_{out}$ is the outflow of bikes.

The inflow and outflow functions account for the movements of the users and repositioned bikes. We can control the system through repositioning in order to set $x$ to an optimal value throughout the day.

Lacking the dataset corresponding to the user movements and violates the GDPR privacy law, we cannot build a station network considering the system changes. Instead there is only information on each individual station. We propose an implementation of a MPC controller, which provides a solution by attending to the needs of the system.

Our model must take into account predictions of the user movements. The MPC solves this problem very well as it considers the user movements as disturbances in the system.

The model must consider the effects of repositioning in the long term, for which the MPC considers the full sequence of inputs over a future horizon, where the inputs in our system are the repositions.

Furthermore, we want our model to be subject to constraints on the repositioning vehicle and station capacity. A MPC is very effective at dealing with constrained control problems.

Since our bike repositioning system is modeled with linear dynamics, it simplifies the MPC control problem to a series of direct matrix algebra calculations that are fast and robust. This way, a MPC fits our problem very well.
4.2 MPC Background

Model Predictive Control is used in several industries and it is one of the most successful modern control techniques [32].

Assuming a linear model of the plant, the discrete-time linear dynamics are [32]:

\[ x(k + 1) = Ax(k) + Bu(k) + Cd(k) \]

where \( x(k) \in \mathbb{R}^q \) is the state of the system at time step \( k \), \( u \in \mathbb{R}^r \) is the vector of manipulated variables and \( d \in \mathbb{R}^s \) is a vector of measurable disturbances. \( A, B \) and \( C \) are matrices of proper dimensions.

The performance measure (to be minimized) is given by:

\[
J(x(0), u_{[0,T-1]} = \sum_{t=1}^{T} x(t)^T Q x(t) + \sum_{t=0}^{T-1} u(t)^T R u(t) \tag{4.2}
\]

where \( T \) is a finite future horizon, \( Q \) is a positive semidefinite \( N \times N \) matrix, where \( N \) is the size of \( x \), and \( R \) is a symmetric, positive definite \( M \times M \) matrix, where \( M \) is the size of \( u \). Matrix \( R \) cannot be positive semi definite as there would be uncontrolled nodes.

The MPC considers the full sequence of input steps required to move the system from its current state to a future target, while minimizing a user-chosen cost. The control system then applies the first inputs and reads the updated state provided by feedback. Ignoring the previous calculated future inputs, the MPC recalculates the optimal input and future state, and the process repeats [33].

MPC is also called Receding Horizon Control as it calculates the optimal control sequence for a particular finite future horizon \( T \), which moves forward by the time elapsed since the last control. Theoretically, if an infinite horizon is considered, the MPC law corresponds to the solution of an optimal linear-quadratic (LQ) control problem. When a horizon is implemented, there is a certain discrepancy between both solutions, which tends to increase as the horizon is reduced. This way, it is concluded the MPC does not output the optimal solution [32].

Additionally, the state and input variables can also be constrained:

\[
\underline{u} \leq u_i(k) \leq \bar{u} \text{ for all } i=1,2,...,M \text{ and } k = 0,1,2,...,T \\
\underline{x} \leq u_i(k) \leq \bar{x} \text{ for all } i=1,2,...,M \text{ and } k = 0,1,2,...,T
\]

where \( \underline{u} \) and \( \bar{u} \) are the minimum and maximum values of the input respectively. Analogously, the same concept is applied to \( x \). This MPC problem described here is known to be robustly asymptotically stable with respect to bounded state and measurement noise [32].

In Figure 4.1, the dotted line and circle mean that the disturbance input \( d(k) \) does not exist, but serves the purpose of creating a state to the plant model for which the controller is designed. The observer block is responsible for the set point tracking and estimates the disturbance.
4.3 Problem Statement and Notation

As mentioned at the beginning of the chapter, the state of the system is represented by the state variable \( x_n(t) \), which corresponds to the number of bikes docked at station \( n \) at time slot \( t \).

Additionally, we are considering a virtual station with index 0 (will be referred to as cloud), to account for the trips in progress at time slot \( t \). Whenever a bike is rented from a certain physical station, it is placed in the cloud, for the duration of the trip. When it is returned to a physical station, it is removed from the cloud. The existence of the cloud forces the number of bikes in the system to be constant.

Accordingly, \( x(t) \) has length \( N+1 \), where \( N \) is the total number of physical stations, such that the index \( n \) can take the values: \( \{0, 1, \ldots, N\} \).

A bike station network is constructed where each node corresponds to a station, and the arcs between stations correspond to the repositioning of bikes between a pair of stations. Hence, we build a connected graph \( G(t) = (\mathcal{N}, \mathcal{A}(t)) \), where \( \mathcal{N} \) is the set of nodes and \( \mathcal{A} \) is the set of arcs. Each node with a positive node number represents a station, in which the node number 0 represents the virtual station. An example of the graph can be seen in Figure 4.2 for a system with small dimensions.

It is not a directed graph as it does not make sense to have repositions between stations that are too far. Therefore, there are only arcs between stations which have a travel time lower than a user chosen value. The virtual station is connected to all the stations as any bike in use will go through it.

In addition, since the set of arcs is dependent on driving time, which changes throughout the day, the graph will also be time-dependent. For instance, higher traffic will increase travel time and thus, decrease the number of arcs.
Figure 4.2: Example of a connected geographical graph for a system with 4 stations, represented by the nodes. Station 0 is virtual and has the bikes which are currently in use by the user. The arcs define the number of bike repositions $u$. There are arcs between stations 1, 4; 1, 2; 2, 3; and the virtual station 0 is connected to all stations.

The dynamics of the system can be described by the following discrete-time equation:

$$x_n(t+1) = x_n(t) + \sum_{m \rightarrow n} u_{m \rightarrow n}(t) - \sum_{n \rightarrow p} u_{n \rightarrow p}(t) + v_{0 \rightarrow n}(t) - v_{n \rightarrow 0}(t)$$  \quad (4.3)$$

where $u$ is the control variable, which represents the bicycles to be moved by the bike sharing system operator; and $v$ is the disturbance variable, which represents the user movements.

The cost function from Equation 4.2 is adapted to our problem, in which we minimize the "distance" to the target state $\bar{x}$ while minimizing the control movements.

The MPC problem is formulated as follows:

$$\begin{align*}
\text{minimize} & \quad \sum_{t=1}^{T} (x(t) - \bar{x}(t))^T Q(x(t) - \bar{x}(t)) + \sum_{t=0}^{T-1} u(t)^T R u(t) \\
\text{subject to} & \quad x(t+1) = x(t) + Au(t) + Bv(t), \quad (x_0 \text{ given}) \quad (4.4a) \\
& \quad 0 \leq u(t) \leq V_C, \quad t = 0, \ldots, T-1 \quad (4.4b) \\
& \quad \underline{x}(t) \leq x(t) \leq \bar{x}(t), \quad t = 1, \ldots, T \quad (4.4c) \\
& \quad x(t) + Bu^+(t) \leq c, \quad t = 0, \ldots, T-1 \quad (4.4d) \\
& \quad x(t) - Bu^-(t) \geq 0, \quad t = 0, \ldots, T-1 \quad (4.4e)
\end{align*}$$

where $c$ is the stations capacity, $V_C$ is the vehicle capacity, $\bar{x}$ is the target state, $\underline{x}$ and $\bar{x}$ define a minimum quality of service in which the stations should not have less bikes than the defined lower limit, nor more bikes than the defined upper limit; $u^-(t)$ and $u^+(t)$ represent the minimum and maximum values of the control for each station, respectively.

Matrix $Q$ defines the distance to target state penalty. Matrix $R$ defines the control movements penalty. Matrix $A$ selects each arc from $u_{m \rightarrow n}(t)$ and adds the value to each station. Matrix $B$ selects the user movements from $v(t)$ for each station where all the bikes coming from the cloud to a certain station $i$ count as positive flow, and the bikes leaving station $i$ to the cloud count as negative flow.
Constraint 4.4c limits the control movements \( u \) to the capacity of the repositioning truck, which cannot be negative either. Constraint 4.4d limits the state variable \( x \) and represents the minimum service quality that we want to impose on the system. This constraint forces a minimum number and maximum of bikes in a station.

The MPC controller is admitting the rebalancing process happens immediately at the beginning of the time slot. However, there is a delay until repositions take effect and if there are multiple trucks assigned to a station, the trucks arrive at different times. Constraints 4.4e and 4.4f impose that the first trucks to arrive do not overflow (tries to insert bikes on a full station) the stations, or underflow (removes bikes from an empty station).

Our MPC problem is solved efficiently since it is a convex quadratic programming problem (QP). Hence, if feasible, it will have an optimal global solution. The proof of convexity is verified in Appendix A.

We are doing a relaxation of the problem since the number of bikes is an integer number. The output of the model will be rounded to the nearest integer. Alternatively, forcing the variables to be an integer number would turn the problem into a MILP problem, which is not convex, making it more challenging to solve and the solution is not guaranteed to be globally optimal.

### 4.4 User movements Prediction

The user movements represent the number of rents and returns at each station. Vector \( v(t) \) (Equation 4.5) has length \( 2N \), and it is a concatenation of the vectors \( v_{n \rightarrow 0}(t) \) and \( v_{0 \rightarrow n}(t) \) from Equation 4.3. Vector \( v_{n \rightarrow 0}(t) \) represents all the bikes leaving station \( n \) to the cloud which has length. Similarly, \( v_{0 \rightarrow n}(t) \) represents that arrive at station \( n \) and leave the cloud.

\[
v(t) = \{ v_{1 \rightarrow 0}(t), ..., v_{N \rightarrow 0}(t), v_{0 \rightarrow 1}(t), ..., v_{0 \rightarrow N}(t) \}
\]

(4.5)

In this estimation process, we compare several methods of obtaining \( \hat{v}(t) \): firstly using histograms to calculate historical mean values, from where we extract 3 different values, secondly a Weighted K-nearest-neighbors method for the most similar days inventory-wise. Thirdly, we use the MSWK algorithm by Liu et al. [16] which considers weather features to the demand prediction.

#### Strategy 1: Histograms

Historical data was used to compute monthly histograms. In total 3 different estimated values \( \hat{v} \) were extracted from these histograms.

Firstly, we calculated how many bikes were docked, or rented at each station for a given hour and the same day of every month. This way, if the target day that we want to estimate the user flows is the 1st of January, at 15h, we extract the observed flows for the 1st day of every month at 15h. When creating a histogram, and normalizing it by the number of data points, we obtain a “probability” (“\( p \)” ) for the number of bikes leaving-entering the station.

#### Average Value Method (AV)

In the AV method, the average value of user movements for the same day of the month is extracted from the histograms. From Figure 4.3a, the average value is approximately 7, and that is the value assigned to \( \hat{v}_{7 \rightarrow 0}(t) \)

#### Most Probable Value (MP)

In an alternative method, we can extract the most probable demand for the same day of every month. From Figure 4.3a, the \( \hat{v}_{7 \rightarrow 0}(t) \) would be 3 or 4, which are the most probable number of bikes to be rented. As a tie-breaker, 4 is chosen as it is closer to the mean value 7.
Monthly hourly average (MHA)  Alternatively, it is calculated the average hourly demand for a whole month. One example is shown in Figures 4.3b and 4.3d which show the average user movements for each hour of the day during the month of April. The estimated user movements $\hat{v}(t_{target})$ for our target day is extracted according to the target hour $t_{target}$.

![Figure 4.3](image)

Figure 4.3: Monthly histograms of user movements for the Citibike system, from where data will be extracted for the Average Value Method (AV), Most Probable Value (MP) and Monthly Hourly Average (MHA) methods

Strategy 2: Inventory K Nearest Neighbours (IKNN)

The most similar days are the ones with the most similar occupation rate ($\text{occupied docks}/\text{total docks}$) overall. The following function $M$ gives the similarity value:

$$M(x(0), x(p)) = \sum_{n=1}^{N} \left( \frac{x_n(0) - x_n(p)}{c_n} \right)^2$$

where $c_n$ is the station capacity and The vector $\hat{v}(t)$ is calculated as a weighted arithmetic mean of historical examples, where the most similar days will have a higher weight. It is given by:

$$v_{0\to n}(t) = \frac{\sum_{p=1}^{K} M(x(0), x(p))v_{0\to p}(t)}{\sum_{p=1}^{K} M(x(0), x(p))}$$
Similarly for the drop-off demand:

\[ v_{n\rightarrow 0}(t) = \frac{\sum_{p=1}^{K} M(x(0), x(p)) v_{p\rightarrow 0}(t)}{\sum_{p=1}^{K} M(x(0), x(p))} \]

**Strategy 3: Meteorology Similarity Weighted K-Nearest-Neighbors (MSWK)**

The previous strategies are one-factor historical average methods since they only consider historical data and no other features. Upon analysis of the state of the art and considering the available data, the prediction algorithm by Liu et al. [16] is applicable with a few modifications. Since in the GIRA system we do not have a dataset with individual trip records available like in the Citibike system, we are not able to implement the full algorithm. As such, we use the MSWK algorithm to predict not only the pick up demand, but also the drop off demand. The Inter State Transition Bike [16] model constructs a station network and predicts the destination of a bike whenever there is a pick up event, generating the drop off demand. This way, the drop off demand is coherent with the pick up demand as the total flow of the system is null. To deal with this problem, a formulation is presented in Equation 4.7.

First, the temperature \( T_e \) and the weather conditions \( W \) are extracted and it is calculated its similarity based on a Gaussian Kernel function, shown in Equation 4.6a and 4.6b.

Additionally, a 3D Gaussian kernel function is chosen to calculate the similarity of the humidity \( H \), wind speed \( S \) and visibility \( V \) between \( p \) and \( q \) days, as seen in Equation 4.6c.

The similarity function is the linear combination of the above weather similarities, given by:

\[
M(D_p(t), D_q(t), a) = \delta_w(D_p, D_q) \sum_{i=1}^{3} a_i \lambda_i
\]

where \( a \) is the vector of the weights of the different similarities.

The pick up \( O \) and drop off \( r \) and drop off is then calculated through a Similarity Weighed K-Nearest-Neighbors algorithm:

\[
O(t) = \frac{\sum_{p=1}^{K} M(D_p(t), D_q(t), a) \times O(D_p(t))}{\sum_{p=1}^{K} M(D_p(t), D_q(t), a)}
\]

\[
r(t) = \frac{\sum_{p=1}^{K} M(D_p(t), D_q(t), a) \times r(D_p(t))}{\sum_{p=1}^{K} M(D_p(t), D_q(t), a)}
\]

The weight of different factors is trained to improve the accuracy of bike pick-up demand prediction, minimizing the mean absolute error of estimated value \( \hat{O} \).

\[
\min_a \frac{1}{N} \sum_{i=1}^{N} \left| \hat{O}(D_q(t); a) - O(D_q(t)) \right|
\]
Performance Comparison

The $\text{MAE}$ (Equation 3.4) of the demand prediction was measured for the testing set comprised of the month of January 2019 for both Citibike and GIRA systems. The tested methods used were:

- $\text{AV}$: Average value, for the same day of every month
- $\text{MP}$: Most probable for the same day of every month
- $\text{MHA}$: Monthly Hourly Average
- $\text{IKNN}$: Inventory Similarity Weighted K Nearest Neighbors
- $\text{MSWK}$: Meteorology Similarity Weighted KNN

The $\text{MAE}$ for each method can be seen in Figure 4.4.

![Figure 4.4: MAE for the prediction of pick up demand and drop off demand. The MSWK has the lowest MAE, as expected since it considers weather features, opposite to the other methods which only consider the historical demand.](image)

In New York, the $\text{MAE}$ is generally higher in all the prediction methods compared to Lisbon, as the demand is higher and the stations have more capacity. Hence, in New York, there is also has a bigger margin for error than in Lisbon, as a small difference in bicycles can quickly jam or empty the stations.

Regularizing the estimated user movements

For the MPC to have at least one feasible solution, we design an algorithm which calculates $\nu$ while taking into account the results produced by the prediction algorithms. For this, we consider the reposition non-existent ($\alpha = 0$) and the algorithm finds the feasible solution which with the user movements closer to the estimated.

Furthermore, since the inflow and outflow are predicted individually for each station, they might be incoherent and the total flow in the system might be unbalanced. Using this algorithm before running the MPC problem, vector $\nu$ is forced to be coherent.
\[
\text{minimize } \sum_{t=0}^{T-1} ||v(t) - \hat{v}(t)||^2
\]

subject to
\[
x(t + 1) = x(t) + Bv(t), t = 0, ..., T - 1 \text{ (x0 given)}
\]
\[
x \leq x(t) \leq \mathcal{T}, t = 1, ..., T
\]
\[
v(t) \geq 0, t = 0, ..., T - 1
\]

where \(\hat{v}(t)\) is the output of the several algorithms from the previous strategies.

\textbf{Matrix B}

As matrix B selects users movements \((v)\) to each station, it must have dimensions of \(N\times2N\), which multiplied by vector \(v\) of dimensions \(2N\times1\), outputs a vector of size \(N\), which corresponds to the "disturbance" at each node, the user movements at each station. The bikes leaving the station create a negative flow on the station, whereas the bikes entering create a positive flow. This way, matrix B will be composed of \(-1, 1\) for the selection of \(v_{n>0}\) and \(v_{0>n}\) respectively, and 0 for user movements that are not related to the station \(n\). Analyzing a simple example in a 2-station system, we have:

\[
Bv(t) = \begin{bmatrix}
1 & 1 & -1 & -1 \\
-1 & 0 & 1 & 0 \\
0 & -1 & 0 & 1
\end{bmatrix}
\times
\begin{bmatrix}
v_{1>0} \\
v_{2>0} \\
v_{0>1} \\
v_{0>2}
\end{bmatrix}
= \begin{bmatrix}
v_{1>0} + v_{2>0} - v_{0>1} - v_{0>2} \\
v_{0>1} - v_{1>0} \\
v_{0>2} - v_{2>0}
\end{bmatrix}
\]

\textbf{4.5 Optimizing User Service}

Analyzing the objective function from 4.4b:

\[
\sum_{t=1}^{T}(x(t) - x_{des}(t))^TQ(x(t) - x_{des}(t)) + \sum_{t=0}^{T-1}u(t)^TRu(t)
\]

In the reposition process, we have two main problems: we want to minimize the customer loss, providing the best service, while at the same time do the least repositions possible. Hence, the objective function is divided into two components: the optimization according to the user service and the cost of repositioning.

Regarding the optimization to the user, we are minimizing the "distance" between the present an estimated future states to the target state. This is done recurring to a norm \(Q\), which can also be represented by \(||x - \hat{x}||_Q\). For simplicity and respecting the fact that \(Q\) is a semi definite positive matrix, we assigned it to be an identity matrix of size \(N\), which makes this function a \(l_2\) norm and corresponds to a Sum of Squared Errors (SEE) function.

\textbf{4.6 Minimizing repositions}

Ideally, we want the system to be as close to the target state \(\hat{x}(t)\) as possible, in order to provide the best service to the user. To do so, a lot of controls \(u(t)\) have to be generated and this implies high repositioning costs for the system administrator. As such, we also need to minimize the number of non-zero controls.
Firstly, it does not make sense economically for a truck to travel a long distance for a reposition, unless the station is in a remote place. This way, we want to hard-limit the travel time of trucks by 15 minutes. We cannot limit the control links by distance, since it does not account for traffic changes throughout the day.

For the traffic prediction, we are using Google Distance API [34], which gives us a distance and driving time for a certain time of the day. Instead of calculating the driving time for all pairs of stations, we prune beforehand the ones which are too far. This is done by calculating the "Manhattan distance" (\(l_1\) norm) between 2 stations to approximate to New York streets and then query the driving time from Google Distance on all the pairs which are no farther than a parameter \(D_{\text{max}}\), which is explained in Section 4.7.

Matrix R will be responsible for penalizing the control variables due to long distance between stations. It is a diagonal matrix, where each position of its diagonal is associated with a control \(u\) and represents the driving time between stations.

Each pair of stations which are closer than \(D_{\text{max}}\), have an arc added in the graph \(\mathcal{G}\), which means this pair of stations is concatenated to the vector \(u\). This vector will have a total size of the number of arcs, as well as the driving time queried from Google Distance API is added to the correspondent position in the diagonal of matrix R.

### 4.7 Graph size and frequency of control

The MPC computes a control every time slot \(t\). The reposition process of all the trucks must be completed before the next reposition is generated, in order to keep the system coherent, instead of generating actions which are not considering the unfinished ones from previous time slot.

The total reposition time is given by:

\[
t_r = t_{i \rightarrow 1} + t_{\text{load}} + t_{1 \rightarrow 2} + t_{\text{unload}}
\]

where \(t_r\) is the total reposition time, \(t_{i \rightarrow 1}\) is the driving time from the initial position of the trucks to the first station, \(t_{\text{load}}\) is the loading time of the bikes onto the truck, \(t_{1 \rightarrow 2}\) is the driving time to the second station and \(t_{\text{unload}}\) is the unloading time of the bikes from the truck.

The variable \(t_{1 \rightarrow 2}\) depends on the graph size and on the time of the day. If the reposition is done during rush hours, the driving time will increase due to traffic. As a result, the frequency of control of the MPC is dependent on the parameter \(D_{\text{max}}\), which should be time-variant in order to account for traffic.

A high \(D_{\text{max}}\) will increase the graph size, which will allow repositions on stations which are farther away, than if \(D_{\text{max}}\) is low. This way, the time slot chosen for the MPC should also be larger, in order for all the trucks to finish at the same time.

A small \(D_{\text{max}}\) might make the graph unconnected, creating islands of stations, which cannot reposition between them. Hence, if users have high flows between these islands, the MPC model will be unfeasible as it cannot perform controls.

### 4.8 Target State

The choice of optimal inventory values is crucial to maximize the MPC’s capabilities. With the different patterns of user movements over the day, the system profits from different inventory values at different times. This way, it is considered that \(\dot{x}(t)\) is time-variant (Equation 4.4a).
A key question is how the target state for each station is computed. In most of the studies, this quantity is fixed, generally at half the station capacity, as in Rainer-Harbach et al. [35] and Raidl et al. [36]. Schuijbroek et al. [37] compute a lower and upper bound on the service level requirement of each station by using a queuing system and Raviv et al. [38] compute a measure of dissatisfaction for different replenishment periods, given an initial inventory, the station size and stochastic demand patterns.

Another option to determine $\bar{x}(t)$ is to divide the day into different episodes, admitting that the target value in each episode will be constant. The episodes can be based on the observed episodes by Li et al. [1], seen in Figure 2.1. Usual empty stations during rush hour should have a higher bike inventory before rush hour; analogously, a lower inventory before rush-hour for jammed stations. A station which is neither usually jammed nor empty, should keep having a target inventory at mid-capacity.

For simplicity, in this work the target value considered was half of the capacity of the stations for all the time slots. The tuning of target inventory value should be further explored in future work.
Chapter 5

Results

While the output of the demand prediction models can be compared with real data to evaluate its’ accuracy, reposition algorithms need a simulated environment where they can be compared. A simulator allows us to see how the dynamic system responds to the rebalancing changes. The efficiency of different rebalancing methods can be compared using different metrics, defined by the parameters outputted by the simulator.

5.1 Simulator

The analyzed simulators were SUMO (Simulation of Urban Mobility)\(^\text{[39]}\) and PEBISS\(^\text{[40]}\).

SUMO is a road traffic simulator, but at the moment of the writing of this work, there is no implementation of bikes. The main advantage of this simulator is the ability to account for multimodal traffic (for vehicles, public transport and pedestrians).

PEBISS\(^\text{[40]}\) was designed specifically for bike sharing systems. It has several features needed to test our rebalancing methods, such as several metrics as well as statistical data of the stations and users. It also includes several other features, like E-bikes and Smart Phone Application to simulate a real environment.

To avoid unnecessary complexity and building a bike sharing system from the ground up, we adopted the latter simulator, which has all the features needed and changes in the rebalancing methods are easily implemented in Matlab.

5.2 Outline

PEBISS was designed using Barcelona data, thus, we will be comparing the results from Barcelona with New York. Lisbon will be presented as a test case in the end, since the demand is not well defined and the trip generation is hard to predict, rendering the MPC ineffective.

As a first approach to the simulator, the metrics will be compared with the repositioning methods available out of the box in PEBISS simulator, in order to analyze which metrics give us more information on repositioning.

With more insight on the important metrics, we will implement our MPC model and train it while trying to balance the customer loss and the repositioning costs.

To conclude, we will present final comments on the performance of our method.
5.3 Metrics

PEBISS calculates several metrics during run time. These are calculated hourly and are divided into
costs to the user and agency. Their formulas are as following:

- **Agency**: costs to the system administrator in order to have the system operational
  - Reposition: Cost of the reposition. It is the only dynamic cost to the operator.
    \[ R_C = C_t t_r \]  
    where \( C_t \) is the cost of reposition per hour per team and \( t_r \) is the total time invested in a hour
    in reposition

- **Customer Loss**: the cost perceived by the user due to time consumption to use the system
  - Bad service Penalty: in case the users find an empty/full station at origin/destination and have
    to divert its original course, they perceive the time lost to be more expensive, because it was
    wasted.
    \[ B_S = \beta_t t_l \]  
    where \( \beta_t \) is the cost of time lost per hour and \( t_l \) is the cumulative hourly lost access time.
  - Death Penalty: cost associated with the user leaving the system ("death"). When a user
    leaves the system it is said to be "dead". The user can die due to the lack of bikes nearby, the
    destination being too far from the destination station, if the user cancels the trip beforehand(no
    nearby stations), if it is non-electric and the route is a climbing path.
    \[ D_P = L \times D_L \]  
    where \( L \) is the cost of life and \( D_L \) is the total number of deaths in a hour.

The coefficients values are the ones proposed by Casado et al. [40], which can be seen in Table 5.1.
These coefficients are an estimation of the real costs, but in real life, they don’t follow a linear function.
As such, comparing them directly has no value since their values are meaningless. We can, however,
compare the reposition methods within each metric.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_t )</td>
<td>Cost of repositioning per vehicle per hour</td>
<td>13.03</td>
</tr>
<tr>
<td>( \beta_t )</td>
<td>Cost of lost time</td>
<td>9.12</td>
</tr>
<tr>
<td>( L )</td>
<td>&quot;Death&quot; cost</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 5.1: Parameters for each Metric and the value used

5.4 Inputs to the Simulator

The several inputs to the simulator used are:

- **Station data**: coordinates in UTM, the station capacity and the initial inventory.
- **Demand and Attraction** for each minute and station. The demand(users/h) is the number of users
to be created which will have a certain station as origin, whereas the attraction (users/h) is an
estimation of the number of users arriving at that station. The trip generation will take into account
the attraction, but it is not guaranteed the number of users return bikes in a region is the same of
the attraction in that area.

- Service Area (Figure 5.1): defines the limit area for which the users are assigned an origin and
destination. Outside of this boundary, any users would be created would immediately die due to
being too far from any station. This boundary is defined as being approximately 1 km away from
the outer stations.

- $W_{max}$: distance a user is willing to walk to its origin and destination, is set to 10 minutes[41], which
  corresponds to 750 meters.

- Average riding distance: the average riding time is 17.5 minutes[41], as such the input given was
  4 km.

- Depot: position of the trucks at t=0 minutes, to where the trucks also return at the end of simulation
  at t=1400 minutes. In the case of CitiBike and Lisbon, its real position is unknown, so the location
  was set at the center of the service area.

5.5 Users creation and Trip Generator

An influence area is assigned to each station, in the form of Voronoi polygons[1], where the demand is
uniformly distributed. According to the demand in each area, the user is created and assigned a trip
distance, following a Poisson distribution. Next, potential destinations are obtained by setting an area
defined by a variable radius of $[d_s - W_{max}, d_s + W_{max}]$, where $d_s$ is the assigned trip distance and $W_{max}$
is the maximum distance the user is willing to walk. The final destination is the most likely station to have
a drop off event, which is obtained by a probability function defined as:

$$P(i) = \frac{\lambda_t(i)}{\sum_{k=1}^{X}\lambda_t(k)}$$

where $\lambda_t$ is the attraction rate for each station given as an input and $X$ is the total number of stations
in the potential destinations area. Similarly to the user creation, the final destination will be within the
influence area of the destination station, assuming a uniform distribution of destination.

Furthermore, the outer influence areas are inside the service region, which is a boundary 1km away
from the outer stations, as can be seen in Figure 5.1. The users can only use the system within the
service area, according to the demand of each influence area.

Figure 5.1: The service region is a boundary defined to be 1km away from the outer stations.

In Figure 5.2 it is shown the hourly demand of each system. The New York system has 2 defined rush hour demand peaks at 8 am and 4 pm. In Barcelona, there is a demand peak at 9 am and 2 smaller peaks at 3 pm and 18 pm. Lisbon shows a similar pattern as Barcelona. Additionally, the demand in New York is generally 3 times higher, which is normal due to the dimensions and population density of the city.
Figure 5.2: Demand during a day, in each system. The blue line corresponds to the real demand given as an input. The orange line corresponds to the users created by the simulator excluding the ones who die. When a user leaves the system, due to station assignment, the system generates another one, in order to adjust the system.
5.6 Repositioning Functions

The author made available 2 types of dynamic repositioning methods to be selected in the simulator. The first method is based on fixed routes and the second uses a pool system where the stations with a need/excess of bikes placed, in order to be rebalanced. There is also the option to not have continuous rebalancing, which is used as a baseline. These 2 methods are also in accordance with the spirit of the GDPR respecting the privacy concerns, similarly to our MPC model.

5.6.1 Fixed Routes

In this method, the map is divided into several areas, equal to the number of existent trucks. Every truck is assigned an area with a single route passing through all the stations. Independently of the current state of the stations, the reposition truck will follow the tour for the whole simulation and at each stop, would load the excess of bikes if it has enough capacity, or unload all needed bikes if available. If the station is balanced, the truck will skip it and proceed to the next one.

This method uses the Travelling Salesman Problem (TSP) to assign a route to each truck. The TSP problem produces a route in which the truck travels to each station only once and goes back to the origin station in the end, while minimizing the total distance\[42\].

5.6.2 Pool System

In this method, the stations with a lack or excess of bikes will set an alarm and are introduced into a pool. In Figure 5.3 it is shown the scheme which will cause an alarm. Every time there is a pair of stations, in which a station is a provider of bikes and the other a consumer, a truck is assigned for repositioning, using the Hungarian method.

However, there is a setback: in case there are too many empty stations, the reposition will only be made after there is a station that sounds the alarm for bike excess. This happens especially at the beginning of the rush hours, when there is more renting than returning events.

\[\text{Figure 5.3: Alarms definition scheme for continuous reposition} \[40\]\]

5.7 MPC for Rebalancing

In this first approach to our MPC repositioning method, we are considering a time slot \(t\) of 30 minutes, with a frequency of 2/h. This means all the trucks must finish the repositioning meanwhile, before a new one is assigned. The parameter \(D_{\text{max}}\) (from section 4.4) defines the number of reposition controls between stations. A higher distance between stations leads to the graph having more arcs. We are assigning it a "Manhattan" distance of 800m.
For computational reasons, we are using a time horizon of $T=9$, which equals to 4.5 hours in advance when using a time slot of 30 mins.

Additionally, we are considering an infinite amount of trucks available, although we can use at most the number of trucks equal to the number of arcs between the stations in the graph.

Firstly, we tested the model from Equation 4.4, which minimizes the difference between the optimal inventory value and the R-norm of the control links $||u||_R$.

It is important to note that the baseline reposition methods used have real-time monitoring to choose the next destination, whereas the MPC predicts the user movements. Similarly to a real life application, there are a lot of uncertainties in the demand prediction, as the simulator randomly creates users within the most popular areas and if the user dies, the simulator compensates by creating another user in a different place than what was predicted.

5.7.1 Metrics Evaluation

The metrics from Section 5.3 are used to compare the MPC with the different rebalancing methods designed by Casado [40], in the 2 systems: Citibike (New York), Bicing (Barcelona).

The Figures below represent each metric for each hour of the day, for the 4 reposition methods: MPC, Fixed Routes, Pool System, No reposition.

![Reposition](image1.png)

(a) New York

![Reposition](image2.png)

(b) Barcelona

Figure 5.4: Hourly Reposition Metric, for each reposition method. The MPC has a much higher cost of reposition than the baseline methods.

In Figure 5.4, it is shown the Reposition metric, one of the most important metrics as it tells the cost of moving the trucks (Equation 5.1). It accounts for the fuel and salary of the operators and it is the only metric related to the cost to the agency.

The MPC has a high cost of repositioning, as it usually only moves small numbers of bikes and uses more trucks than the other methods. It is also important to note the repositioning cost of the Fixed Routes and Pool system is similar, although the Pool System shows better results overall in the other metrics.

In Figure 5.5, the bad service metric (Equation 5.2) is higher whenever there is a rerouting of the user due to empty/full stations. A lower value is a good indicator of the quality of the repositioning. For both Bicing and Citibike, the fixed routes method has very little advantage compared to no repositioning, whereas the pool system lowers the bad service metric by more than half. The MPC has the best results
in Barcelona and New York in the Bad Service metric, as they are bigger systems and benefit from the approach that the MPC has on minimizing the customer loss for the whole system.

Figure 5.6: Death Penalty Metric comparison. The MPC has the advantage as had the least number of users leaving the system due to having no bikes available.

The Death Metric not only gives us the quality of the repositioning, but also the quality of the system itself. Comparing both Citibike and Bicing results in Figures [5.7b] and [5.7a] for the Fixed Routes repositioning method, it can be seen that a lot of users leave the system in Barcelona not only due to lack of bikes when arriving at an empty station, but also due to their destination being far from the docking station. This is caused by a lower station density than New York’s and because of the large boundaries of the service region defined. On the contrary, for Citibike, there are barely any dead users due to this reason as the system is denser even though it also has a higher user demand. Analyzing Figure [5.6] both the Pool System and the MPC present the best results. Additionally, the Fixed Routes method has
little advantage compared to no repositioning method.

Figure 5.7: Number of users who leave the system due to lack of bikes available, or the destination is farther than 10 minutes walking from the system. It is important to note several users die in the Bicing system (Barcelona) due to lower station density. These plots were extracted from a simulation using the Fixed Routes method.

On the other hand, for the Pool System the total metric cost decreased, which means the reposition was effective: the increase in repositioning led to a decrease in customer loss.

As a final remark, in the three systems, the target inventory throughout the entire day is the initial inventory. This is not optimal since there is an asymmetric flow in the morning and afternoon, hence we would benefit by having different target inventory values for each rush hour. As mentioned in Section 4.8, several studies have been performed in this case and further work should be developed to continue.

5.8 MPC Reposition Cost Reduction

5.8.1 MPC regularization

Comparing to the methods designed by the author as can be observed in Figures below, it is observed that the MPC obtained the best results in terms of providing the best service for the user, as can be seen by the death and bad service metrics. However, it came with a high cost of repositioning.

One way to reduce the repositioning metric is to decrease the number of non-zero control pairs $u_{i \rightarrow j}$. To achieve sparsity, $l_1$ norm serves as an approximation to $l_0$ (cardinality), but has the advantage of being convex and thus efficient to compute. Hence, we use $l_1$ regularization to encourage many of the control links $u_{i \rightarrow j}$ in our model to be exactly 0 and reduce the number of repositions. This is done by replacing the $R$-norm by the $l_1$ norm in the objective function:

$$\sum_{t=1}^{T} (x(t) - \bar{x}(t))^T Q(x(t) - \bar{x}(t)) + \sum_{t=0}^{T-1} \lambda ||u(t)||_1$$ (5.4)

where the parameter $\lambda$ controls the trade-off between the repositioning cost and providing a better service for the user.

The weight of $||u||_1$ is modified for several test runs and the effect on each metric was analyzed. Therefore, $\lambda$ is assigned several values comprised between $10^{-3}$ and $10^7$ and its influence on the KPIs is done logarithmically.
The following Figures plot the influence of the parameter $\lambda$ on each metric:

(a) New York

(b) Barcelona

Figure 5.8: Reposition metric for different values of the parameter $\lambda$. A higher lambda make the MPC minimizing the number of controls, reducing the reposition costs. A lower $\lambda$ will force the controller to be more "nervous" and do more repositions, which in turn will cause an increase in the reposition metric. As the parameter increases, the reposition metric lowers since the controller will further minimize the control.

(a) New York

(b) Barcelona

Figure 5.9: Bad Service metric for different values of the parameter $\lambda$. A higher lambda will reduce the service quality. With a decrease in the number of repositions with lambda, the service quality will also drop. The bad service metric represents the detours by the user when the station becomes empty upon arrival, or full when the user wants to dock. The users included in this metric have done a re routing to a nearby station no farther than 10 minutes on foot.
Figure 5.10: Death metric for different values of $\lambda$. A higher lambda will reduce the service quality

Similar to the Bad Service Metric, the death metric increases with lambda, as the number of repositions decrease and users die. A high death/bad service metric will lead to a decay of the system in the long term, as people will stop using the system and the bad publicity spreads fast. It is the system administrator’s choice to choose between providing the best service, or give up a little bit of quality to lower the associated costs.

The next set of Figures represent the trade off between each metric of the baseline methods, MPC with $||u||_R$ norm minimization and $\lambda||u||_1$ norm minimization with different values for the weight $\lambda$. The MPC points are represented by blue-colored points while baseline methods are represented by different colored points:

Figure 5.11: Trade-off between Reposition metric and Bad Service+Death in New York. The blue points represent the MPC with the different $\lambda$ as the regularization parameter and also the MPC with R norm minimization. The MPC has the advantage regarding user service, presenting fewer users that leave the system, or re-route from their original station due to empty/full stations. However, the MPC loses to the baseline methods in terms of reposition costs.
In Figures 5.11a,b, it is shown the advantages and disadvantages of each reposition method for the City of New York. As expected, a lower $\lambda$ for the $l_1$ norm of the MPC leads to a lower Bad Service Penalty, at the cost of repositioning. Users don’t detour from their original station compared to the baseline methods, providing a better service.

For $\lambda \leq 100$, the service provided to the user is sensibly worse, but the repositioning cost lowers. Comparing to the baseline methods, in the Pool System users have to re-route from their original station just as much, but there are a higher number of deaths. Having no reposition leads to a terrible service, which is seen by the red dot on the far right of both Figures. The Fixed Routes method has little advantage over no reposition, having similar deaths and re-routing by the users. These methods however have a low reposition, which the system administrator might prefer if they want to save up money on operational costs. As confirmed by Figures 5.10, 5.8 and 5.9, from values of $\lambda$ between 10 and 1000, there is a sudden rise in the repositions costs with $\lambda$ and a decrease in reposition costs. This causes the clustering of data points seen in Figures 5.11 and 5.12.

In Barcelona, the results are similar. However, the baseline method Pool System provides a better customer service than the MPC $l_1$ norm regularized with $\lambda < 100$, at a much lower reposition cost. As such, if the system administrator wants to trade quality for reposition cost, they can only choose a MPC with a lower $\lambda$ than 100. The other baseline methods still hold the worst customer service, as predicted, but present low reposition costs.
5.9 Test case: Lisbon

(a) Bad Service metric. The MPC is showing poor performance, even having worse values than the no reposition baseline, as a result of poor prediction.

(b) Death Penalty. The MPC and the Pool System have the advantage in this had few users leaving the system due to unavailable stations nearby.

(c) Reposition metric. The MPC has the highest reposition costs, meanwhile the Pool System and Fixed Route methods have a low value.

Figure 5.13: Different metrics throughout the day in the city of Lisbon

The simulator was given as input the demand and attraction, which were the hourly average pick up and drop off demands during the month of June, in 2018. However, since the GIRA system is not matured yet, the reads in the demand did not follow the pattern usually observed in the matured cities, in which the working areas are jammed of bikes in the morning, due to the commute to work; and in the evening the working areas are emptied. As such, we think the simulator did not generate properly simulate these fluctuations and the MPC did not accurately predict the demand. Additionally, due to being a smaller system, the simulator might consider a higher cycling time for an average trip than what Lisbon users ride. The implementation of the MPC controller in the GIRA system is just a curiosity. In a system in development, demand prediction becomes a harder task and might even do worse than no reposition, as is observed in this case. The MPC has high reposition costs which are not justified, while the pool system has the best results overall.
To finalize this test case, it is seen the Pool system is always on the left of all the MPC variations, which means it holds the least “deaths” and re-routings of the users, while at the same time keeping the reposition costs low.

5.10 Performance review

It was observed the MPC obtained the best results in terms of providing the best service for the user. However, it came with a high cost of repositioning. The ideal reposition method uses a small $\lambda$ with a well-tuned target value, so that it provides the best service for the user with no empty or full stations. However, the cost of reposition in order for that to happen can be incomportable. This way, we have to give up quality of the system to be economically viable.

Increasing the number of vehicles for the baseline methods, in order to attain better results in the user cost metrics, which in turn increases reposition costs, is computationally incomportable as these methods must have a low computational time to be used in real-time. This way, it is impossible for them to achieve the lower user cost metric values that our MPC method did.

The predictive abilities regarding the user movements may make the MPC perform poor rebalancing decisions. The MPC does not have access to the exact rent and return demands at each station, but instead uses real data which is also given to the simulator. The discrepancy is due to the trip generation by the simulator, which has several parameters, such as the median trip distance, which the MPC does not have access to, simulating the real life uncertainty of the user movements.

To conclude, each method has its advantages and disadvantages regarding better customer service or less reposition costs, thus there is no best method. The MPC can provide the best service in the cities of New York and Barcelona compared to the real-time monitoring reposition methods, but in the end, it is up to the system administrator to decide if the trust of the customers and better mobility in the cities have more value than the operational costs.
Chapter 6

Discussion and Future Work

We began this work by doing an Exploratory Data Analysis on the datasets available on both GIRA (Lisbon) and Citibike (New York) systems. It allowed us to identify the minimum common data between both systems so we could design a data model which could be applied to both systems. The maturity of the Citibike system leads to a well-defined relationship between the weather variables and the pick up demand. On the other hand, the GIRA system is still in development, to which the number of stations and bikes increased during the data collection period. This eliminated the pattern between the demand and the weather variables throughout the year, as the user base is different during the different seasons, making these incomparable.

Due to the recent European Data Protection Law, GIRA system has a more restrict dataset, lacking the trip records that are available for the Citibike system. Hence, we only had data on the inventory of each station, having no information on how the system behaved. The lack of this dataset also imposed incompatibility with the state of the art demand prediction algorithms, as most created a station network to better predict the system dynamics.

The prediction algorithm we used was an adaptation from previous work, which considers the weather variables, and uses the $K$ most similar days to predict the pick up and drop off demands. This multi-source demand prediction algorithm is better than the one-factor prediction algorithms used as baseline, as it considers not only historical demand data but also the weather features. As future work - consider events in the city for prediction (have a database and add a special parameter for it)

To solve the dynamic repositioning problem, we presented a Model Predictive Control which is efficient to compute due to its convexity. Furthermore, it attenuates the lack of knowledge on the system dynamics, by considering the individual demands of each station and producing a solution considering the system needs.

A simulator was implemented to compare the performance of our reposition method with existent reposition algorithms. The efficiency of the different algorithms was attested using different metrics, which take into account the quality of the service provided to the user and the cost of reposition.

The MPC produced better results in regards to reducing the number of re-routings by the users due to finding an empty station when picking up a bike, or a full station when trying to the bike, as well as reducing the number of users who leave the system due to having no available stations nearby. However, it has the disadvantage of requiring a high number of trucks. Each method has its advantages and disadvantages, and it is up to the system administrator to choose if trading off a better user service for a reduction in operational costs is viable. The repositioning cost of the MPC is high as it considers an infinite number of trucks available. Further work should be developed in order to design a model which limits the number of trucks. The formulation would be a combinatorial problem, which is more challenging to compute.
Additionally, our MPC model requires that all repositions finish during a specific time slot, and a new reposition is generated after all others are completed, in order to keep the system coherent as new repositions would overlap the old ones since the model does not consider the repositions in progress. Future work could focus on formulating the combinatorial problem in a multi-agent way, which would consider the active repositions, and a new action is given to the truck as soon as it finishes the reposition, without waiting for the completion of the other trucks.

A future approach to the problem could consider a reposition which balances the number of electric and classic bikes in the stations, as users are more inclined to use electric bikes in a hilly city such as Lisbon and Barcelona.
Bibliography


Appendix A

Proof of convexity of our MPC

A.1 Notation

$||x||$ represents the generalization of the absolute value in $\mathbb{R}$. When we specify a particular norm, we use the notation $||x||_{\text{subscript}}$, where the subscript is a mnemonic to indicate which norm is meant.

A.2 Convex Functions

A convex optimization problem is one of the form:

$$\begin{align*}
& \text{minimize} & f_0(x) \\
& \text{subject to} & f_i(x) \leq b_i, i = 1, \ldots, m,
\end{align*}$$

where the functions $f_0, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ (both the objective and constraints functions) are convex.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain is a convex set and satisfies:

$$f(\theta a + (1 - \theta) b) \leq \theta f(a) + (1 - \theta) f(b), \forall a, b \in \text{dom} f, 0 \leq \theta \leq 1$$

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called a norm if:

- (1) $f$ is non-negative: $f(x) \geq 0$,
- (2) $f$ is definite: $f(x) = 0$ only if $x=0$
- (3) $f$ is homogeneous: $f(tx) = |t|f(x)$, for all $x \in \mathbb{R}^n$
- (4) $f$ satisfies the triangle inequality: $f(x + y) \leq f(x) + f(y)$, for all $x, y \in \mathbb{R}^n$

Norm $l_p$ with $1 \leq p \leq +\infty$, is proven to be a norm.

With $p = 2$, we obtain the $l_2$ norm, commonly referred to as Euclidean Norm, which gives the magnitude of the vector.

For $p=1$, it results in the $l_1$ norm $\sum_i |x_i|$, which is the sum of all the components of a vector.

49
Our problem is formulated as follows:

\[
\begin{align*}
\text{minimize} & \quad x^{(1)}, \ldots, x^{(T)}, u(0), \ldots, u(T-1) \\
& \quad \sum_{t=1}^{T} (x(t) - \bar{x}(t))^T Q(x(t) - \bar{x}(t)) + \sum_{t=0}^{T-1} u(t)^T R u(t) \\
\text{subject to} & \quad x(t+1) = x(t) + A u(t) + B v(t), \quad (x_0 \text{ given}) \\
& \quad 0 \leq u(t) \leq V_C, t = 0, \ldots, T-1 \\
& \quad x \leq x(t) \leq \bar{x}, t = 1, \ldots, T
\end{align*}
\]

(A.3a)

(A.3b)

(A.3c)

(A.3d)

It is a convex problem if both the objective and constraints functions are convex\[43\]. The objective function is a sum of quadratic functions. The first quadratic function can be represented by

\[
\begin{align*}
& \quad f_1(x) = w^T Q w, \text{where } w = x - \bar{x}. \\
& \quad \text{By the definition of convexity from Equation } A.2
\end{align*}
\]

(A.4)

for \( \theta \in [0, 1] \).

Simplifying:

\[
\begin{align*}
& \quad \theta^2 a^T Q a + (1 - \theta)^2 b^T Q b + \theta(1 - \theta) a^T Q b + \theta(1 - \theta) b^T Q a \\
& \quad \leq \theta a^T Q a + (1 - \theta) b^T Q b \\
& \quad \Rightarrow \theta(1 - \theta) a^T Q a + \theta(1 - \theta) b^T Q b - \theta(1 - \theta) a^T Q b - \theta(1 - \theta) b^T Q a \geq 0 \\
& \quad \Rightarrow a^T Q a + b^T Q b - a^T Q b - b^T Q a \geq 0 \\
& \quad \Rightarrow (a - b)^T Q (a - b) \geq 0
\end{align*}
\]

(A.5a)

(A.5b)

(A.5c)

(A.5d)

which holds true when \( Q \) is positive semi-definite \( Q \succeq 0 \). If \( Q \) is positive definite, then the function is strictly convex as the condition only holds true if \((a - b)^T Q (a - b) > 0\). If \( Q \) is the identity matrix, then it is the \( L_2 \) norm of \( w \).

The same applies for the second quadratic function \( f_2 = u^T R u \). The convexity of the objective function is preserved as it is a minimization of a summation of non-negative convex functions\[43\].

Considering the convex set which defines the domain of \( u \) and \( x \) in Equations A.3c,d; the linear constraint (Equation A.3b) is composed of a sum of convex functions in their domain. Hence, the MPC has convex constraints.

A regularization using \( L_1 \) norm is also presented in our formulation:

\[
\begin{align*}
& \quad \sum_{t=1}^{T} (x(t) - \bar{x}(t))^T Q(x(t) - \bar{x}(t)) + \sum_{t=0}^{T-1} \lambda ||u(t)||_1
\end{align*}
\]

(A.6)

A \( L_1 \) norm is a norm, thus it is convex non-negative by the definition.

Analogously, the summation of non-negative convex functions preserves the convexity of our objective function.

With convex constraints and objective functions, our problem is convex.