



Bike Sharing System Simulator with User Incentive Methods

Rodrigo Miguel de Jesus Plácido

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Duarte Pedro Mata de Oliveira Valério

Prof. Susana Margarida da Silva Vieira

Examination Committee

Chairperson: Prof. Carlos Baptista Cardeira

Supervisor: Prof. Duarte Pedro Mata de Oliveira Valério

Member of the Committee: Prof. António Ramos Andrade

September 2021

Dedicated to my life inspiration and forever beloved grandfather.

Acknowledgments

I would like to express my appreciation to EMEL for collaborating with me in this study.

I am particularly grateful for the counselling and monitoring given by my two supervisors: Duarte Valério and Susana Vieira, whose timely and meticulous feedback were crucial to the writing of this thesis.

Ultimately, a thank you is not enough to express my gratitude to all my family and friends. A special thank you to my mother and sister for their patience and unconditional support, without whom nothing would have been possible; to my grandparents who are everyday inspiration of how hard work and honesty pay off; to my girlfriend who pushes me to be better every day. For this, I am extremely grateful.

Resumo

O interesse em criar cidades sustentáveis e verdes livres de emissões de carbono cresceu nas últimas décadas, com os meios de transporte não poluentes a assumirem um papel fundamental neste processo de desenvolvimento. Ao aderir a estes critérios, e ao conceito de mobilidade partilhada, surgiram os Sistemas de Bicicletas Partilhadas (SBP). Estes estão presentes em centenas de cidades e são hoje uma alternativa competitiva aos automóveis, metro ou autocarros. No entanto, a sua manutenção em condições perfeitas, para que os clientes finais tenham sempre uma boa experiência, não é uma tarefa fácil. Os fluxos de bicicletas tendem a gerar uma falta ou excesso das mesmas em algumas estações, impedindo os utilizadores de as recolher ou largar. Existem duas grandes alternativas para resolver este problema e levar a rede a uma condição estável: ou os operadores deslocam as bicicletas utilizando veículos para as transportar entre estações, ou os utilizadores envolvem-se no processo de balanceamento da rede através de incentivos. Este documento está centrado na segunda alternativa. Para a avaliar foi criado e implementado um modelo baseado em dados históricos do SBP de Lisboa, Gira, num software de simulação (Anylogic). Trata-se de um modelo baseado em agentes com uma interface visual onde as viagens de bicicleta e os estados das estações podem ser observados em tempo real, e onde alguns parâmetros do modelo podem ser facilmente alterados. Dois métodos de incentivo ao utilizador foram computados e testados no simulador, tanto individualmente como em conjunto com um reposicionamento noturno através de veículos. Após várias simulações, os resultados mostraram que se todos os utilizadores participassem em tarefas de reposicionamento, seria possível manter o sistema equilibrado. Uma vez que esta não é uma tarefa fácil, concluiu-se que, para as metodologias experimentadas, e de forma a manter os requisitos de ocupação das estações, os métodos de incentivo ao utilizador e as tarefas de reposicionamento têm de ser utilizados em conjunto. O efeito que os métodos de incentivo ao utilizador têm sobre o número de estações visitadas e de bicicletas transportadas durante o reposicionamento é também mostrado. Embora neste trabalho sejam apresentadas algumas conclusões valiosas, deve ser feita uma investigação futura sobre os custos relacionados com os incentivos e o seu efeito nas tarefas de reposicionamento, tendo sempre em conta a percentagem de cooperação dos clientes.

Palavras-chave: Sistema de bicicletas partilhadas, Incentivos ao utilizador, Reposicionamento estático, Simulador

Abstract

The interest in creating sustainable and green cities free of carbon emissions has grown in more recent decades, with non-pollutant means of transportation taking a fundamental role in this development process. By joining these criteria, and the concept of active shared mobility, Bike Sharing Systems emerged. They are present in hundreds of cities and are nowadays a competitive alternative to cars, metro or buses. Nevertheless, maintaining these systems in perfect conditions, so that final customers always have a good experience, is not an easy task. Bike flows tend to generate a lack or excess of bikes in some stations preventing users from picking up or dropping off bikes. There are two major alternatives to solve this issue and bring the network to a steady condition: either operators relocate bikes using trucks to transport them between stations, or users are engaged in the balancing process using incentives. This document is focused on the second alternative. To evaluate it, a model was created and implemented based on historical data from Lisbon BSS, Gira, in a simulation software (Anylogic). This is an agent-based model with a visual interface where bike trips and station states can be observed in real-time, and some model parameters can be easily changed. Two user incentive (UI) methods were computed and tested in the simulator, both alone and in conjunction with a nightly vehicle repositioning. After several simulations, results have shown that if all users participate in repositioning tasks, it would be possible to keep the system balanced. Since this is not an easy task, it was concluded that, for the experimented methodologies, and to maintain station level requirements, user incentive methods and repositioning tasks have to be both used. The effect that UI methods have on the number of visited stations and transported bikes during repositioning is also shown. Although some valuable conclusions are presented in this work, future research about the costs related to UI and its effect on repositioning tasks, also considering customer cooperation, must be taken.

Keywords: Bike-sharing system, User incentive, Static repositioning, Simulator

Contents

Acknowledgments	iii
Resumo	iv
Abstract	v
List of tables	ix
List of figures	x
List of acronyms	xii
1 Introduction	1
1.1 Topic overview	2
1.2 Motivation	3
1.3 Objectives	3
1.4 Contributions	4
1.5 Thesis outline	4
2 Related work	5
2.1 Static	5
2.2 Dynamic	7
2.3 User incentives	8
2.4 Simulators	9
3 Theoretical background	11
3.1 Trip generation and trip time	11
3.1.1 Poisson distribution	11
3.1.2 Lognormal distribution	12
3.2 User behaviour and incentives model	13

3.3	Repositioning problem	14
3.3.1	Routing problem	16
3.3.2	Clustering problem	17
3.3.3	Heuristic MIP	18
3.3.4	Skellam distribution	19
4	Gira BSS characterization	21
4.1	Geographical distribution and flow intensity	21
4.2	Temporal patterns	25
4.3	Operator tasks	29
5	Implementation	33
5.1	Used software	33
5.2	Modelling	34
5.2.1	Trip generation	34
5.2.2	User behaviour	36
5.2.3	User incentives (UI)	36
5.2.4	Repositioning	38
5.3	Anylogic simulator	40
5.3.1	Agents	40
5.3.2	Simulator inputs/outputs	41
5.3.3	Simulation procedures	43
5.3.4	Visual interface	44
6	Evaluation	47
6.1	Data filtering	47
6.2	Parameters choice	48
6.3	Verification and validation	49
6.4	Results and discussion	54
6.5	Service level	55
6.6	Average extra effort	57
6.7	Lost users	59

6.8 Repositioning results	60
7 Conclusions and future research	63
7.1 Conclusions	63
7.2 Future work	64
Bibliography	65
Appendix A	71
A.1 Trip time figures	71
A.2 Repositioning results	72
A.3 Agents	74
A.4 Principal simulator codes in Java	79
A.5 Simulator user manual	87

List of tables

5.1	AMSE between real and fit PMFs of the generated trips for all combinations of (s_{out}, s_{in}, t)	35
5.2	AMSE between real and fit PMFs of the total trip time from s_{out} to s_{in} for all combinations of (s_{out}, s_{in})	36
6.1	Model versus simulation error metrics	51
6.2	Average service level (without NR; CC=1.0; Rad=1.0 km)	57
6.3	Average extra effort (without NR; CC=1.0; Rad=1.0 km)	58
6.4	Average percentage of lost users (without NR; CC=1.0; Rad=1.0 km)	60
6.5	Repositioning results with 4 trucks	61
A.1	Principal parameters of agent Main	75
A.2	Principal parameters of agent Station	75
A.3	Principal parameters of agent Person	76
A.4	Principal parameters of agent Bike	77
A.5	Principal parameters of agent Truck	78

List of figures

3.1	Poisson distributions for $\lambda = 5, 15$ and 25	12
3.2	Lognormal distributions	13
3.3	Skellam distribution	19
4.1	Network geographic distribution (March 2019)	22
4.2	Number of available docks in stations 105, 420 and 472	23
4.3	Number of available bikes in the network at 6 AM (start of operation)	23
4.4	Total number of trips per day	24
4.5	Trip intensity map	24
4.6	Average number of trips throughout a day for all weekdays	25
4.7	Bubble plot of station difference between arrivals and departures	26
4.8	Station 305 daily (workday) pattern of arrivals and departures	27
4.9	Station 307 daily (workday) pattern of arrivals and departures	27
4.10	Station 486 daily (workday) pattern of arrivals and departures	28
4.11	Station 446 daily (workday) pattern of arrivals and departures	28
4.12	Station 481 daily (workday) pattern of arrivals and departures	29
4.13	Station 413 daily (workday) pattern of arrivals and departures	29
4.14	Percentage of tasks per type of operation	30
4.15	Average number of repositioning visits per day at each station	30
5.1	PMF of the average number of trips between start and end stations (8:40 AM to 9:00 AM)	35
5.2	Real PMF and other PDF fit functions to trip time between start and end stations	37
5.3	Stacked bar graph of types of cooperation	38
5.4	Poisson distribution functions ($\lambda_{in}=15, \lambda_{out}=10$)	39
5.5	Skellam distribution function ($\lambda_{in}=15, \lambda_{out}=10$)	39

5.6	Trust level distribution ($\lambda_{in}=15$, $\lambda_{out}=10$)	39
5.7	Simplified flowchart of the Anylogic model	45
6.1	Cumulative percentage of trips for different trip times	48
6.2	Average number of trips throughout day - real versus simulation (Scenario 1; workday) . .	52
6.3	Average number of trips throughout day - real versus simulation (Scenario 1; weekend) .	52
6.4	Scenario (3) (no repositioning) VS scenario (4) (with repositioning) average number of lost users throughout the day	53
6.5	Example of repositioning tasks in the map (4 trucks)	54
6.6	Service level over 28 days	56
6.7	Service level (without NR; Rad=0.5 km)	56
6.8	Service level (without NR; CC=0.5)	56
6.9	Service level (with NR; Rad=0.5 km)	56
6.10	Average number of neighbours around a station	58
6.11	Average extra effort (without NR; Rad=0.5)	59
6.12	Average extra effort (without NR; CC=0.5)	59
6.13	Average extra effort (with NR; Rad=0.5 km)	59
6.14	Percentage of lost users (without NR; Rad=0.5 km)	60
6.15	Percentage of lost users (without NR; CC=0.5)	60
6.16	Percentage of lost users (with NR; Rad=0.5 km)	60
A.1	Real PMF (Lognormal fit) versus simulation PMF of trip time (all trips)	71
A.2	Real PMF (Lognormal fit) versus simulation PMF of trip time (trips from 101 to 104) . . .	72
A.3	Real versus simulation average number of repositioning visits per day at each station . .	72
A.4	Example of stations' level before and after repositioning	73
A.5	Agent Main in Anylogic	74
A.6	Agent Station in Anylogic	75
A.7	Agent Depot in Anylogic	76
A.8	Agent Person in Anylogic	76
A.9	Agent Bike in Anylogic	77
A.10	Agent Truck in Anylogic	78

List of acronyms

AMSE Average Mean Squared Error

BSS Bike Sharing System

CC Customer Cooperation

CPU Central Processing Unit

DB Database

M1 Method 1

M1-0.5 Method 1 with WPC=0.5

M1-0.7 Method 1 with WPC=0.7

M1-1.0 Method 1 with WPC=0.1

M2 Method 2

M2-0.7 Method 2 with WPC=0.7

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

MIC Minimal Intervention Control

MILP Mixed Integer Linear Programming

MIP Mixed Integer Programming

MSE Mean Squared Error

NC No Control

NR Nightly Repositioning

O-D Origin Destination

PC-MA Preemptive Control via Mobile App

PDF Probability Density Function

PMF Probability Mass Function

Rad neighbourhood Radius

UI User Incentive

VAF Variance Accounted For

WPC-MA Weak Preemptive Control via Mobile App

Chapter 1

Introduction

In more recent years, Bike Sharing Systems (BSS) have been widely adopted in many major cities all over the world. The adoption of the bike as a mean of transportation is expected to continue growing, mostly due to governments' efforts to face climate changes, reduce traffic, and also due to increasing health awareness [1–3].

These systems consist of bikes spread around the city and where users can pick them up, travel to their intended destination and drop off the bikes “at specific locations or anywhere in the city depending on the type of (. . .) system, locking technology, and payment mechanisms.” [2].

According to Nath and Rambha [2], two types of systems are recognised: free-floating, which allows people to place bikes anywhere; and station-based, which forces people to pick up and drop off bikes from stations. These stations may use docks or geo-fences to hold bikes. This document will only address the second system type since the problem addresses Lisbon BSS, which is station-based. However, with either free-floating or station-based, trips dynamics may take too many bikes from or to determined stations emptying them out or overloading. There would be no problem if the difference between arrivals and departures was kept close to zero; however, this does not happen for every station: some are more propitious to have departures and other arrivals, leaving those stations in unbalanced states. Besides the possibility of not encountering a bike if a station is empty, a problem that can be solved by using other means of transport, stations might also be full. Actually, this represents a bigger problem since it does not allow people to dock their bikes and consequently forces them to travel to another station with empty docks.

The most common way to deal with this issue is by using trucks to move bikes between stations. BSSs surpass this difficulty by applying either static repositioning (system rebalancing when the system is closed for users, usually during the night shift) or dynamic repositioning (rebalancing operations while the system is open for users) [2, 4, 5].

Another approach to this issue is by having a direct influence on user trips, recommending them to grab bikes at stations highly occupied and/or drop off bikes at stations with low occupations. This alternative

may be achieved by changing the prices of trips (the most used technique), offering points, etc [2, 6, 7].

This work addresses Lisbon station-based BSS, Gira, and user incentive approaches to keep the system balanced. An interactive agent-based simulation of Gira is created in Anylogic, based on treated historic data, to estimate the impact of user-based approaches in service quality (percentage of customers satisfied) and repositioning tasks of a Nightly Repositioning (NR). The following sections shall bring an overview of the problem context and the present study importance to the scientific community.

1.1 Topic overview

The rise of automobile usage around the world led to an increasing concern on traffic and gas emissions; consequently, eco-friendly transportation started to be promoted and widely adopted. According to Maibach, Steg, and Anable [8], in first-world countries such as the United Kingdom, the Netherlands and the United States of America, half of the car trips are “less than 5 miles” long, and one of the alternatives are bike sharing systems since they are a good option for short-distance trips and increase connectivity to public transportation networks. Their rapid growth in many countries is also due to lower set up costs charged by this kind of system [2].

In 1965, the first generation of Bike Sharing System took its first steps in Amsterdam with only 50 bicycles (White Bicycle Plan) [9]. Followed by Vélos Jaunes in La Rochelle, France (1974) and Green Bike Scheme in Cambridge, United Kingdom (1993) [2]. Bicycklen program in Copenhagen in 1991 was one of the first second-generation BSSs [2]. They had the particularity of being coin deposit. Customers had to insert coins to unlock bikes and would receive them back once they returned bikes.

These systems failed due to theft and security issues [2]. Since then, BSSs have undergone many changes. The third generation was launched in 1995, with IT-based systems incorporating “advanced technologies for bicycle reservations, pick up, drop off, and information tracking” [10]. The fourth generation was marked by the communication with a mobile app and “real-time information on bike availability” [2]. Recently, the fifth generation emerged, and with it dockless bikes appeared, resulting in free-floating setups [2]. From December 2016 onwards there are about a thousand cities with BSSs [11]. The benefits of BSSs also include the fact that cycling is accepted as a healthy travelling choice for individuals, over-weighting the health risks with a benefit/risk ratio of 19:1 [12]. Although BSSs have been encouraged by public agencies and users around the world, there are still some challenges to face, such as maintenance costs, low profits, theft and vandalism [13–15]. Nevertheless, such issues have been addressed to a certain extent, through the use of technologies such as GPS or anti-theft alerts [16, 17]. Another problem for the success of BSSs is the user’s perspective of cycling as an unsafe commute mode, with mixed traffic and the lack of dedicated bike lanes [2].

1.2 Motivation

Operating the rebalancing of a BSS by truck takes a lot of effort, leading to considerable costs and not contributing to the “green” transportation goal if these are powered by pollutant fuels [6].

That is why any way of reducing the number of rebalancing trips by truck must be taken into consideration and evaluated. The most used strategy is to constantly try to improve the rebalancing by reducing the number of trips or travelled distance [18, 19], but there is another strategy that could have, or not, an impact on system rebalancing: user participation [7, 20]. If every time a station is expected to be empty or full the user accepts the alternative station that is presented to him, the probability of stations getting full or empty would be reduced, as well as truck operating costs. Although this model seems ideal, it is not guaranteed to be the best: incentive costs should also be taken into account to balance revenues and expenses [2].

Lisbon BSS, Gira, has already implemented a user incentive strategy for people docking bikes on stations with occupations below 30% and departing from a station with an occupation above 70% [21]. The incentives received consist of attributing extra points, in this case, 100 points, that can be used to pay for future trips, since 500 points worth 1€ and a daily pass costs 2€.

However, the impact of this or other strategies has not yet been tested in the past. This work will attempt to give some answers by evaluating the performance of a BSS with and without incentives, with varying customer cooperation (CC) percentages.

1.3 Objectives

This study aims to understand how much impact users could have in the balance of Gira BSS, if they actively participate in repositioning, by changing the start and end stations of their trips. It is also intended to evaluate the effect that user incentive strategies could have in truck-based repositioning. The following objectives have been set in order to fulfil this aim:

- Properly analyse data;
- Create a simulator of Gira BSS in Anylogic, based on historical data, with a visual interface for configuration of variables and visualisation of bike trips around the city and station states;
- Implement two types of user incentive methods, with one analogous to the current methodology used by Gira.
- Implement a static repositioning method.

1.4 Contributions

The model and the simulator presented in this thesis can be of major interest to Gira, given that it evaluates the impact of a user incentive strategy by withdrawing information on the percentage of dissatisfaction and the impact on truck repositioning operations.

Another contribution is the simulator itself, which could be applied not just for the purpose of this thesis but in different projects related to Gira.

1.5 Thesis outline

This document comprises 7 chapters:

- The first chapter introduces the problem under analysis, the methods used and the importance of presenting solutions to address it. Afterwards, research objectives and contributions are stated. This last section presents the structure of the document by chapter.
- Chapter 2 approaches relevant previous works related to this by topic of research. The methodologies and software used or the objectives traced by each are some of the issues addressed.
- Chapter 3 covers the theoretical background of this research by explaining the models applied and the probability distributions used.
- To better understand Gira BSS, its characteristics, the patterns of its users, as well as the number of redistribution actions implemented by operators, are introduced in chapter 4. A map of the stations with their names and a trip intensity map is shown. Patterns along the day for the whole system and for different types of stations are also addressed.
- Chapter 5 includes an analysis of the simulator implementation and the models proposed to evaluate the main topic of this thesis. All the choices and the assumptions made are explained, a diagram of the simulation is shown and all simulator agents are presented.
- Chapter 6 starts by verifying and validating the simulator. Then, results withdrawn from several simulations (with different models and parameters) are analysed. Models are compared, for example, by service quality measures or the number of repositioned stations.
- Last chapter starts by assessing if the initial objectives were attained or not, presenting then the conclusions and finally future research on the studied area along with some applicabilities this simulator may have in the future.

Chapter 2

Related work

The growing importance of BSSs and all issues that might arise from them led to an increasing number of articles about this subject. One of the most studied problems is repositioning. Ghosh and Varakantham [22] categorise the repositioning research into three topics: static, dynamic and user-incentives. This work is mainly focused on the latter but uses static repositioning to compare results obtained from a BSSs simulator that was built for the purpose.

Some of the literature related to repositioning and BSS simulators is presented in this chapter.

2.1 Static

Static repositioning consists of finding the best routes and inventory instructions for a fleet of vehicles, while the system is idle for customers to use, for example, during the night. The goal is to rebalance the system so that every station achieves a pre-determined inventory level. This type of method is usually processed overnight when the movement is negligible [22]. To evaluate formulations, considerations such as objective function, computational running time, number of vehicles, stations and docks, etc., have to be considered. According to the used method to solve the repositioning problem, the available literature can be divided into four types of solution methods: exact algorithm, heuristic or metaheuristic, approximation method and hybrid algorithm.

Exact algorithm

Dell'Amico, Hadjicostantinou, Iori, and Novellani [18] proposed four different Mixed Integer Linear Programming (MILP) formulations, setting as objective function the total travel cost and using as solver a branch-and-cut algorithm. Its best formulation is able to solve “instances with up to 50 vertices” in one hour using a fleet of vehicles. Their study includes data from 22 cities of 7 countries.

Melo [23] based its work on the second formulation of Dell'Amico et al. [18] and built two heuristics to

solve the problem. The first minimised travelled distances and rebalancing operation costs, while the second did not contemplate operation costs. A set of 10 stations from Lisbon BSS (Gira), was used to evaluate the efficacy of the heuristics. Heuristic 1 overcame heuristic 2 by presenting better solutions for low computing times.

Just like Melo [23], Rodrigues [24] based its work on Dell'Amico et al. [18] formulations. This time, the four formulations were evaluated, for 74 stations of Gira. He confirmed the conclusion of Dell'Amico et al. [18] that the third formulation presented, generally, outperforms the remaining. Proposing route length saves between 30% and 75% when compared to the initial route plan.

Most of the articles solve their models considering that, at the end of the repositioning, a predetermined inventory level of bikes at each station should be attained. Erdoğan, Laporte, and Calvo [25], instead of using a target level, considered target inventory intervals, calling the problem: Static Bicycle Relocation Problem with Demand Intervals, “a variant of the One Commodity Pick up and Delivery Travelling Salesman Problem” studied by Hernández-Pérez and Salazar-González [26, 27]. A branch-and-cut algorithm with Benders decomposition is used to solve this problem. Only one vehicle and virtual networks of 30, 40 and 50 stations were considered for the study. Instances take, on average, less than six minutes to be solved.

Approximations

Benchimol, Benchimol, Chappert, De La Taille, Laroche, Meunier, and Robinet [28] presented two solution techniques to solve the pickup-and-delivery problem: a 9.5-approximation algorithm, and a 2-approximation greedy algorithm, respectively meaning that the found solution is 9.5 and 2 times lower than the optimal balancing tour, represented by the optimal value of an integer linear program. The model was designed for a single vehicle with the ability to visit the same station more than once, and the objective of minimising total travel distance. Nevertheless, they do not provide computational results.

Heuristic or metaheuristic:

The objective of Rainer-Harbach, Papazek, Raidl, Hu, and Kloimüller [19] was “minimising the tours’ total duration and the overall number of loading actions”. They received data from Citybike Wien and compared the method GRASP/PILOT (Greedy Randomised Adaptive Search Procedure / Preferred Iterative LOK ahead Technique) hybrid with Variable Neighborhood Search with Greedy Heuristic. Their work considers up to 700 stations repositioned and up to 21 vehicles. The time limit was set to one hour for the largest instances (180 to 700 stations), 30 minutes for medium (60 to 90 stations) and 15 minutes for smaller (15 stations).

Ho and Szeto [29] used iterated tabu search as the metaheuristic solver and as objective a penalty cost function based on inventory levels after repositioning. Their method achieved results in less than a second, deviating less than 0.5% from the optimal when considering less than 100 stations and one

vehicle. The time increased to 8 seconds when considering 400 stations. However, they assumed that the depot could be visited more than once during repositioning, being both the pick up and drop off node, and with sufficient bikes to deal with all demand.

Schuijbroek, Hampshire, and van Hoesel [30] proposed a cluster first route second heuristic and compared it with the Mixed Integer Programming (MIP) and a constraint programming. The objective function consists of minimising the total travelled distance. Their heuristic handles multiples vehicles, which are allowed to visit the same station more than once. It also claims the achievement of getting “In a minute, (...) better solutions for almost all instances than the best MIP solution after 2 hours”. Experiments were carried out with real information from Hubway, in Boston, with 60 stations and 2/3 vehicles (vehicle capacity = 22) to rebalance the system; and Capital Bikeshare in Washington with 135 stations and 5 vehicles (vehicle capacity = 5). Schuijbroek et al. [4] is its recent article update, where loading and unloading time factors were added. This thesis’ work implemented the method of Schuijbroek et al. [4] to get repositioning instructions.

Hybrid (of exact method and heuristic):

Chemla, Meunier, and Calvo [31] combined a branch-and-cut with tabu search. The first solves a relaxation of the problem, and the second obtains an upper bound of the optimal solution. They focus on the single-vehicle problem and define the minimisation of the total travelling distance as a goal. Results show solutions for networks with up to 100 vertices.

A three-step metaheuristic able to solve the problem for multiple vehicles is proposed by Forma, Raviv, and Tzur [32]. The first step clusters stations considering the geographic position and inventory level. The second routes a vehicle through each cluster to obtain inventory targets. The third solves the original repositioning problem, taking into consideration the results from the previous steps. The difference from this method to cluster-first route-second, e.g. Schuijbroek, Hampshire, and Van Hoesel [4], is vehicles’ ability to visit different clusters.

2.2 Dynamic

Static repositioning is a good option to reset a BSS to an ideal state or when stations have low fluctuations throughout the day but performs poorly when the spatio-temporal demand patterns display high variance, easily resulting in full/empty stations [2]. Dynamic repositioning tries to overcome these issues by executing repositioning tasks throughout the day and in real-time.

According to Nath and Rambha [2] research on dynamic repositioning is broken down into two approaches. The first considers the operating period is divided into a finite number of time steps, assuming time-varying demand as already known. This is an extension of static repositioning formulations but repeated along the day [2]. An example of it is Ghosh, Varakantham, Adulyasak, and Jaillet [33] which proposed an optimisation formulation for this problem setting as objective the reduction of lost customer

demand; The second approach breaks the overall problem into multiple dynamic rebalancing problems, using a rolling horizon method. For every interval of time, demand is observed, forecasts are updated for the next time interval and, finally, the rebalancing problem is solved based on new data. [2]. Shui and Szeto [34] used the last approach. Their objective function was to reduce total unmet demand, fuel and CO_2 emissions of vehicles over the operational period.

2.3 User incentives

This category is characterised by the help of customers on stations' rebalancing operations, in exchange for an incentive. Price incentives are the most used approach, but there are other options like promotions, points, extra minutes, etc.

Chemla, Meunier, Pradeau, Calvo, and Yahiaoui [7], for example, use an open-source simulator with freely set parameters to evaluate and compare several heuristics with the intention of solving the dynamic problem (single-vehicle), which they proved to be NP-hard (Non-deterministic Polynomial-time hard). A dynamic pricing strategy is also used to influence the users' decisions in its ending station; and compared against heuristics. Low, medium and high demand, as well as different network sizes, were compared, in four hours of simulation.

Fricker and Gast [20] propose a stochastic model of a homogeneous bike sharing system; in other words, all stations have the same capacity and demand rate. They analyse the influence of a two-choice model. Users pick up two possible final destinations, chosen at random, and are influenced to return the bike to the emptiest station. According to them, the state of the network was improved by an exponential factor, with these incentives.

Unlike Chemla et al. [7] or Fricker and Gast [20], Pfrommer, Warrington, Schildbach, and Morari [5] created a model based on historical data from London Cycle Hire. A tailored routing algorithm was computed to redistribute bikes with multiple vehicles. In addition, the authors also created a dynamic price incentive-based Model Predictive Control with the objective of optimising operating costs for a certain service level. Results have shown that considering only price incentives without staff redistribution, service level could be kept above 87% on weekends. During workdays, especially during rush hours, price incentives were insufficient.

A dynamic pricing mechanism, using the approach of regret minimisation in online learning, is used by Singla, Santoni, Bartók, Mukerji, Meenen, and Krause [35] to obtain optimal pricing policies. Surveys asking people for their preferences about private cost and walking distance were conducted to validate their assumptions. After simulations, using historical data from Boston's Hubway, an experimental deployment was performed in a European city for thirty days, resulting in an acceptance rate of 60%.

Aeschbach, Zhang, Georghiou, and Lygeros [6] focused on customer's ability to balance the system, proposing four different control strategies, where the way they interact with customers is what distinguishes them. They intended to evaluate strategies' efficacy by treating CC as a variable parameter;

this way, different percentages of cooperation give different service levels. They found that a CC of 50% with a neighbourhood Radius (Rad) of 700 meters is enough to balance London's Barclays Cycle Hire (nowadays, Santander Cycle Hire). Part of their work is applied and experimented in this thesis.

Ghosh and Varakantham [22] take a different path to solve the problem, where payments are the given incentive. First, they generate potential repositioning tasks based on a mixed integer linear optimisation and then incentive people (using payment/trip-based incentives) to execute these tasks by attaching bike trailers to their bikes. The percentage of users interested in trailer tasks was varied to observe the effects on lost demand. They carried experiments with data from Hubway BSS (Boston).

Haider, Nikolaev, Kang, and Kwon [36] named its heuristic approach: Iterative Price Adjustment Scheme (IPAS). The goal is to incentivise people to take bikes from or park them at imbalanced stations to make them more imbalanced, therefore creating hub stations. This way, the number of stations visited by trucks to carry inventory repositioning are reduced. Results were obtained with data from Capital Bikeshare in Washington, D.C.

Pan, Cai, Fang, Tang, and Huang [37] formulated the problem as a Markov Decision Process with the objective of maximising service level. A deep reinforcement learning algorithm called Hierarchical Reinforcement Pricing was proposed to decide how to pay people. Experiments were conducted using Mobike's (Beijing, China) dataset and compared with other state of the art methods like Singla et al. [35].

Patel, Qiu, and Negahban [38] developed a customised station object in Simio simulation software where incentives are given to customers willing to pick up bikes from stations that will soon become full. They used experimental data from CitiBike system in Jersey City and results showed that incentives help improve bike availability and service level. Nevertheless, total profit can be reduced if the discount offered is excessively increased.

2.4 Simulators

Given that BSS simulators are a recent trend, there is not much literature on the topic, nevertheless, in the last few years, their popularity increased [39].

The most common objective presented in simulators' literature is to evaluate repositioning schemes and Caggiani and Ottomanelli [40] is one of those examples. They propose a method where it is assumed that an operating day is divided into discrete time intervals. At the beginning of each interval, the network is updated, and trips are generated "based on relative O-D (Origin-Destination) attractiveness". People who can not find bikes at the departure station are removed and those who arrive at a full station must wait until there is a free dock.

Chemla et al. [7] created the OADLIBSim, a discrete-event open-source simulator programmed in C++. The purpose of this simulator was to evaluate dynamic rebalancing algorithms along with dynamic pricing. Users arrive at the initial station i following a Poisson distribution, being drawn at random to destination j according to a probability $p_{i,j}$. Users facing empty stations on the departure and full stations on

arrival choose their next destination minimising walking and riding distance. To compare strategies, the number of satisfied users is taken into consideration to measure quality.

Ji, Cherry, Han, and Jordan [41] developed a Monte Carlo simulation model with the objectives of simulating the operations of an e-bike sharing system and evaluate how many bikes and batteries are needed to meet demand and recharge rates. Trips generation rate followed a Poisson distribution and riding speed a Normal distribution. Parameters like the number of swappable batteries and battery recharging profiles were also considered.

Dubernet and Axhausen [42] evaluated bike redistribution systems with a simulation tool, built on the multi-agent simulation framework MATSim. Customers behaviours are modelled in line with their daily plans.

Saltzman and Bradford [43] applied Arena 14's software package to build a detailed animated model of San Francisco BSS. They investigated system performance when changing daily initial conditions, such as the number of bikes and docks. Trips are created every hour following a Poisson process, and trip duration a Lognormal distribution. Riders arriving at a full station wait 2 minutes for an empty dock; if, even so, no bike leaves the station, they ride to the nearest station with available docks.

Jian, Freund, Wiberg, and Henderson [44] tried to minimise the number of unsatisfied customers by optimising both bike and dock allocations. Their approach is based on a discrete-event simulation model that operates every minute.

MATLAB was the software chosen by Soriguera, Casado, and Jiménez [45] to implement their agent-based simulation model. By changing system parameters and daily operations, they intend to reach the best system design. Unlike the model presented in this thesis, they consider electric and traditional bikes. This approximation to reality, however, increases its dimension and complexity. Users facing a no-service event at departure, i.e. no bike available, travel to the closest station with available bikes and those who do not use the app travel to the closest station, independently of their level. No-service event at arrival takes all users to the closest station with free spots.

Fernández, Billhardt, Ossowski, and Sánchez [39] built a simulator called Bike3S, again, an agent-based simulator. Their tool is designed to test different station capacities, station distributions, and balancing strategies. In contrast with most works, exception made for Saltzman and Bradford [43], this simulator presents a visual interface and allows several configurations to be altered according to preference.

Summarising, simulators are used for modelling, evaluate truck-based rebalancing approaches or user incentive-based approaches. The majority of articles addressing this last topic use simulators to obtain results, e.g. Pfrommer et al. [5], Chemla et al. [7], Waserhole and Jost [46]. Almost all of them generate trips with Poisson probability distributions and only a few show a visual interface.

Chapter 3

Theoretical background

This chapter will present the methods chosen to implement in the model, as well as the theory behind them.

3.1 Trip generation and trip time

To generate trips, the work of Pfrommer et al. [5] was followed. The authors start by defining set S as containing all stations $s \in S$, and by distinguishing the days of the week between workdays and weekend days, according to the binary variable $w \in \{workday(wd), weekend(we)\}$. Each day is divided into 72 time slices $t \in T := \{1, \dots, 72\}$ of 20 minutes each. The authors also define that the number of trips, for each $\{\text{week day type } (w), \text{departure station } (i), \text{arrival station } (j), \text{time slice } (t)\}$ relation, follows a Poisson distribution with mean $\lambda_w(i, j, t)$. These mean values are empirically determined by accessing available data and computing the average number of journeys of every (w, i, j, t) relation. An extensive explanation of this approach can be found in section 5.2.1.

After trip generation it is necessary to compute the trip time. Pfrommer et al. [5] used the average trip time between stations (i, j) from historic data to compute these values. In this thesis, trip time is computed based on a Lognormal distribution. Therefore, given that trip generation makes use of two distribution functions, Poisson and Lognormal, their theory is presented below.

3.1.1 Poisson distribution

It was Siméon Denis Poisson, a French mathematician, to give his name to this distribution, which is a discrete probability distribution that describes the probability of the number of times an event can occur, according to a determined mean rate for a determined interval of time or space [47].

A discrete random variable X , follows a Poisson distribution of parameter $\lambda > 0$, if for $k \in [0, \infty]$ the probability mass function of X is given by [48]:

$$f(k; \lambda) = P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k \in \mathbb{N}_0 \quad (3.1)$$

where

- e is Euler's number ($e = 2.71828\dots$),
- k is the number of occurrences,
- $k!$ is the factorial of k .

This distribution is characterised by an expected value and variance equal to λ [49, 50].

To estimate parameter λ given a sample of n measured values $k_i \in \{0, 1, \dots\}$, for $i = 1, \dots, n$, the following maximum likelihood estimate is used [51]:

$$\hat{\lambda}_{MLE} = \frac{1}{n} \sum_{i=1}^n k_i. \quad (3.2)$$

Some examples of Poisson distributions are presented in figure 3.1.

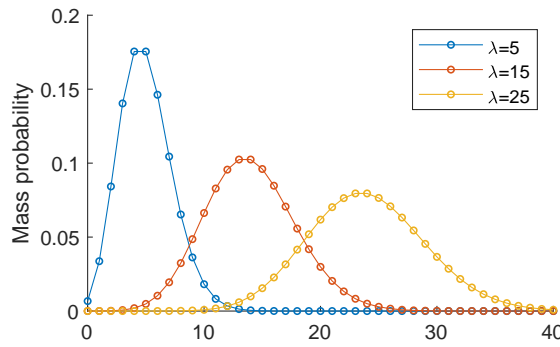


Figure 3.1: Poisson distributions for $\lambda = 5, 15$ and 25

3.1.2 Lognormal distribution

"More than a century ago, Galton [52] pointed out that if X_1, X_2, \dots, X_n are independent positive random variables and $T_n = \prod_{i=1}^n (X_i)$, then $\log(T_n) = \sum_{i=1}^n \log(X_i)$ when appropriately standardised for mean and variance, will tend to a standard normal variable as $n \rightarrow \infty$. Then, the limiting distribution of the variable T_n would be Lognormal (Log-normal or Galton). In a follow-up article, McAlister [53] derived expressions for the mean, median, mode, variance and some percentiles of the Lognormal distribution." [54].

"If Y is normally distributed with mean μ and variance σ^2 , then the random variable X defined by the relationship $Y = \log(X - \gamma)$ is distributed as a Lognormal and denoted as $\text{lognormal}(\gamma, \mu, \sigma^2)$ ", where γ is the distribution minimum value. With a transformation of random variables, the density function of a Lognormal random variable X is obtained [54]:

$$f(x; \gamma, \mu, \sigma^2) = E(X = x) = \frac{1}{\sqrt{2\pi}\sigma(x-\gamma)} e^{-(\log(x-\gamma)-\mu)^2/(2\sigma^2)}, \quad \gamma < x < \infty \quad (3.3)$$

Its mean and standard deviation were derived by Yuan [55], as follows:

$$\bar{X} = \gamma + \beta\sqrt{k}, \quad \bar{\sigma}_X = \beta\sqrt{k(k-1)} \quad (3.4)$$

where, $k = e^{\sigma^2}$ and $\beta = e^\mu$.

Figure 3.2 presents examples of Lognormal distributions for different values of μ and σ , not considering γ impact.

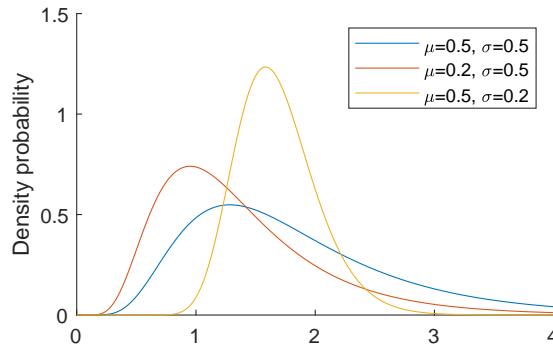


Figure 3.2: Lognormal distributions

To estimate μ and σ given a sample X of n measured values $x_i \in \mathbb{R}^+$ and $\gamma = 0$ the following expressions are used [56]:

$$\hat{\mu} = \frac{\sum_{k=1}^n \ln x_i}{n}, \quad \hat{\sigma} = \sqrt{\frac{\sum_{k=1}^n (\ln x_i - \hat{\mu})^2}{n}} \quad (3.5)$$

3.2 User behaviour and incentives model

To model user behaviour and present alternative stations to users that accept to cooperate in BSS balancing, the work of Aeschbach et al. [6] was followed, which considers the concept of neighbouring stations given a certain radius.

Below, the four strategies evaluated are explained:

- **No Control (NC):** A customer facing an empty station on departure walks to the nearest station; if it is also empty, leaves the system. On the other hand, a customer arriving to a full station rides to the nearest station, and, so on and so forth, until they find an empty dock.
- **Minimal Intervention Control (MIC):** Customers experiencing a full/empty station on arrival/ departure are prompted to ride to the neighbouring station with the most empty/full slots. In case

there are no neighbouring stations filling the requests, the customer is sent to the closest station, regardless of its level.

- **Preemptive Control via Mobile App (PC-MA):** This control demands direct communication between the customer and the app before the journey starts. The customer must register in the app what is the journey he intends to do, and the app outputs the station in the neighbourhood of the intended departure with more bikes, as well as the station in the neighbourhood of the intended arrival with fewer bikes. Additionally, the user gets to decide whether he accepts or not, randomly and according to a CC ratio. In the case of a no-service event, he follows the same procedure as in MIC.
- **Weak Preemptive Control via Mobile App (WPC-MA):** This strategy's difference to PC-MA is that, whenever a customer inputs their desired stations, if the occupation percentage is above or below 50% the app tells him to use that same station.

To evaluate these strategies, the authors used the concepts of “empty-station event” and “full-station event”, the first meaning a person trying to rent a bike from an empty station and, the second, that a person trying to return a bike without empty docks. Both events are referred to as “no-service events”. The quality of the service is measured by the service level, also used in the works of Pfrommer, Warrington, Schildbach, and Morari [5] and Singla, Santoni, Bartók, Mukerji, Meenen, and Krause [35]:

$$Service\ Level = \frac{\#Customers - \#No\ Service}{\#Customers} \quad (3.6)$$

To measure the average combined distance at start and end caused by the changed journey, Aeschbach et al. [6] used the average extra effort:

$$Extra\ Effort\ (km) = \frac{\sum_{(s_{out}, s_{in})} [d(s_{out}, \bar{s}_{out}) + d(s_{in}, \bar{s}_{in})]}{\# Customers\ (with\ positive\ effort)} \quad (3.7)$$

where $d(s_{out}, \bar{s}_{out})$ is the distance between the starting station chosen by the customer (s_{out}) and the starting station chosen by the control strategy (\bar{s}_{out}); and $d(s_{in}, \bar{s}_{in})$ is the distance between the ending station chosen by the customer (s_{in}) and the ending station chosen by the control strategy (\bar{s}_{in}). To clarify, for this measure, only users who have decided to participate, and have actually changed their journey, were considered, users who participated, but the result of the control strategy were the same stations, were not considered.

3.3 Repositioning problem

To check the influence of user incentives on repositioning, the heuristic of Schuijbroek et al. [4] was implemented. Gira operates a dynamic repositioning; however, we did not have access to this algorithm, so an alternative had to be found. Since the traced objective is to evaluate the performance of user incentives and not the efficacy of repositioning methods, it was decided to apply an approach that fulfils its

purpose and does not take too much computational effort, as it is the case of static repositioning, which is able to reset the BSS every day. Schuijbroek et al. [4] solve the repositioning problem with a cluster first route second heuristic and target intervals s_i^{min} and s_i^{max} . Their intention is to split the problem into smaller single-vehicle routing problems, to reduce the complexity of the problem and achieve results in a much smaller interval of time.

In the following sections, the clustering and routing problem formulations are presented, along with the MIP heuristic algorithm.

To compute target interval levels, s_i^{min} and s_i^{max} , the reasoning of Hulot, Aloise, and Jena [57], who applied Skellam distribution for this purpose, was used. Therefore, the theory about this Skellam distribution is also presented.

Below, the list of nomenclature related to clustering and routing formulations are introduced:

Indexes

i, j	Vertexes
t	Time step
v	Vehicle

Sets

S	Set of stations
N	Set of stations with artificial vertex 0
T	Set of time
V	Set of vehicles

Parameters

s_i^0	Initial inventory of station $i \in S$
s_i^{min}	Minimum required number of bikes of station $i \in S$
s_i^{max}	Maximum required number of bikes of station $i \in S$
C_i	Capacity of station $i \in S$
$d_{i,j}$	Distance between station $i \in S$ and $j \in S$
Q_v	Capacity of vehicle $v \in V$
q_v^0	Initial inventory of vehicle $v \in V$
d^-	Routing costs of picking up one bike
d^+	Routing costs of delivering one bike
s_i^+	Minimum number of bike deliveries at station $i \in S$
s_i^-	Minimum number of bike pick ups at station $i \in S$

Decision Variables

$x_{i,j,t,v}$	Binary decision variable to indicate whether vehicle $v \in V$ traverses arc (i, j) in time $t \in T$
$y_{i,t,v}^-$	Decision variable to indicate amount of bikes to pick up, by vehicle $v \in V$
$y_{i,t,v}^+$	Decision variable to indicate amount of bikes to deliver, by vehicle $v \in V$
$z_{i,v}$	Binary decision variable to indicate bike assignment of station $i \in S$ to vehicle $v \in V$
h_v	Auxiliary decision variable to indicate routing costs

3.3.1 Routing problem

The objective of a routing problem is to identify repositioning instructions vehicles should follow to redistribute the network, according to predefined station occupation targets. Therefore, the resulting decision variables must show the sequence of stations to visit ($x_{i,j,t,v}$) and the amount of bikes to drop ($y_{i,t,v}^+$) or pick up ($y_{i,t,v}^-$) in every visited station. Routing MIP formulation is presented below:

$$\text{Minimise : } H \quad (\text{P1})$$

subject to :

$$s_i^0 + \sum_{\substack{t \in T \\ v \in V}} (y_{i,t,v}^+ - y_{i,t,v}^-) \geq s_i^{\min} \quad \forall i \in S \quad (3.8)$$

$$s_i^0 + \sum_{\substack{t \in T \\ v \in V}} (y_{i,t,v}^+ - y_{i,t,v}^-) \leq s_i^{\max} \quad \forall i \in S \quad (3.9)$$

$$\sum_{\substack{i \in S \\ j \in N}} x_{i,j,1,v} \leq 1 \quad \forall v \in V \quad (3.10)$$

$$\sum_{j \in N} x_{i,j,t,v} \leq \sum_{j \in S} x_{j,i,t-1,v} \quad \forall i \in S, t \in T \setminus \{1\}, v \in V \quad (3.11)$$

$$\sum_{\substack{i \in S \\ t \in T \\ v \in V}} x_{i,i,t,v} = 0 \quad (3.12)$$

$$y_{i,t,v}^- \leq Q_v \sum_{j \in N} x_{i,j,t,v} \quad \forall i \in S, t \in T, v \in V \quad (3.13)$$

$$y_{i,t,v}^+ \leq Q_v \sum_{j \in N} x_{j,i,t,v} \quad \forall i \in S, t \in T, v \in V \quad (3.14)$$

$$\sum_{\substack{t \in T \\ v \in V}} y_{i,t,v}^- \leq s_i^0 \quad \forall i \in S \quad (3.15)$$

$$\sum_{\substack{t \in T \\ v \in V}} y_{i,t,v}^+ \leq C_i - s_i^0 \quad \forall i \in S \quad (3.16)$$

$$q_v^0 + \sum_{\substack{i \in S \\ \tilde{t} \in T: \tilde{t} \leq t}} (y_{i,\tilde{t},v}^- - y_{i,\tilde{t},v}^+) \geq 0 \quad \forall t \in T, v \in V \quad (3.17)$$

$$q_v^0 + \sum_{\substack{i \in S \\ \bar{i} \in T: \bar{i} \leq t}} (y_{i,t,v}^- - y_{i,t,v}^+) \leq Q_v \quad \forall t \in T, v \in V \quad (3.18)$$

$$h_v = \sum_{\substack{i,j \in S \\ t \in T}} d_{i,j} x_{i,j,t,v} + \sum_{\substack{i \in S \\ t \in T}} d^- y_{i,t,v}^- + \sum_{\substack{i \in S \\ t \in T}} d^+ y_{i,t,v}^+ \quad \forall v \in V \quad (3.19)$$

$$H \geq h_v \quad \forall v \in V \quad (3.20)$$

$$x_{i,j,t,v} \in \{0, 1\} \quad \forall i \in S, j \in N, t \in T, v \in V \quad (3.21)$$

$$y_{i,t,v}^-, y_{i,t,v}^+ \in \mathbb{N}_0 \quad \forall i \in S, t \in T, v \in V \quad (3.22)$$

$$H, h_v \geq 0 \quad \forall v \in V \quad (3.23)$$

Constraints (3.8) and (3.9) impose stations to be within target intervals s_i^{min} and s_i^{max} .

Constraints (3.10) to (3.14) handle vehicle routing. (3.10) does not allow vehicles to have more than one route each. Constraint (3.11) guarantees that there are no bikes lost. Constraints (3.12) prevent dwelling. Constraints (3.13) and (3.14) make sure that deliveries and pick ups only occur when arriving or leaving a station, respectively. (3.15) and (3.16) constraint the amount of transshipments. (3.17) and (3.18) ensure that the vehicles do not travel with more bikes than their capacity or with negative values.

Each vehicle routing costs are defined by constraint (3.19), adding total distance, loading and unloading costs. Constraints (3.20) to (3.23) linearise the objective $H = \max_{v \in V} h_v$. In order to uniformise the solutions, only the maximum value is considered.

3.3.2 Clustering problem

Grouping stations into clusters can be done in many ways. The most obvious would be to cluster them by distance. This strategy by itself would not solve the problem for a BSS application, considering that it could join all the stations in need of bikes or with excess. The alternative presented by Schuijbroek et al. [4] introduces the Maximum Spanning Star approximation, denoted by $\max_{i \in S_v} \sum_{j \in S_v} d_{i,j}$, and takes into consideration (un)loading time approximations. The MIP formulation to solve this problem is presented below:

Minimise: \hat{H}

subject to:

$$\sum_{v \in V} z_{i,v} = 1 \quad \forall i \in S \setminus S_0 \quad (3.24)$$

$$\sum_{v \in V} z_{i,v} \leq 1 \quad \forall i \in S_0 \quad (3.25)$$

$$q_v^0 + \sum_{v \in V} s_i^0 z_{i,v} \geq \sum_{i \in S} s_i^{min} z_{i,v} \quad \forall v \in V \quad (3.26)$$

$$-(Q_v - q_v^0) + \sum_{i \in S} s_i^0 z_{i,v} \leq \sum_{i \in S} s_i^{max} z_{i,v} \quad \forall v \in V \quad (3.27)$$

$$\hat{h}_v \geq \sum_{j \in S} d_{i,j} (z_{i,v} + z_{j,v} - 1) + \sum_{j \in S} (d^+ s_j^+ + d^- s_j^-) z_{j,v} \quad \forall i \in S, v \in V \quad (3.28)$$

$$\hat{h}_v \geq \sum_{j \in S} d_{i,j} (z_{i,v} + z_{j,v} - 1) + \sum_{j \in S} (d^+ s_j^+ + d^- s_j^+) z_{j,v} - d^- q_v^0 \quad \forall i \in S, v \in V \quad (3.29)$$

$$\hat{h}_v \geq \sum_{j \in S} d_{i,j} (z_{i,v} + z_{j,v} - 1) + \sum_{j \in S} (d^+ s_j^- + d^- s_j^-) z_{j,v} - d^+ (Q_v - q_v^0) \quad \forall i \in S, v \in V \quad (3.30)$$

$$\hat{H} \geq \hat{h}_v \quad \forall v \in V \quad (3.31)$$

$$z_{i,v} \in \{0, 1\} \quad \forall i \in S, v \in V \quad (3.32)$$

$$\hat{H}, \hat{h}_v \geq 0 \quad \forall v \in V \quad (3.33)$$

with

$$s_i^+ = \max\{s_i^{min} - s_i^0, 0\} \quad \forall i \in S \quad (3.34)$$

$$s_i^- = \max\{s_i^0 - s_i^{max}, 0\} \quad \forall i \in S \quad (3.35)$$

Cluster feasibility is guaranteed by constraints (3.24) to (3.27), with constraint (3.24) ensuring that insufficient stations are visited and (3.25) allowing self-sufficient stations to be visited. Taking into consideration truck maximum capacity, minimum and maximum inventory targets are imposed by equation (3.26) and (3.27), respectively. Constraints (3.28), (3.29) and (3.30) impose minimum required bike deliveries and pick ups by deriving lower bounds on the (un)loading times. With constraint (3.29) dealing stations with more deliveries and (3.30) with more pick ups. Constraints (3.31) to (3.33) linearise makespan $\hat{H} = \max_{v \in V} \hat{h}_v$.

However, we observe that (P1) is practically intractable for realistic station sets with $|S| \geq 50$ and vehicle fleets with $|V| \geq 3$.

3.3.3 Heuristic MIP

After presenting clustering and routing methodologies, the heuristic algorithm will follow the work of Schuijbroek et al. [4]. The algorithm has the following sequence:

1. Solve the Clustering Problem (P2).
2. For each $v \in V$ solve the Routing Problem (P1) with $S = S_v$ and $V = \{v\}$ to obtain $H^*(S_v, \{v\})$, which is the optimal solution for P1.
3. $\tilde{H} = \max_{v \in V} H^*(S_v, \{v\})$.

3.3.4 Skellam distribution

Skellam distribution emerged in 1946 with Skellam [58]. This is a discrete probability distribution which corresponds to the difference of two statistically independent random variables X_1 and X_2 both having both a Poisson distribution with parameters $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$, respectively [59]. Considering $K = X_1 - X_2$, the expression for this distribution is [59]:

$$f(k; \lambda_1, \lambda_2) = P(K = k) = e^{-\lambda_1 - \lambda_2} \left(\frac{\lambda_1}{\lambda_2} \right)^{k/2} I_k(2\sqrt{\lambda_1 \lambda_2}) \quad (3.36)$$

where $I_k(z)$ is the modified Bessel function.

Figure 3.3 shows an example of how two stochastic variables give rise to a Skellam distribution.

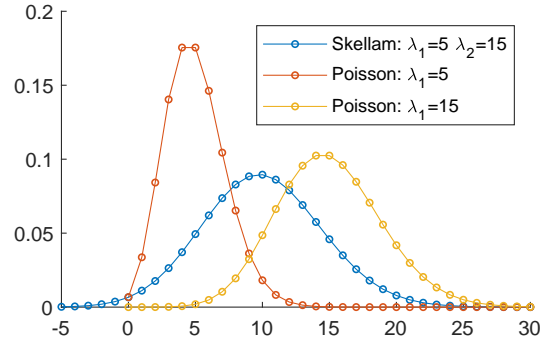


Figure 3.3: Skellam distribution

Chapter 4

Gira BSS characterization

Gira is Lisbon's BSS [60]. It belongs to EMEL - Empresa Municipal de Mobilidade e Estacionamento de Lisboa had its first tests on the 21st of June of 2017 and was officially launched on the 19th September 2017, in *Parque das Nações*, with 10 stations and a capacity of 100 bicycles [61, 62]. EMEL intends to "contribute for an increasing life quality of residents and visitors" by transforming "Lisbon into a more accessible city, less polluted, and much less stressful" [60, 61]. Currently (July 2021), there are 60 km of bike lanes available in Lisbon, but according to Gira's website, this number will "soon" be increased to 150 km, allowing the expansion and safety of the network [60].

On the 4th of February 2019 a protocol was signed between IDMEC and EMEL, giving IDMEC access to Gira BSS data, with the objective of promoting studies about urban mobility, as well as new technologies and parking operation methods. The present work is one of those studies, with data from January to March of 2019. In this chapter, geographical distribution, as well as other characteristics of the network, are presented.

4.1 Geographical distribution and flow intensity

From January to March 2019, the network had a total of 74 stations, each one composed of several docks, and illustrated in figure 4.1. There are also hundreds of bikes both in the depot and on the streets.

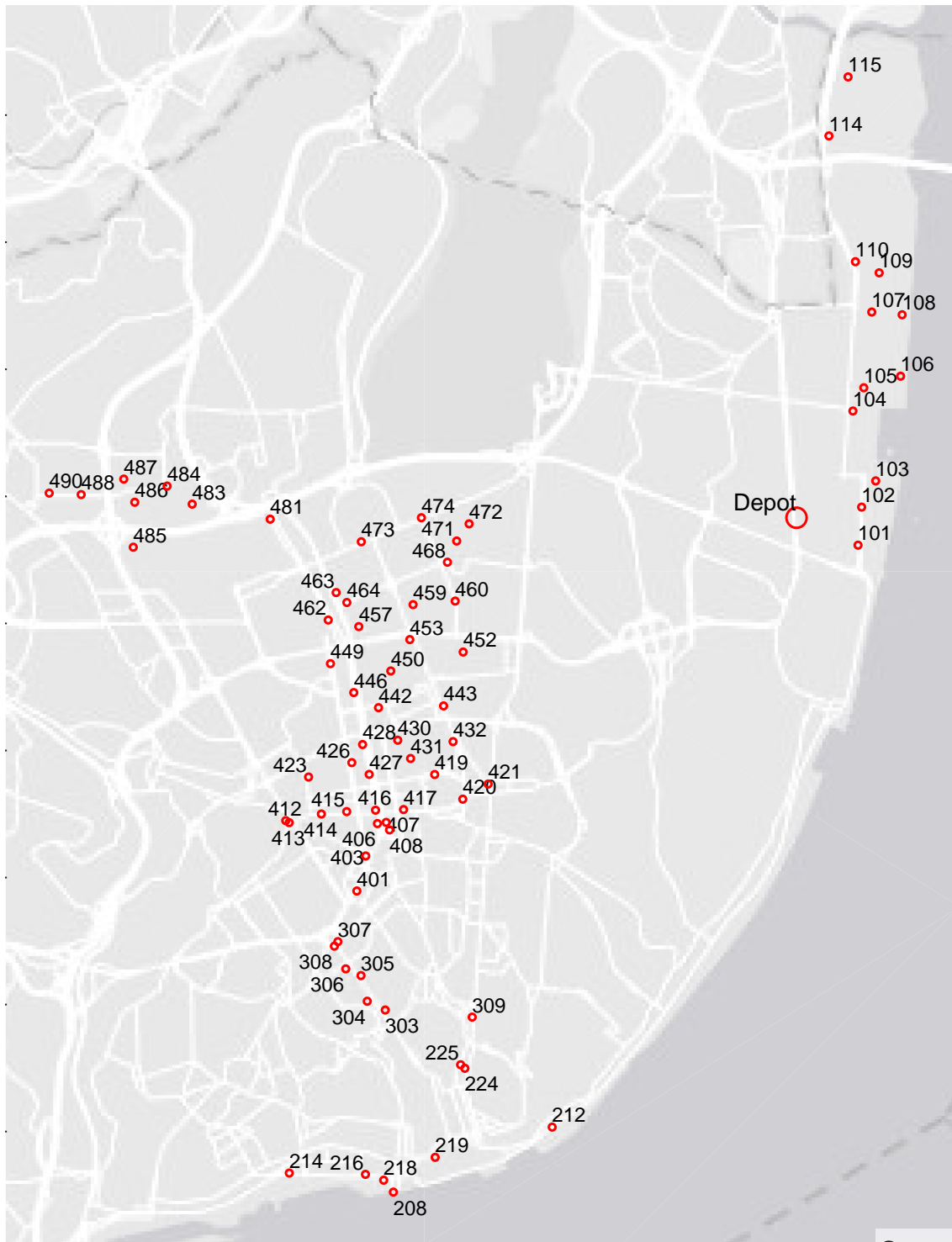


Figure 4.1: Network geographic distribution (March 2019)
Sources: Esri, DeLorme, HERE, MapmyIndia, Garmin, FAO, NOAA, USGS

In order to meet the demand while saving resources, stations have a different number of docks. The most requested stations have normally 40 docks and the less requested 10 or more. This number may vary over time, for example, due to broken docks that need to be fixed. These changes were identified and plotted in figure 4.2, where, for example, illustrated station 105 appears to be constantly changing the number of available docks between 37 and 40, and the station 472 shows its number of docks

reduced only for a few days.

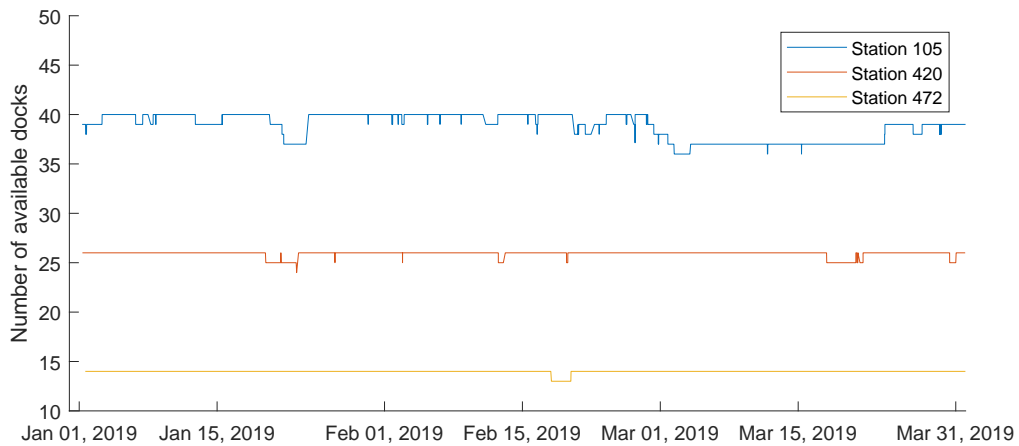


Figure 4.2: Number of available docks in stations 105, 420 and 472

Just like the number of docks, due to failures, thefts, cleaning, repositioning and other types of problems, the number of available bikes in the system is always changing. These changes are illustrated in figure 4.3 where the number of bikes at the beginning of each day (bikes can be rent from 6 AM to 2 AM of the next day) is constantly varying, but also keeping, more or less, the number of bikes in the range of 600 to 750 in January and February. These values drop to 450, in March, even though the number of trips maintained its level (see figure 4.4).

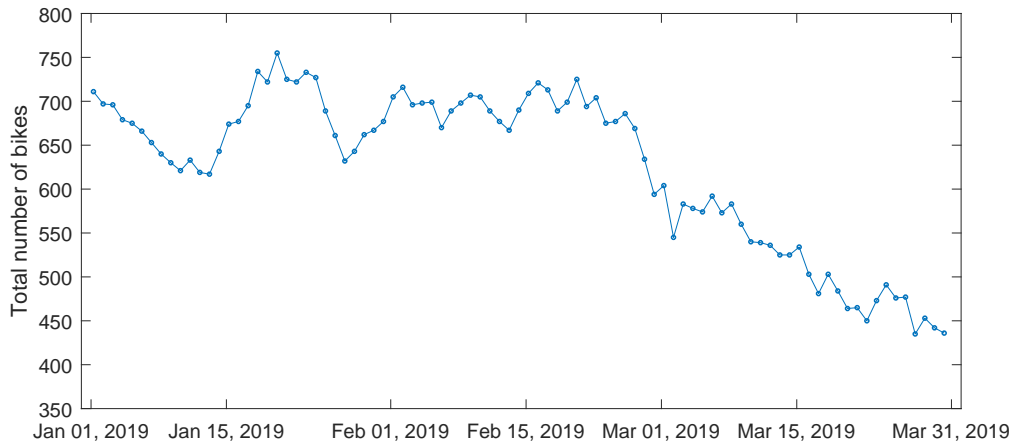


Figure 4.3: Number of available bikes in the network at 6 AM (start of operation)

With a simple look at the network distribution (see figure 4.1) and a proper analysis of customer behaviour (see figure 4.5) it is possible to claim that this BSS could easily be divided into two major zones, the centre of the city, including the downtown; and *Parque das Nações*, particularly with station 104 serving as the link to the centre of the city and also all of the others linking the downtown. When it comes to the centre zone, station 446 is the most requested, connecting every part of the city.

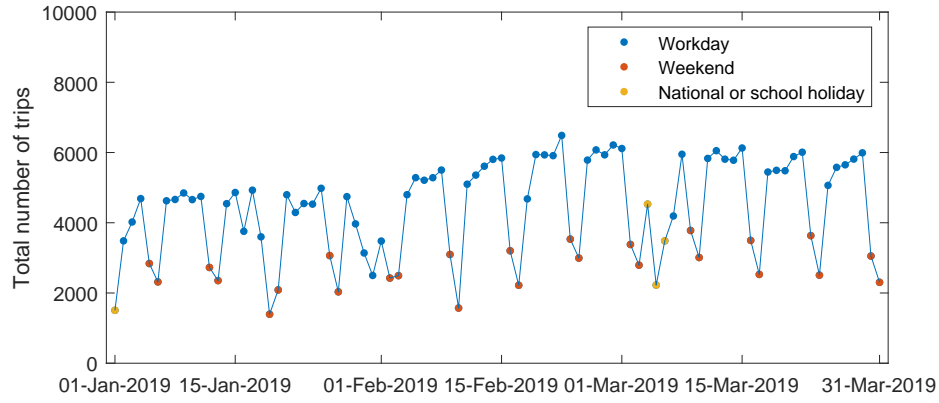


Figure 4.4: Total number of trips per day

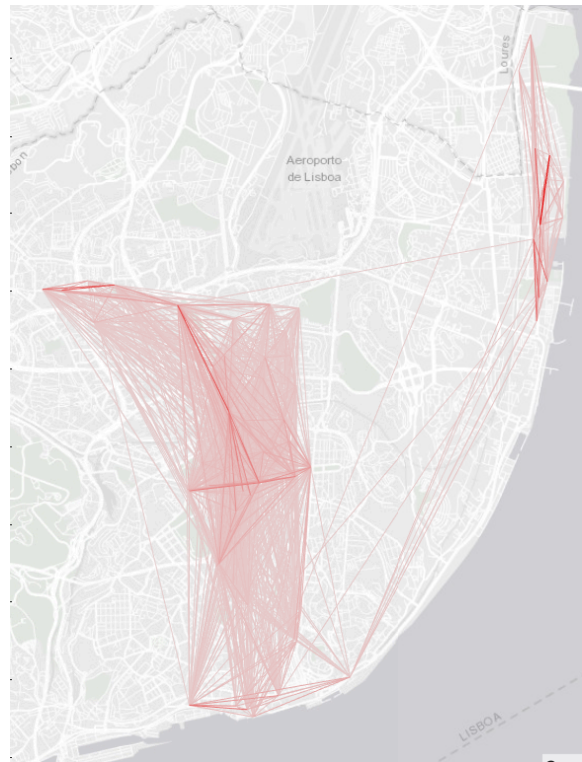


Figure 4.5: Trip intensity map

Rain, social events, holidays, among other factors can increase or decrease the number of trips during a day - Lucas and Andrade [63] addressed the weather effect when estimating hourly origin-destination demand in Lisbon BSS - however, some patterns repeat over and over again. The most obvious is shown in figure 4.4 with a considerable difference between workdays when there is a lot of movement and traffic inside the city, reflected in more trips; and weekends with much less traffic, due to less movement. This difference will also be presented through the daily pattern in section 4.2. It must also be highlighted, that during the holiday of the 1st of January and the Carnival week starting on the 4th of March, the number of trips dropped when compared to the other workdays.

4.2 Temporal patterns

Temporal patterns seem to preserve their shape in many BSSs, with two well-identified peaks of traffic during workdays, one in the morning when people arrive at their jobs, and one in the afternoon, slightly bigger, when they leave them. There is also a lunchtime curve that tends to be small in most BSSs. At the weekend, the pattern changes to a convex curve with approximately the same level of trips, depending on the BSS. Some examples are Chicago, Seattle and Paris BSSs [64–66].

Lisbon BSS, just like the others, is not an exception to the typical city patterns. Figure 4.6 displays the average number of trips for all workdays at 20 minute time intervals. As we can see, for all 5 workdays the pattern is quite similar, with the two regular peaks: the morning peak starting at 7:30 AM which has its maximum at approximately 9 AM; the one in the afternoon starting at 3 AM and peaking at 6 PM. Another typical peak is the lunchtime peak, which reaches its maximum at 1:30 PM. Throughout the weekend the number of trips increases until 4 PM and from there on, starts to decrease, with Saturday surpassing Sunday by a little.

Since temporal patterns were observed, figure

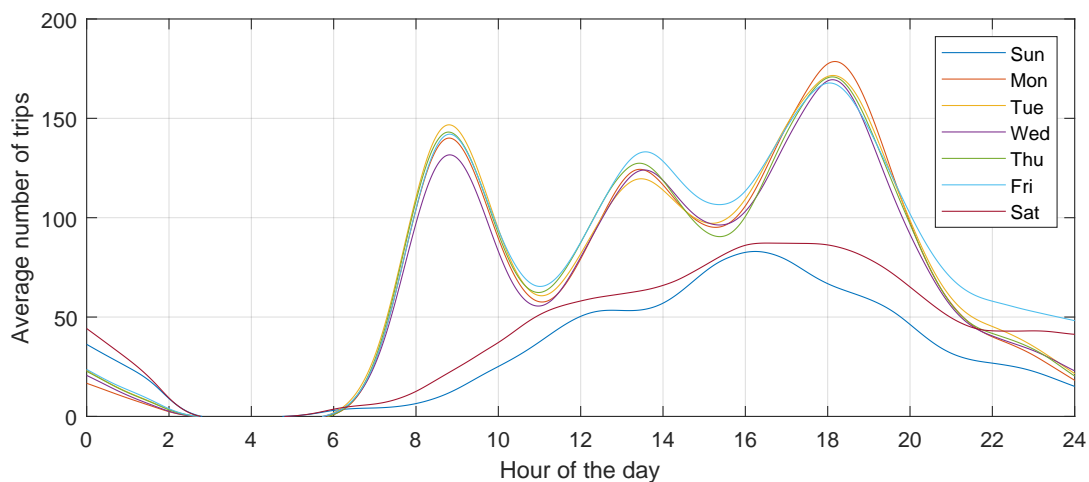


Figure 4.6: Average number of trips throughout a day for all weekdays (20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

Customers daily patterns can also be identified in figure 4.7 through their preferable arrival and departure stations, for different time intervals, and workdays versus weekends. This graph plots the difference between arrivals and departures at each station for different time intervals and days of the week, where an orange bubble represents a station with a tendency to lose bikes and a blue bubble a station with a tendency to gain bikes. In the centre, during workdays from 6 to 11 AM, people flow preferably from *Entrecampos*, which has metro and train stations, to *Saldanha*, *Marquês de Pombal*, and *Avenida da Liberdade*. In *Parque das Nações* there are stations near metro and train stations, which usually have a higher arrival flow of bikes when compared to the departure flow. This pattern reverses its direction in the afternoon peak. During the lunch break, peak bubbles are smaller; this does not mean that there is less flow: it means the stations are more likely balanced. This balance is also present at night when

there is a lower trip flow. Throughout the weekend, patterns of the first and second half look like the ones presented by workdays in the morning and afternoon, respectively, but with a lower intensity.

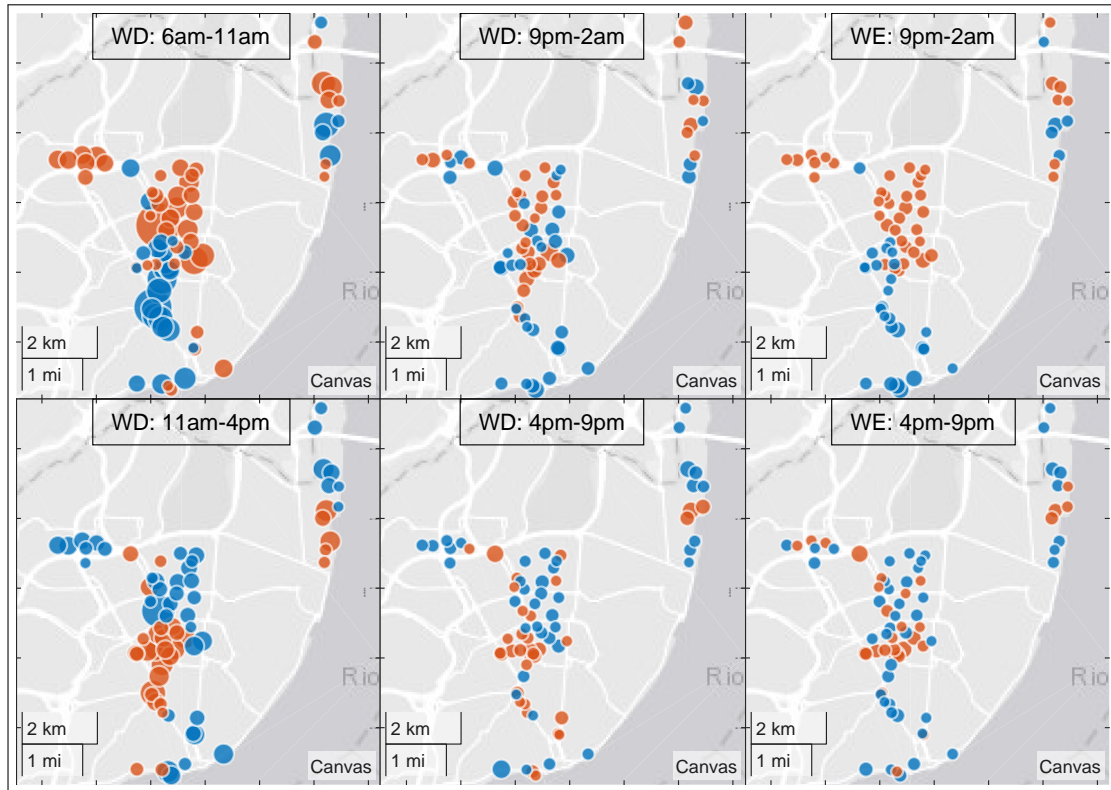


Figure 4.7: Bubble plot of station difference between arrivals and departures (negative difference-orange; positive difference-blue) Workday (WD): 6AM-11AM (top-left); 11AM-4PM (top-center); 4PM-9PM (bottom-left); 9PM-2AM (bottom-center). Weekend (WE): 6AM-4PM (top-right) and 4PM-2AM (bottom-right).

To give us a better understanding of the network during workdays when stations variance is prominent, an explanation of the more relevant station behaviour will be given below, looking at the most used stations. This explanation has already been provided by Rodrigues [24] in his thesis, in which he also makes an analysis of Gira. Therefore, some of the information obtained in its work will be projected here with an added month of data. He considered that the categorisation criteria of daily usage patterns used by Dell'Amico et al. [18]. This criterion groups bikes into one of three categories:

1. Stations with a peak of arrivals in the morning, more prevalent than departures, and a "phased-out afternoon outflow". This pattern is typical of downtown stations, as shown by figure 4.7. Typically, these stations have blue bubbles during the morning and orange bubbles in the afternoon. Examples of such cases are stations 305 (see figure 4.8) and 307 (see figure 4.9):

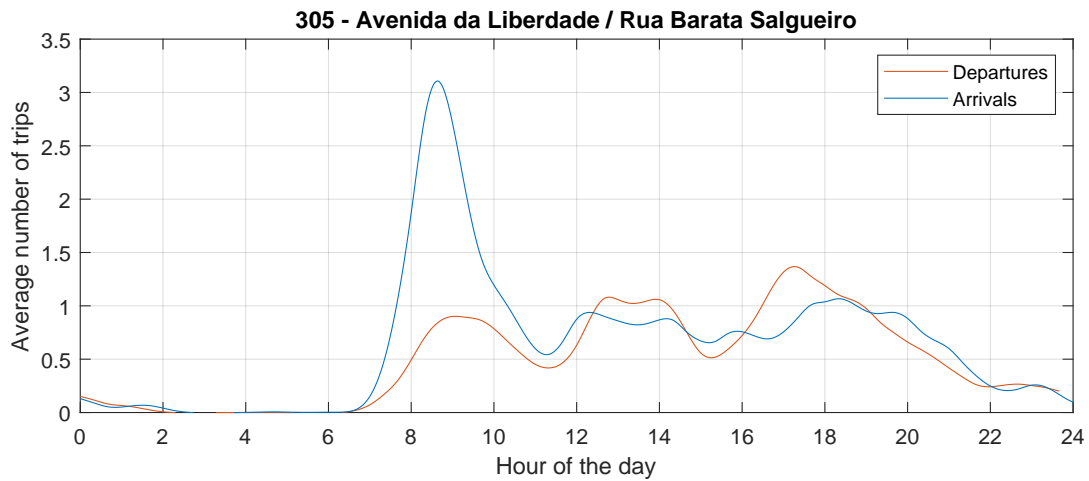


Figure 4.8: Station 305 daily (workday) pattern of arrivals and departures (20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

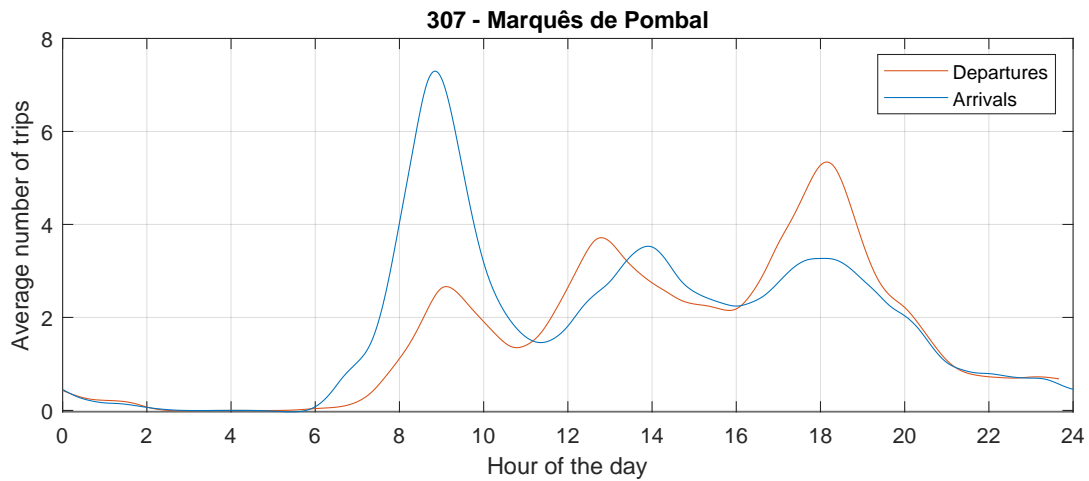


Figure 4.9: Station 307 daily (workday) pattern of arrivals and departures (20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

2. Contrary to the first group, this type of station has a peak of departures in the morning and more arrivals in the afternoon. According to Dell'Amico et al. [18] these stations are found near "park-and-ride" areas. Rodrigues [24] applied this logic to stations near "train stations in the middle of the city" and added stations near "some habitational areas". Stations 486 (see figure 4.10) in *Entrecampos*, with both metro and train stations nearby, or station 446 (see figure 4.11) in *Telheiras*, with a metro station and a habitational area, are a good example of it.

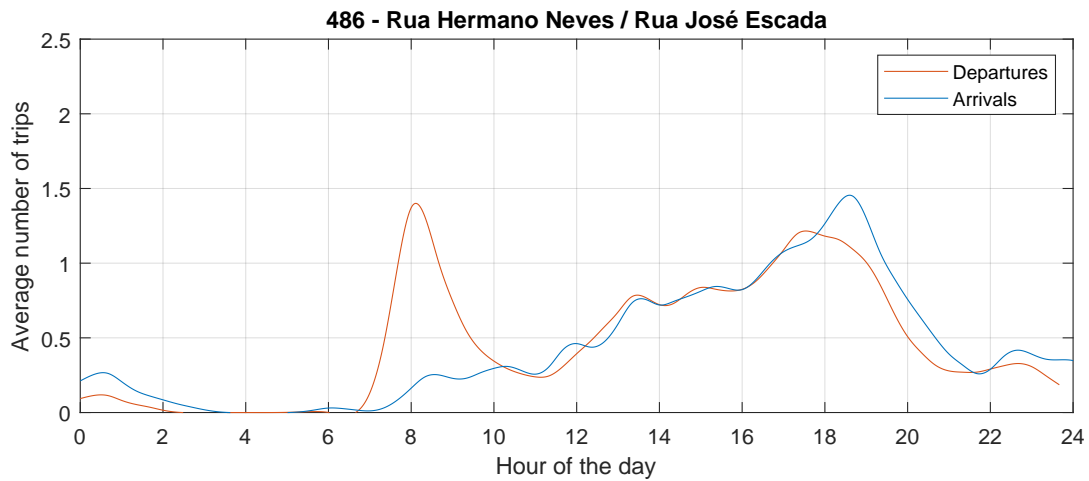


Figure 4.10: Station 486 daily (workday) pattern of arrivals and departures (20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

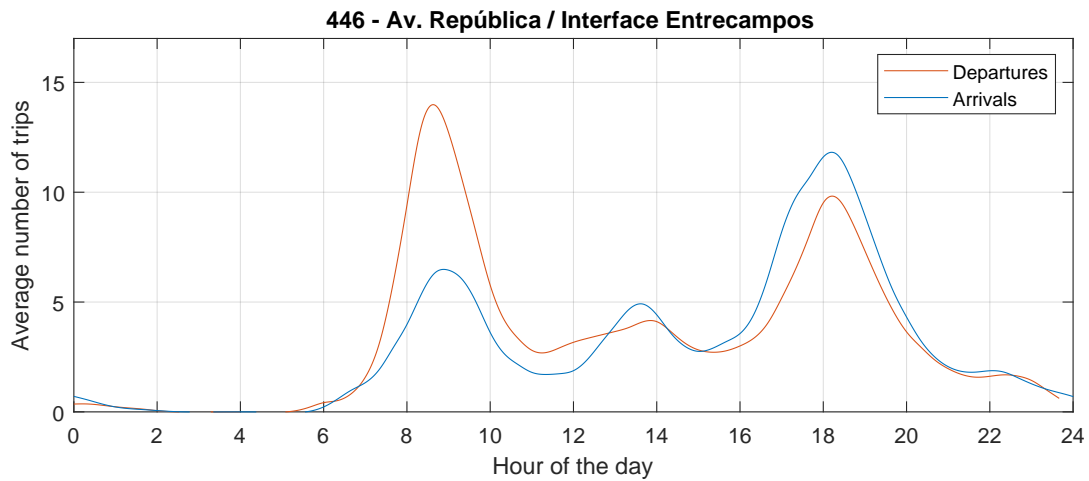


Figure 4.11: Station 446 daily (workday) pattern of arrivals and departures (20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

3. The last group corresponds to "stations that present similar arrival and departure patterns during the course of the day" and, consequently, are capable of keeping themselves relatively balanced. Some examples are stations 481 (see figure 4.12) and 413 (see figure 4.13).

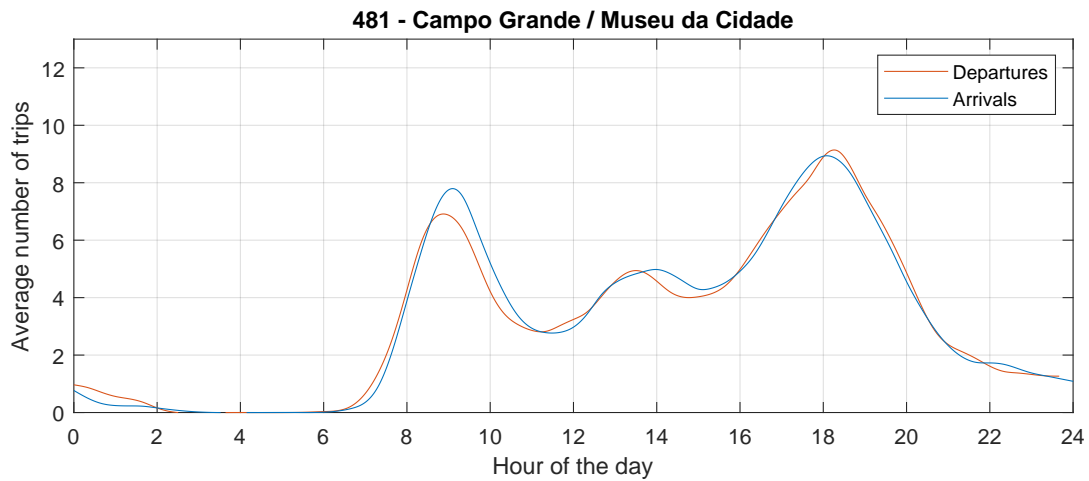


Figure 4.12: Station 481 daily (workday) pattern of arrivals and departures (20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

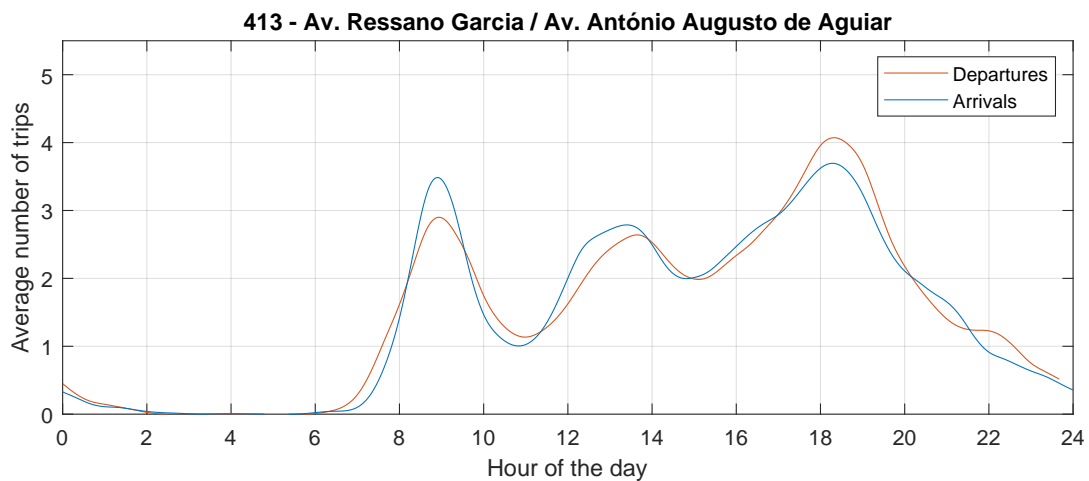


Figure 4.13: Station 413 daily (workday) pattern of arrivals and departures (20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

4.3 Operator tasks

During the period of available data (January to March of 2019), Siemens was the company responsible for the maintenance of all Gira assets, having live information about assets' location, state, or occupancy. This allowed them to plan actions, but also to intervene when something is wrong with some of these assets. To perform these actions, Siemens has multiple teams of operators working day and night.

Outside operations can be divided into four types: repositioning (includes repositioning between stations, added bikes coming from depot and bikes transported to depot), inspection and cleaning, corrective maintenance, and trip assistance. Pie chart 4.14 displays the proportion of each task type for a total of 14.804 completed tasks within the three months of analysis. Showing that repositioning tasks represent 18% of the total amount of the operator's actions, with corrective maintenance taking the largest part.

Repositioning operations are performed across all day; therefore, a dynamic methodology is executed. Operation management team have a program responsible for obtaining the best sequence of reposition-

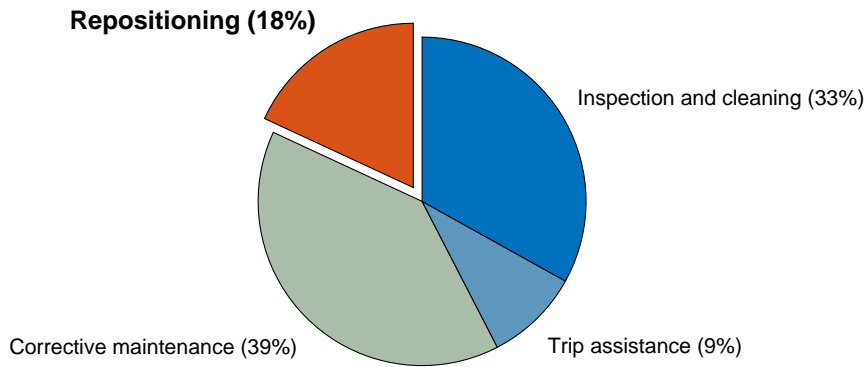


Figure 4.14: Percentage of tasks per type of operation

ing tasks; these include stations to visit and the number of bikes to drop off or pick up in each visit. Unlike Gira's real type of operation (dynamic repositioning), this work will, as already mentioned, implement a static repositioning methodology to rebalance the system during the night.

Some of the information about the characteristics of the operations was presented by Rodrigues [24] and again confirmed to this work. It is stated in the work of Rodrigues [24] that Siemens has a fleet of five electric trucks, this being a contractual imposition, with each truck able to transport up to 5 bikes, and the capability of attaching a trail to increase its capacity. There are three different trails, one able to transport 6 bikes, another 7 and the last, 10 bikes. Trucks have an autonomy of 110 km without trail, and 80 km with the trail, a low range when compared to the combustion engine, so distances to chargers must be considered.

In Rodrigues [24] analysis to operation behaviours, he states that there are five different teams working at shifts of 8h each, and usually, during the week there are 2 to 4 teams working at the same time. In what concerns repositioning there are more operations during workdays, most of them occurring after the morning peak, and less in the weekend. Through the analysis of the average number of total actions performed across the day, he concludes that most of the interventions are of a "responsive nature, instead of preventive", since the number of operations follows usage patterns.

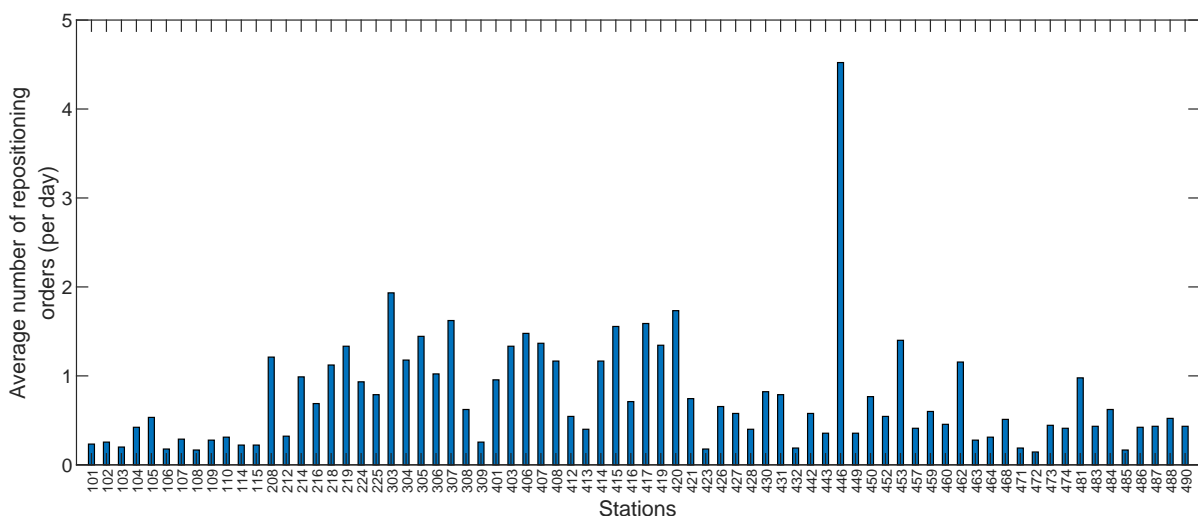


Figure 4.15: Average number of repositioning visits per day at each station

Reverting to repositioning tasks, figure 4.15 displays the average number of visits the stations receive per day. In three months, around 5.000 visits were recorded, with an average of 56 visits per day, in a total of 2.678 work orders. These orders usually have descriptions like: Remove N bikes from station A to station B , when bikes are being transported between stations, but can also acquire the format: Take N bikes to station C . This is, for example, related to the transportation of bikes from depot to a certain station.

One station can be visited more than once a day. Figure 4.15 shows that station 446 - *Av. Republica/Interface Entrecampos* is the one that requires more visits, a comprehensive behaviour justified by its daily pattern shown in figure 4.11, where the demand clearly overcomes the supply during the morning, reversing its behaviour in the afternoon. Other stations, on average, do not require more than one or two visits a day.

Chapter 5

Implementation

The similarity between simulators and reality depends on the models used, their size and the number of parameters. They cannot perfectly reproduce reality, but so far, simulators are still one of the best tools to forecast behaviours and reach some conclusions. Therefore, one of the best ways to predict how will UIs work, without having the risk of losing money in real experiments, is by testing them on simulators.

Setting as objective to understand customers' influence on repositioning, a simulation model of Gira in Anylogic was created, based on data from 3 months of 2019 (January to March). Primarily, this simulator generates trips and shows bikes moving between stations on a map of the city. However, it has also the possibility of activating user incentive methods and nightly repositioning, due to an interface where the user is able to configure some options.

Software choice, model formulation and the simulator application to Anylogic, are the items presented in this chapter. The objective is to give the reader a proper explanation of what and how it was created.

5.1 Used software

Anylogic

Anylogic was the chosen simulation software. According to their website, Anylogic "is utilised in thousands of commercial organisations and academic institutions" being the "leading simulation modelling software for business applications", used in industries like supply chain, manufacturing, transportation, warehouse operations, rail logistics, healthcare, marketing, asset management, etc. [67].

Anylogic uses three methods: discrete event, agent-based and system dynamics.

- The first consists of a process flowchart where blocks are operations connected by transitions. These can be conditions, timeouts, rates, messages or agent arrivals.
- Agent-based modelling offers the chance of creating agents with their own characteristics. These agents can be a population of agents, such as trucks or consumers; a single agent, like a factory,

hospital, a supplier; or simply an agent type like a transaction or a structural part of the model, such as sub-process.

- System dynamics is a highly abstract method of modelling, where the dynamism is modelled with flows, dynamic variables, rates, time-dependent functions, etc.

MATLAB

To make the analysis, data processing as well as the processing of results, MATLAB was the chosen software.

IBM ILOG CPLEX

To solve the repositioning MIP presented in the section 3.3 IBM ILOG CPLEX Optimisation Studio software package (version 12.10) was used. This seems to be the most used software to solve static repositioning problems, e.g. Dell'Amico et al. [18], Rainer-Harbach et al. [19], Ho and Szeto [29], Forma et al. [32].

5.2 Modelling

5.2.1 Trip generation

One of the main processes when building a BSS simulator is trip generation. Following the work of Pfrommer et al., days were split into 72 slices, so that a new number of trips are generated every 20 minutes for every start-end station relation. This number relies on a Poisson distribution of parameter $\lambda_w(s_{out}, s_{in}, t)$, where s_{out} is the initial station, s_{in} the end station, t the day time interval (e.g. 8:40 to 9am), and w the day type (workday/weekend). $\lambda_w(s_{out}, s_{in}, t)$ values are obtained and stored after taking the average number of trips of each (w, s_{out}, s_{in}, t) relation from real trip data. Since from 2 AM to 6 AM the system is not available for customers, the number of trips generated is zero within this interval.

Each of these new trips has a unique trip time that is sampled from a Lognormal distribution of parameter $TripTime(s_{out}, s_{in})$.

Since from figure 4.6 it is possible to identify a clear correlation between workdays and weekend, with the objective of reducing the size of input data ($\lambda_w(s_{out}, s_{in}, t)$) and increase sample's size, days were decoupled into two types, workdays and weekends.

Although this simulation creates trips every 20-minute interval, these are not launched in that precise moment, instead, they start after a random uniformly distributed number of minutes between 0 and 20. This avoids caveats like the ones modelled by Pfrommer et al. [5] (all trips created for a determined interval of time, are launched at a precise moment; for example, all trips created for the interval 10 AM to 10:20 AM are launched at 10 AM), approximating this implementation to reality.

Trip generation and trip time are the two variable inputs that bring dynamism to the system. Below, there is an analysis and explanation of these variables:

$\lambda_w(s_{out}, s_{in}, t)$ probability distribution

Data analysis has shown that the Probability Mass Function (PMF) of the number of trips generated at each (w, s_{out}, s_{in}, t) relation seems to fit a distribution. Therefore, were applied Matlab tools to find the parameters of the Poisson and Normal that best suit the PMF. Figures 5.1, 5.1b and 5.1c show this behaviour and give also a perception of which distribution, between Normal and Poisson, best suits the received data for the relation (w, s_{out}, s_{in}, t) . A simple visual analysis shows Poisson as the best fit. To confirm it, the Mean Squared Error (MSE) was used as a measure of goodness of fit:

$$MSE = \frac{1}{n} \sum_{k=1}^n (\hat{y}_k - y_k)^2 \quad (5.1)$$

where n is the number of data points, \hat{y}_k the value returned by the fit and y_k the actual value for data point k .

Table 5.1 shows that Poisson presents the lower values of AMSE, which, for this case, represents the average value of all $MSE(s_{out}, s_{in}, t)$. This result confirms the use of Poisson in other researches, e.g. Chemla et al. [7], Fricker and Gast [20] or Aeschbach et al. [6].

	Fit PMF	
	Normal	Poisson
Workday	0.2079	0.0149
Weekend	0.0928	0.0223

Table 5.1: AMSE between real and fit PMFs of the generated trips for all combinations of (s_{out}, s_{in}, t)

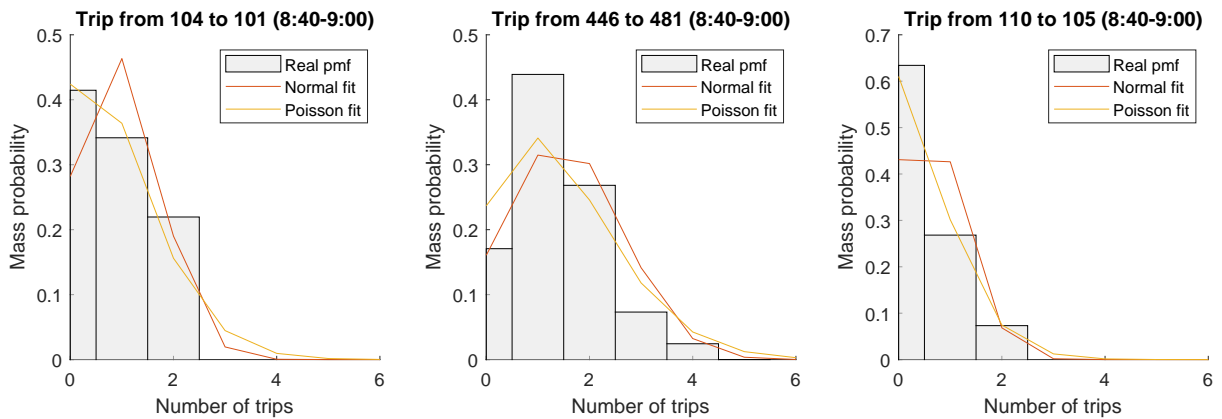


Figure 5.1: PMF of the average number of trips between start and end stations (8:40 AM to 9:00 AM)

***TripTime*_(*s_{out}*, *s_{in}*) probability distribution**

Just like the number of trips between stations at interval t , trip time also seems to fit a distribution. For this case, the only variables chosen were arrival and departure stations, without considering time variable t or weekday type w . To check which is the best fitting distribution, Normal, Poisson, Rayleigh, Gamma or Lognormal (with $\gamma = 0$) were taken into account. A visual examination of figures 5.2b, 5.2c and 5.2a shows Lognormal as the best fit for the trip time between stations. This is confirmed by table 5.2, where the Lognormal has the lowest AMSE (average value of all $MSE(s_{out}, s_{in})$).

Fit PMF				
<i>Normal</i>	<i>Poisson</i>	<i>Rayleigh</i>	<i>Gamma</i>	<i>Lognormal</i>
0.0997	0.1088	0.1059	0.0895	0.0785

Table 5.2: AMSE between real and fit PMFs of the total trip time from s_{out} to s_{in} for all combinations of (s_{out}, s_{in})

Although this work found this Lognormal relation, from the research made has not been found any article that modelled trip duration this way. For example, Patel et al. [38] determined trip duration "based on the distance and the average bike speed." Nevertheless, Lognormal distribution is slightly addressed by Ljubenkova, Kon, and Ratti [68] claiming: "Trip duration follows a Lognormal distribution with a median of 10 minutes" in Boston Blue Bikes BSS; and also by Fricker and Gast [20] who evaluates the effect of exponential, average, Lognormal and uniform distributions in the "proportion of problematic stations as a function of the average trip duration" coming to the conclusion that the problematic "is increasing when going from deterministic to uniform to Lognormal to exponential distribution".

5.2.2 User behaviour

The behaviour of users was modelled according to the work of Aeschbach et al. [6]. At pick up moments, when a user arrives to a station without available bikes (empty station), that person goes to the nearest station; if that station also does not have bicycles, the person leaves the system. At drop off moments, when a user arrives at a station without available docks (full station), the user travels to the nearest station that is not full.

5.2.3 User incentives (UI)

After trips are generated, users have the opportunity of helping in the rebalancing of the system. Topically, receiving an incentive for it (points, trips, etc.).

The basis of this thesis is to evaluate the effect of user incentives in Gira BSS. This way, two main methods were used and tested:

- Method 1 (M1) followed the work of Aeschbach et al. [6] (see section 2.3). It consists of using App where people may inform the system of its start and end stations, subsequently receiving, as

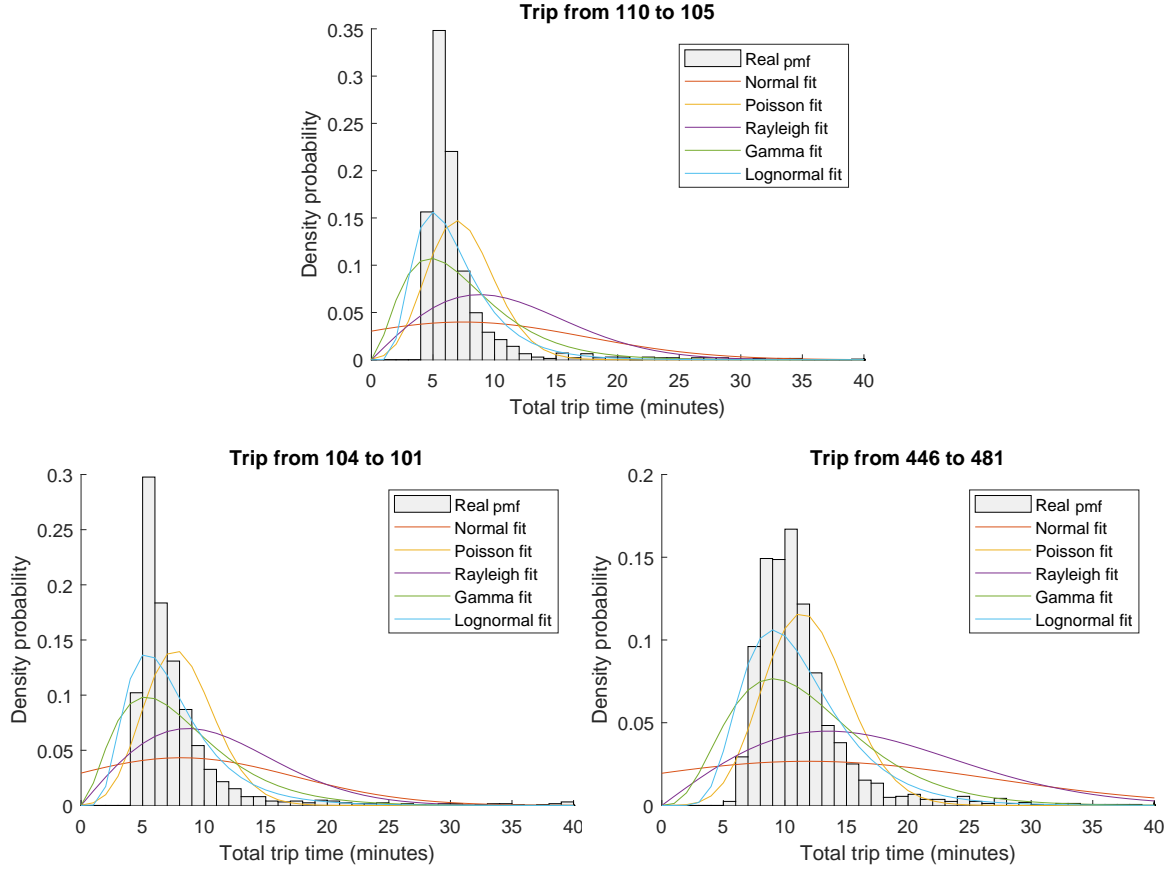


Figure 5.2: Real PMF and other PDF fit functions to trip time between start and end stations

output, alternative stations - that could help to balance stations inventory - to begin and conclude its journey. To calculate these alternative stations, the model takes into consideration the concept of neighbours. Stations that are within a certain radius from the inputted station are considered its neighbours. Therefore, the one from the group which has a higher/lower occupancy level is presented as the best departure/arrival station. This strategy is applied separately and only implemented if the departure station occupancy level is above $1 - WPC$ or the arrival station occupancy level is below WPC , where WPC (Weak Preemptive Control) is the station level control parameter, so that, when considering $WPC = 0$, the strategy is not applied, but if $WPC = 1$, it is always applied. It must be emphasised that when the best station from the group is the one introduced by the user, it will also be the one shown after the strategy is applied.

- Method 2 (M2) tries to reproduce the incentive methodology used by Gira. The concept of neighbour is also considered, however, the nearest station with occupancy level above $1 - WPC$ is presented for departure station or below WPC for arrival station, instead of the best from the group. Another difference to the first method is that when none of the neighbours fulfils the requirements (to be above $1 - WPC$ if it is a departure or below WPC if it is an arrival), even though the input station has the worst inventory ratio, the output will be the same as the input. Since Gira applies 0.3 for arrival and 0.7 for departure as the ratios for which people receive points, this was the ratio used ($WPC = 0.7$).

To model people participation behaviour, a CC ratio was taken into account. Since some may not want to accept both arrival and departure alternative station presented by the app, it was also considered that from the users who want to cooperate, $1/3$ would agree with the alternative arrival station, $1/3$ with the departure station and the last $1/3$ to both departure and arrival (see figure 5.3). These considerations were not taken into account by Aeschbach et al. [6]. In their paper, people accept both departure and arrival.

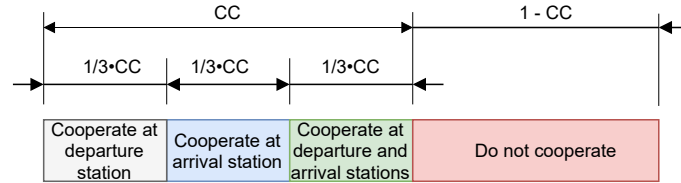


Figure 5.3: Stacked bar graph of types of cooperation

5.2.4 Repositioning

Though Gira operates with a dynamic repositioning and an algorithm, constantly running to obtain the best orders of repositioning, in this thesis, to drop significantly computational time and since the final objective is not to find the best repositioning method, but instead evaluate User Incentive (UI) methods, it was decided to implement static MILP method of Schuijbroek et al. [4], which has the particularity of achieving good results in a few seconds or minutes. This method is described in section 3.3.

In what concerns the implementation of this MILP into the simulator, during every night, the program starts by clustering stations to each truck, and then calculate repositioning tasks for every truck. After they get the instructions, they leave the depot and travel across stations dropping and collecting bikes to balance the BSS. In the end, they return to the depot.

Station patterns in section 4.2 show that the first unbalances occur at the beginning of the day, with stations behaviours inverting during the afternoon. Therefore, a good nightly rebalance should be able to handle not just the morning demand but some in the afternoon.

Unlike many static methodologies in the literature [18, 31, 32], Schuijbroek et al. [4] use target intervals of station-level, instead of target values. To find these intervals the notes of Hulot et al. [57] and Schuijbroek et al. [4] were taken into account. The procedure applied to find these values is presented in the section below.

Intervals of Repositioning

Repositioning methods have two alternatives: they either use fill targets [28, 31], the most applied alternative; or target intervals [4, 25]. Since this thesis work implemented Schuijbroek et al. [4] method, target intervals were found.

According to Schuijbroek et al. [4], the number of arrival and departure bikes during a certain "time

horizon period” follows a Poisson distribution, a behaviour confirmed in Gira BSS in section 5.2.1. With this information, service level probability “ $p_s(f, obj, t)$ ” can be computed as a sum and difference of Poisson processes, (...) a Skellam distribution” [57]. Therefore, with the Poisson parameters of arrivals and departures, it is possible to get station level distribution probability.

To get the best minimum and maximum levels of repositioning, after getting Skellam distribution from Poisson parameters of arrivals (λ_{in}) and departures (λ_{out}) with equation 3.36, the Skellam distribution must be rearranged. This rearrangement should result in the ratio of trust for which station level is kept within zero and station capacity when starting with a certain level. Station levels with a ratio of trust higher than a determined value, for example, 0.85, indicates that there is an 85 % chance that the station will remain in a balanced state.

In the example presented in figures 5.4, 5.5 and 5.6 the station should have at the beginning of the day a minimum of 0 bikes and a maximum of 20.

This trust level curve is found for all levels. The equation to find trust values is the following:

$$\sum_{i=-sl}^{StationCapacity-sl} PMF_{Skellam}(\lambda_{in}, \lambda_{out}, sl) \quad (5.2)$$

where sl is station level.

For example, for $sl = 1$ the sum of the Skellam distribution probabilities from -1 to $(stationcapacity) - 1$ gives the level of trust.

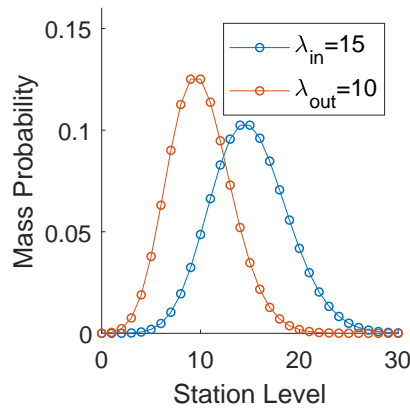


Figure 5.4: Poisson distribution functions ($\lambda_{in}=15$, $\lambda_{out}=10$)

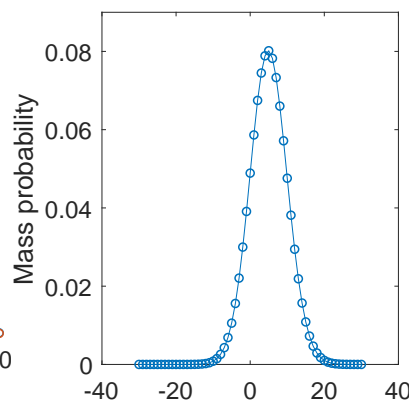


Figure 5.5: Skellam distribution function ($\lambda_{in}=15$, $\lambda_{out}=10$)

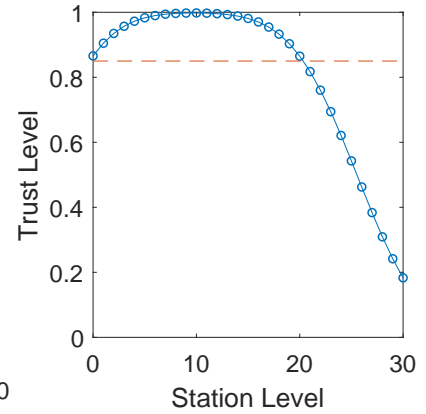


Figure 5.6: Trust level distribution ($\lambda_{in}=15$, $\lambda_{out}=10$)

This work used two sets of repositioning target intervals, one for workdays and another for weekends. For workdays data from the “time horizon period” from 6 AM to 9 AM was used, because the first peak occurs during this period. Trying to increase this period would not be a good practice due to station behaviour. For example, if during the morning there is an average of 10 bikes exiting and 15 bikes entering, balance is positive, but if during the afternoon the balance drops to -5 bikes, a union between morning and afternoon would result in a null balance. This could result in max/min target intervals that could not handle so well the high demand of the morning period.

The "time horizon period" chosen for weekend days was from 6 AM to 1 PM. This period is larger due to stations' demand curves on weekends, more regular during the morning and with a peak at 4 PM.

In this work, to define target intervals, a minimum level of trust of 0.85 was applied to all stations. These intervals were used as inputs to the model and more precisely to the repositioning MILP implemented.

5.3 Anylogic simulator

This simulator combines the dynamics of the system through trips generation; communication with customers to model the effect of users help on repositioning; and a static repositioning that occurs at night. These items as well as simulation procedures and modelled agents will be presented and explained in the following sections. The most important java codes and a simulator user model are available in the Appendix A.

5.3.1 Agents

To implement the model explained in previous sections in Anylogic (simulation software), its major abilities were used: agents and flowcharts. Agents can have only parameters and variables to characterise them, but can also have flowcharts to indicate their states and move them between places. Just to clarify, agent types can either be single or have a population of its type higher or equal to 1. So that, when considering a population agent type, each agent follows its own path in the flowchart and can be accessed individually. To simulate Gira BSS, six different agents were created: Main, Person, Bike, Station, Depot and Truck. Agents' characteristics are summarised below:

- **Agent Main** is the main agent of the program. It is where all the other agents are stored, as well as input files, functions, parameters and variables. Main also has the GIS (Geographic Information System) Map, which during the simulation will show the city map, stations, trucks, and bikes moving between stations, but also the controls where input values are inserted. Main stores the function that initialises the BSS, generate trips and obtain repositioning instructions.
- **Agent Person** defines every user that enters the system right after its trip is created in main and has not yet grabbed a bike. It follows the behaviour of a real user: by deciding whether to accept user incentives or not at the beginning of its trip (a decision based on a CC ratio); or by walking to another station if its intended initial station has no bikes available. All individual agents *person* follow a flowchart and take actions depending on their decisions and the stations' occupations. It is inside this agent that user incentive methods are used to calculate alternative stations and is where the *person* decides to take them or not.
- **Agent Bike** represents every individual agent *bike* in the system. It is activated when a *person* grabs a bike and takes a trip between stations. Just like type Person, its actions are taken by following a flowchart, with full stations taking major importance on the followed path.

- **Agent Station** is a static type agent that represents BSS stations — places where bikes are stored so that users can pick them up or drop them off. It is mainly characterised by its coordinates, name, and identification number, number of docks, occupation percentage, etc.
- **Agent Depot** is also a static agent that represents BSS depot. It is the place where trucks initiate and finish their tasks and where bikes can be stored if at the end of the repositioning trucks are not empty. It is only characterised by its location.
- **Agent Truck** represents the population of trucks responsible for repositioning tasks. After repositioning instructions (path and number of bikes to drop or collect) are calculated, in agent Main, for each truck, they are activated to move bikes between stations or to depot. To complete its tasks, follows a flowchart, just like Bike and Person. It is mainly characterised by instructions received and the number of bikes capable of carrying.

Printscreens and key parameters of the agents are shown in Appendix A in section A.3.

5.3.2 Simulator inputs/outputs

To feed the simulator, some excel files, created in MATLAB, with characteristics of the network, need to be available. These files and resulting Database (DB) tables are explained below.

Inputs:

- *stationProp.xlsx*: contains stations properties, more specifically, station code number, station name (usually refers to the street where the station is placed), number of docks, and geographical coordinates (latitude and longitude).
- *lambda20.xlsx*: contains Poisson parameters ($\lambda_w(s_{out}, s_{in}, t)$) for the demand of each pair of stations for every 20 minutes and day type (workday or weekend). While preparing data for the simulator, this file must be uploaded to a table inside the simulation database called *lambda20*.
- *lambda20pred.xlsx*: contains Poisson parameters ($\lambda_w(s_{out}, s_{in}, t)$) for the demand of each pair of stations for every 20 minutes and day type (workday or weekend). While preparing data for the simulator, this file must be uploaded to a table inside the simulation database called *lambda20Pred*. This file is used for prediction.
- *DistMatrix.xlsx*: contains the distance matrix, with every distance between every station in km;
- *InitialOccupation.xlsx*: vector with initial stations occupation. Its sum results in the total number of bikes used in the simulation;
- *Intervals.xlsx*: Contains target intervals of repositioning, with the minimum (s_i^{min}) and maximum (s_i^{max}) number of bikes at each station in the beginning of the day at 6 AM, for workdays and weekends. So that the system stays balanced at least until 9 AM on workdays, and 12 AM on weekends. These values were obtained based on past demand and Skellam distribution;
- *Neighbors.xlsx*: the first page contains a 2D matrix with the indexes of stations sorted by distance, with the nearest station on the left and the furthest on the right; the second page contains a 2D

matrix with the correspondent distances.

- *TripTime*: this file contains the parameters: minimum, μ and σ ; needed to define the Lognormal distribution of every pair of stations trip time. The first page contains a 2D matrix with the mean of the Normal and, the second page includes the standard deviation.

Apart from the above files, certain input values should also be defined.

Right after the *Run* simulation button is clicked, a new window pops up. This will effectively initiate a run, but the simulation is paused to insert some input values. At this time, values of the following input entries can be changed:

- *Method*: represents the methods allowed for the run. Shows up as a checkbox where it must be chosen if the run allows nightly repositioning and/or user incentive methods.
- *Rad*: value related to user incentive methods, which represents the maximum radius for which stations are considered neighbours from 0 to 1 km.
- *CustCoop*: value related to user incentive methods, which represents the percentage of customers willing to participate in the system balance.
- *wpcPerc*: value related to user incentive methods, which represents the ratio factor for which stations are/are not considered as being in a balanced situation.
- *NbrVehc*: value related to repositioning, which represents the number of vehicles involved in BSS rebalancing tasks.
- *TruckCapacity*: value related to repositioning, which represents every truck bike capacity.

With all the possible inputs checked and inserted, *Run* button should be clicked and the simulation proceeds until the stop button is clicked or a pre-defined stop date is reached.

There are two other parameters to consider, but these cannot be changed in the input window. Those parameters are:

- *Truck speed*: value related to repositioning, which represents every truck speed when travelling between station or to depot.
- *Person Speed*: speed for which a *person* travels to another station after finding an empty station at the beginning of its trip.

Outputs:

Anylogic allows storing information along with runs within database tables. That capability was properly used to store information about each run for different parameters.

- DB table *trip_table*: records all trips departure and arrival stations, departure and arrival dates, type of user (depending on the participation or not in bike repositioning), number of failed departures and arrivals, as well as the extra effort he might have done;
- DB table *ok_table*: records every day counting of service and no-service events in departure and arrival stations;

- DB table *lost_users*: lost users occur after they try to grab a bike in two different empty stations. This table records the date, and the first and second stations visited.

Repositioning tasks and other information related to each truck trip cannot be stored in the database, so an excel file, initially empty, was used:

- *RepResults.xlsx*: records information about repositioning tasks, the number of bikes at the beginning and at the end of repositioning, visited stations, amount of bikes to pick up or drop off, truck total travelled distance and truck operating time.

5.3.3 Simulation procedures

The simulation process is divided into 6 steps that act individually but constantly interact with each other. Those sections are initialisation of the city; trip generation; pre-trip *person* decision process; bike trip; calculus of the repositioning instruction; and truck repositioning operations. All of them will be explained below, based on what was modelled in previous sections.

1. Before the whole dynamic of the BSS starts, the environment must be prepared. BSS assets must be placed in their locations in the map, information inside excel files transferred to arrays, and output files cleared. Based on information from *StationProp.xlsx*, stations are generated in type Station and located in their position. The same occurs to Depot, which, unlike Station, is a single agent. Taking into consideration the number of bikes per station from *InitialOccupation.xlsx*, and running through all stations, an equal number of bikes is generated in type Bike and placed at that station.
2. The second step is when the action starts, with the generation of trips. Every 20 minutes, all (s_{out}, s_{in}) pairs access "WeekArray" (generated from database *lambda20*) to get the Poisson parameter $\lambda_{s_{out}, s_{in}, t}$, and, followingly, generate a random value from the resultant Poisson distribution. The number that pops up results in the equivalent number of people generated in type Person, creating a virtual user with pre-defined preferences of start and end stations (s_{out}, s_{in}) . Therefore, after a trip is generated, a single agent *person* of type Person is created (with that trip initial preferences). A message saying *NewOrder* is then sent to that same agent to start its interaction with the system. The next steps take place in the agent *person*.
3. The third step occurs in agent type Person. Following a flowchart, after *person* receives the message *NewOrder*, he stops for a random timeout period (uniform distribution) between 0 and 20 minutes. This is what introduces time randomness to trip generation. Next, depending on the CC ratio, it is drawn whether the user decides to help with repositioning or not and if the plan is to participate in the start and/or end station. If there is cooperation, user incentive methods are applied and the best stations within a determined Rad are calculated. If the resulting stations are different than the initial, *person* parameters *StartSt* and/or *EndSt* are changed.

4. The number of bikes at the initial station is then checked, and if empty, this agent moves to the nearest station being travelled distance added to the *ExtraEffort* trip property. If that station is also empty, or if the user did not participate in UI methods but has also travelled to two stations and none had bikes, the user leaves the system and is added to the *lost_users* table. Otherwise, trip time is obtained. To get this value, Lognormal parameters for the current (s_{out}, s_{in}) pair are accessed in the *TripTMat* array (obtained from *TripTime.xlsx*) and drawn a value from the respective Lognormal distribution. If the initial station has available bikes: a random bike located in that station is chosen; trip's information (*TripTime*, *StartSt*, *EndSt*, *StartDate*, *ExtraEffort*, etc) is sent to that specific bike; the *moveTo(EndSt)* order is set; and the number of bikes at the station updated. This order will initiate bike movement to s_{in} , with the bike leaving its initial state and initiating its path on the flowchart. When it arrives at the end station, the station's occupation is verified. If it is lower than 100% bike is left there, otherwise, a new path is traced to the closest station that has available empty docks. When this occurs, the *EndSt* variable is changed, and the number of extra meters travelled is added to *ExtraEffort*. This procedure is repeated until a free spot is found.
5. With the bike docked, the number of bikes at the *EndSt* is updated and final information about the trip (*TripTime*, *StartSt*, *EndSt*, *StartDate*, *EndDate*, *ExtraEffort*, etc) is added to *trip_table*.
6. At 2:30 AM repositioning instructions are calculated using the static repositioning method explained in section 3.3. This method takes into consideration the current station's occupation level, by defining that, at the end of the repositioning, this level should meet an interval criterion, with a minimum and a maximum number of bikes. These values are stored in the *IntervalMin* and *IntervalMax* arrays, obtained from file *IntervalOfOccupation.xlsx*. First, in agent Main the clusters of stations inside the block function *Clustering* are calculated; then, for every truck, the sequence of stations to visit and the number of bikes to drop off or pick up are obtained inside the block function *Repositioning*. After this, a message saying *start* is sent for all trucks of type *Truck* and it is started the sequence of trips between stations to transport bikes. When there are no more instructions, the truck returns to the Depot, existing the possibility of returning with or without bikes.

A scheme with part of what has been explained before is shown in figure 5.7.

5.3.4 Visual interface

Throughout simulation runs, a GIS Map shows stations changing their colours depending on occupancy level, users travelling between stations and trucks moving bikes.

When empty, stations turn blue, red if full, or green if neither full nor empty.

The Person icon is shown when the *person* is travelling to a second station searching for bikes, in other words, after visiting an empty first station.

If either the first or second start stations have bikes available, that means that that trip is active in agent

Chapter 6

Evaluation

This chapter presents the results of experiments carried out in the simulator. These experiments took place in a computer equipped with Windows 10, an Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, and 8.00GB of RAM.

Initially, the model will be validated in all of its components, followed by an analysis of UI methods.

6.1 Data filtering

To analyse Gira BSS and obtain valid input values for the simulator, data from the trip database was filtered, starting with 409197 trips recorded. The filters applied are listed below, and if a trip record had any of these characteristics, it was removed:

1. National or school holiday: usually these days have a much lower number of trips and do not follow regular patterns;
2. Missing arrival station: software or hardware errors, as well as stolen bikes, are some of the reasons for this type of record;
3. Trip time higher than 8 hours: this time interval is the maximum a bike is allowed to travel [69];
4. Average trip speed higher than 30 km/h;
5. Trip starting between 2 AM and 6 AM: this is the interval when the system is offline for users [70];
6. Trip starting and ending at the same station with a trip time lower than 90 seconds.

For the fourth filter, it was considered that the maximum speed of a cyclist would be 42 km/h on a straight line between stations [71]. Since trips do not follow this type of line, a detour factor of 1.4 (following the work of Schuijbroek et al. [4]) was considered, resulting in a 30 km/h filter.

To justify the last filter, figure 6.1 shows that about 50% of the trips with the same departure and arrival stations have a trip time below 90 seconds. Below that the increase in the evaluated percentage was relatively high, moreover, did not seem plausible to have someone renting a bike for less than 90 seconds

(a value below 180 e.g. could also be a reasonable choice and fit the logic applied previously, however, this work aimed to exclude as few trips as possible). This type of behaviour is mainly associated with people removing and, immediately after, putting the bike back in the dock, either because it has some defect, or some kind of error occurred. In total there are 20685 trips registered with these characteristics, corresponding to 5% of the total amount of records — a considerable amount of unnecessary noise.

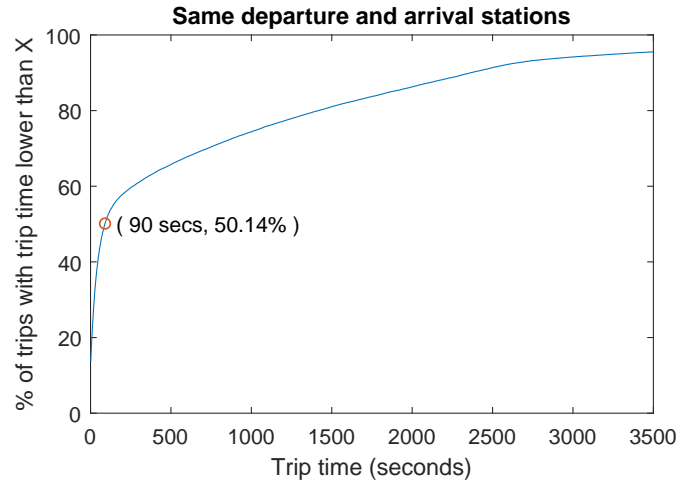


Figure 6.1: Cumulative percentage of trips for different trip times

After filtering, data ended up with 388512 records.

6.2 Parameters choice

Before validating the model and getting results from the application of different methods for different parameters values, some inputs were considered to be constant for every run:

- Truck speed = 50 km/h;
- Person speed = 1.4 m/s;
- $dp = dm = 357$;
- Loading/Unloading task time = Number of bikes to (un)load x 1 minute;
- Number of trucks = 4;
- Truck capacity = 25;

Truck capacity was chosen to be 25, a value that does not fit the 15 maximum capacity referred in section 4.3. This choice is explained by the type of repositioning chosen in this thesis. This allows, for example, a station to have 30 bikes at the end of the day, for a maximum target interval of 5. Given that this method can visit stations only once, 25 bikes would have to be picked up. Therefore, a lower truck capacity could lead certain stations not to be properly rebalanced.

The repositioning method is divided into two phases: clustering and repositioning (repeated for every truck). CPLEX was the solver used in Anylogic, and two parameters were defined:

- MIPGap - The MIP solver will terminate (with an optimal result) when the gap between the lower and upper objective bound is less than MIPGap times the absolute value of the incumbent objective value;
- TimeLimit - Limits the total time expended (in seconds) solving the MIP;

For both phases (clustering and repositioning) a MIPGap of 0.01 and a TimeLimit of 60 seconds were defined.

6.3 Verification and validation

To validate this model and assess how close it fits reality, filtered data from Gira was used, and the parameters presented in the previous subsection were applied. The simulator was validated by examining three major measures: number of trips generated, trip time, and number of users lost.

Datasets

To check the error between reality and the model, two $\lambda_w(i, j, t)$ datasets were created, to input parameters into the simulation. The first testing data, *dataset 1*, included all available trips after being filtered, corresponding to 3 months of trips.

With the goal of analysing if the simulator is a good predictor, a second test dataset, *dataset 2*, was used. Since figure 4.4 shows February and March with a similar number of trips, the predictor test dataset included data from 2nd of January to 15th of March, excluding the Carnival week from 4th of March to 8th of March due to a drop in the number of trips.

Scenarios

Four scenarios, described below, were assessed:

- (1) Since stations with restrictions have an impact on user initial preferences for departure and arrival stations, this first scenario used sink/source stations, which mean that station restrictions were not considered. The dataset used was *dataset 1*, and the comparison was made against the average values per day of the same 3 months of real data.
- (2) Similar to scenario (1), this scenario attempts to evaluate the prediction capabilities of this simulator, using sink/source stations. It was trained with *dataset 2*, and it was tested and compared for two weeks of data (16 of March to 31st of March).
- (3) Scenario for which station and bike restrictions were considered, applying methodologies related to Person and Bike behaviours modelled in previous sections. For this case, UI methods and

repositioning are turned off. *dataset 1* was used as training and were simulated for 28 straight days. Then compared to the average of 3 months of real data.

- (4) Similar to 3rd scenario (dataset and simulated days), except for one difference: *NR* method was activated. The main difference for the third relies on a reset made every night, using a fleet of 4 trucks with a capacity to transport up to 25 bikes each.

Trip generation

Trips were counted in four different formats, with one or two metrics per format.

Three metric types were applied for this validation: Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and Variance Accounted For (VAF), which checks the proximity between departure and arrival signals. A VAF value of 1 means that both signals have the same shape.

These formats and their respective metrics are presented below:

- **Average total trips per day:**

$$MAPE = \text{abs}(Ntrips - \overline{Ntrips}) / \overline{Ntrips} \quad (6.1)$$

where $Ntrips$ is the average number of trips per day in the simulator and \overline{Ntrips} the real demand;

- **Origin-destination matrix (O-D matrix)** or average number of trips per day starting at station S_{out} and ending at station S_{in} :

$$MAE = \text{mean}(\text{abs}(D_{S_{out},S_{in}} - \overline{D_{S_{out},S_{in}}})) \quad (6.2)$$

where $D_{S_{out},S_{in}}$ is a 2D counter matrix with the number of trips starting in S_{out} and ending in S_{in} , and $\overline{D_{S_{out},S_{in}}}$ the OD matrix based on real demand;

- **Average number of trips per day starting at station S_{out} at time interval t** (8:00 to 8:20, for example):

$$MAE = \text{mean}(\text{abs}(D_{S_{out},t} - \overline{D_{S_{out},t}})) \quad (6.3)$$

where $D_{S_{out},t}$ is a 2D counter matrix with the number of trips starting at S_{out} at time interval t , and $\overline{D_{S_{out},t}}$ the respective real matrix;

$$VAF = 1 - \text{var}(D_{s,t} - \overline{D_{s,t}}) / \text{var}(\overline{D_{s,t}}) \quad \forall s \in S_{out} \quad (6.4)$$

where $D_{s,t}$ is the counter vector with number of trips starting in $s \in S_{out}$ at time interval t and $\overline{D_{s,t}}$ the respective real counter vector. Since this metric results in n metrics where n =number of stations, the mean of all stations' VAF, MVAF (Mean Variance Accounted For), will be evaluated.

- Average number of trips per day ending at station S_{in} at time interval t :

$$MAE = \text{mean}(\text{abs}(D_{S_{in},t} - \overline{D_{S_{in},t}})) \quad (6.5)$$

where $D_{S_{in},t}$ is a 2D counter matrix with the number of trips starting in S_{in} at time interval t , and $\overline{D_{S_{in},t}}$ the respective real matrix;

$$VAF = 1 - \text{var}(D_{s,t} - \overline{D_{s,t}}) / \text{var}(\overline{D_{s,t}}) \quad \forall s \in S_{in} \quad (6.6)$$

where $D_{s,t}$ is the counter vector with number of trips starting in $s \in S_{in}$ at time interval t and $\overline{D_{s,t}}$ the respective real counter vector.

From the first format to the last two, trips are separated into more variables, therefore, higher error values are expected in the last two formats but also more importance on validation.

The average results of multiple days were considered. Stations (or pairs) will not be evaluated separately, therefore there might be individual cases with higher errors than the average of errors presented.

Table 6.1 summarises results of the metrics applied for the four scenarios, differentiating working (WD) from weekends (WE).

	Scenario 1		Scenario 2		Scenario 3		Scenario 4	
	WD	WE	WD	WE	WD	WE	WD	WE
(Total Trips)/Day - MAPE	0.16	0.24	0.27	1.72	8.14	3.82	4.41	3.28
(s_{out}, s_{in}) - MAE	0.07	0.08	0.34	0.29	4.11	1.53	4.04	1.52
(s_{out}, t) - MAE	0.13	0.13	0.35	0.33	0.22	0.20	0.19	0.19
(s_{out}, t) - MVAF	0.93	0.76	0.61	0.08	0.80	0.37	0.85	0.44
(s_{in}, t) - MAE	0.08	0.09	0.38	0.33	0.18	0.17	0.16	0.17
(s_{in}, t) - MVAF	0.97	0.87	0.52	-0.03	0.88	0.55	0.89	0.55

Table 6.1: Model versus simulation error metrics

As expected, results show that scenario (1) is the overall best with the lowest MAPE and MAE errors and highest MVAF. Not having restrictions approximated this scenario from real data, validating the way trips are generated. This also affected total trip MAPE and (s_{out}, s_{in}) -MAE of scenario (2) which came in second place for these metrics, showing that when it comes to (s_{out}, s_{in}) relation, without the time factor, the behaviour is consistent to reality. Problems emerged with MVAF values of (s_{in}, t) and (s_{out}, t) decreasing a lot in relation to scenario (1), showing the negative influence of the time factor, also present in $(s_{in}/s_{out}, t)$ -MAE.

Looking to the (s_{out}, t) -MVAF of scenario (1), during working days, and comparing it to scenario (3), this value dropped from 0.93 to 0.80. This drop is not so significant if repositioning is activated, assuming a value of 0.85. MAE errors of (s_{out}, t) and (s_{in}, t) increased from scenario (1) to (4) and (3), reinforcing that scenario (4) takes advantage over (3).

With restrictions activated for scenarios (3) and (4), MAPE and (s_{out}, s_{in}) -MAE errors got highly increased. This is due to empty and full stations that cause non-service events. This way, users see

themselves forced to change initial and/or end stations, or even to abandon the system.

When comparing workdays against weekends with MVAF $(s_{in}/s_{out}, t)$, workdays seem to take an advantage over weekends. Figures 6.2 and 6.3 show how well simulation curves follow the real case on workdays, and not so well on weekends.

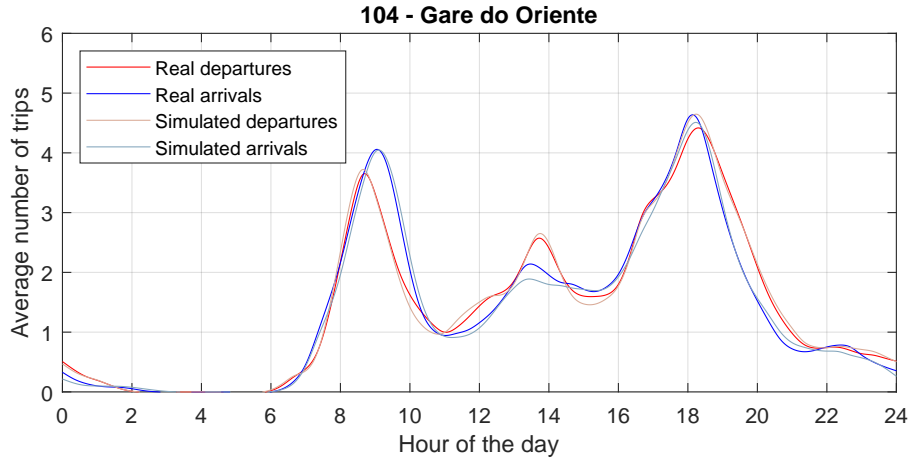


Figure 6.2: Average number of trips throughout day - real versus simulation (Scenario 1; workday)
(20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

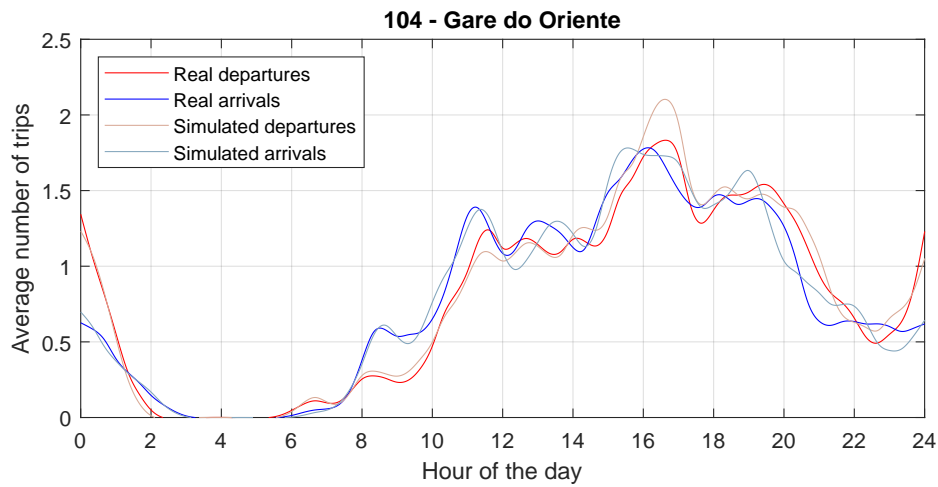


Figure 6.3: Average number of trips throughout day - real versus simulation (Scenario 1; weekend)
(20 minute interval data, "smoothingspline", SmoothingParam=0.9758)

Trip time

Another factor to evaluate is the time users spent in the system travelling between stations.

Figures A.2 and A.1 in Appendix A display real trip time Lognormal fit curves and compares it with simulated PMFs for all scenarios. Figure A.2 shows trips from station 101 to 104 and figure A.1 takes into account every trip recorded. From the visual analysis of these images, the similarity between the real curve and simulated PMF is evident, therefore validating trip time.

Repositioning (intervals and path)

The last factor to check is repositioning instructions. The nightly method applied in this thesis is different from reality, so it is not expected that the number of visits per day at each station is the same. Figure A.3 shows that, in reality, there are, on average, stations visited more than once a day. Given that the methodology applied does not allow a station to be visited more than once, this will be one of the differences between the simulator and real life. Yet, it is shown that some stations are visited with a higher frequency than in the real case. The explanation for this is that repositioning interval level requirements must be strictly fulfilled every day, and one bike outside that interval is enough to require a visit.

Most times, this method leaves stations with the maximum or minimum acceptable level, which is not supposed to be a problem in the morning of the next day, however, the probability of laying outside the admissible intervals at the end of the day is increased, therefore, becoming a station that should be visited during the night.

When comparing the average number of lost users across day between scenario (3) (without repositioning) and (4) (with repositioning) through figure 6.4, the impact of repositioning is clear, mostly during the morning of workdays and from 6 to 1 PM of weekends. This impact was already expected since these were the "time horizon periods" applied to get the intervals of repositioning.

A simple truck route validation can be made by looking at figure 6.5, that shows each truck route and the number of bikes picked up or delivered at each station.

Figure A.4 also validates the repositioning, showing that all self-sufficient station (blue dot) occupancy is maintained while not self-sufficient stations (red dot) occupancy enter the required occupancy level (black line) after repositioning (green dot).

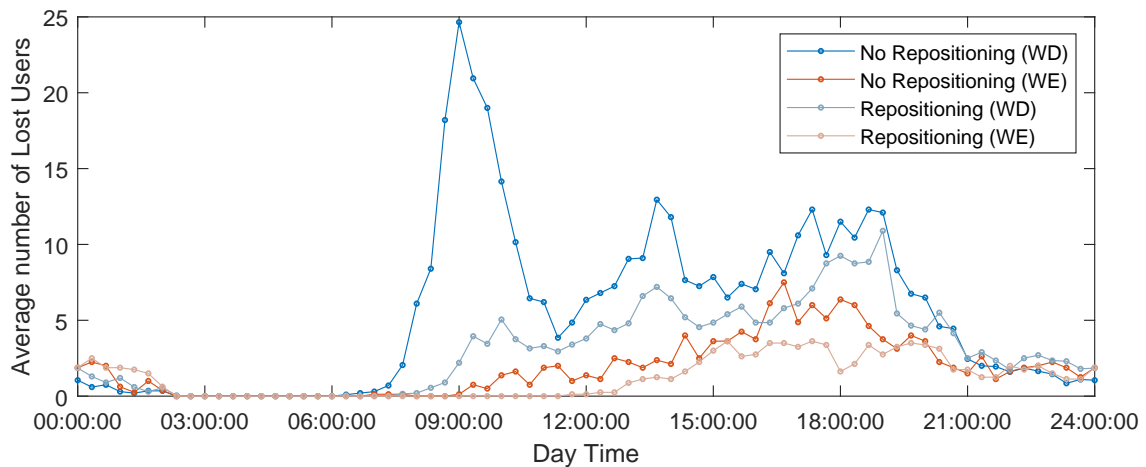


Figure 6.4: Scenario (3) (no repositioning) VS scenario (4) (with repositioning) average number of lost users throughout the day

and the Rad used to define sets of stations. These tests were accomplished with and without NR, with the last taking place during the night interval between 2 AM and 6 AM.

As explained in section 5.2.3, to approximate this model to reality, people choices are differentiated into three levels: help only in departure; help only in arrival; and help in both, being attributed to each case an equal percentage. For example, if CC is defined to 30%, each of the three previous cases has a 10% chance of happening.

For each scenario several experiments were carried out for different parameters. In the first experiments, neighbouring Rad was varied in intervals of 0.1 km from 0.1 km to 1 km, maintaining a CC of 0.5 and without NR. Afterwards, CC levels were varied in intervals of 0.1 from 0.1 to 1, fixing neighbourhood Rad in 0.5 km, and also without NR. Finally, with NR activated and a Rad settled to 0.5 km, experiments with the following CC levels were carried: $CC \in \{0.25, 0.5, 0.75, 1\}$. Experiments were also taken with UI and without repositioning for CC=1 and Rad=1, with the objective of exploring maximum ranges.

To compare results, three metrics were applied:

- Service level - A measure of service quality provided by the bike sharing system, also used in Pfrommer et al. [5], Singla et al. [35] (see section 3.2);
- Average extra effort (km) - Result of summing the extra meters travelled to the alternative station;
- Lost user percentage - A measure of major importance since there is no available data hold by Gira regarding people who leave the system due to empty stations.

The impact of user incentive methods on repositioning (4 trucks) was measured by:

- Visited stations per day
- Bikes transported per day
- Total travelled distance per day (km)
- Sum of all operation times (minutes)
- CPU time per repositioning instructions calculus (seconds)

Before performing the experiments, it was analysed for how long an experiment should be performed. Figure 6.6 represents the service level of an experiment without user incentives and repositioning. A visual examination of it shows a stabilisation around 15th to 20th day. To have a little margin, 28 straight days, for each simulation, were experimented. All experiments were carried four times, therefore, the following results represent the average of four values.

6.5 Service level

Service level is a measure based on the number of customers and the number of no service events. Section 3.2 briefly explains and shows the service level equation 3.6.

BSS service level for different values of Rad is shown in figure 6.8, and the influence of varying CC is represented in figure 6.7. For both cases, experiments were carried without repositioning.

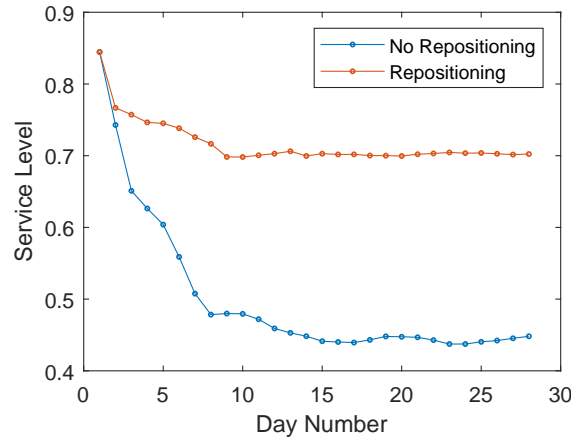


Figure 6.6: Service level over 28 days

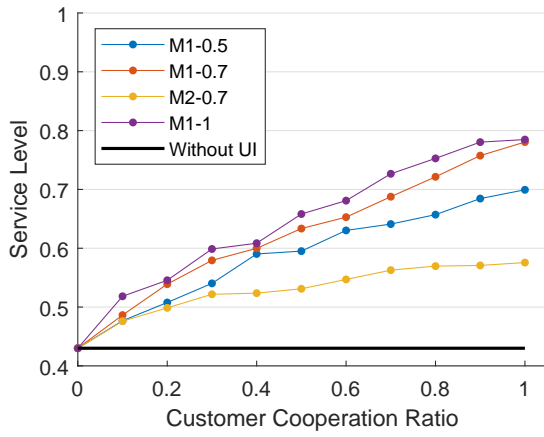


Figure 6.7: Service level (without NR; Rad=0.5 km)

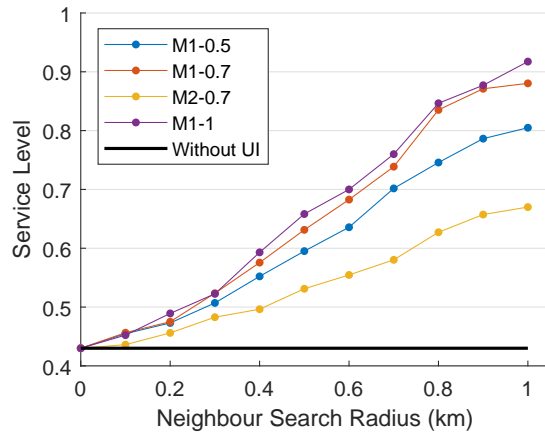


Figure 6.8: Service level (without NR; CC=0.5)

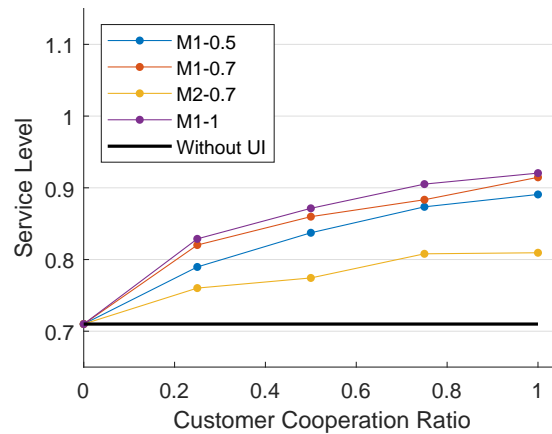


Figure 6.9: Service level (with NR; Rad=0.5 km)

Without user incentives, a service level of 0.43 was achieved, the minimum service level found. The highest level was obtained by the M1-1.0 with CC=1.0 and Rad=1.0 which achieved a level of 0.99, showing that it is possible to obtain a system totally balanced without repositioning, as long as all users accept to participate.

It is clear how an increase in the Rad and CC also increases service level. However, this behaviour is not the same for every scenario tested. M2-0.7 has the worst results, with a difference of only 0.2 when varying Rad from 0 to 1 km. M1 curves have a higher service level when compared to M2, being also evident that a higher WPC shows higher service level values. Nevertheless, it is visible how WPC's influence decreases, with the difference from M1-0.5 to M1-0.7 much higher than the difference from M1-0.7 to M1-1.0.

Although these values indicate a good performance of the methods, they are low when compared to the ones obtained in Aeschbach et al. [6]. M1-0.5 with CC=0.5 and Rad=0.5 km, an intermediate case, obtained a service level of 0.6 a difference of 0.16 to the no UI case, but obtained a value of about 0.83 in Aeschbach et al. [6]. This considerable difference can be related to the fact that this thesis implemented three cases of cooperation, unlike Aeschbach et al. [6] who considered only one — participate in both departure and arrival stations. Another possibility is the number of days simulated. This thesis took 28 straight days of simulation, since Aeschbach et al. [6] did not identify the number of consecutive days of simulation, these values cannot be directly compared. Figure 6.6 shows how this impacts service level. The last possibility for this discrepancy is the BSS used for analysis. In this thesis, data from Gira including 74 stations was analysed, while Aeschbach et al. [6] used data from London's Barclays Cycle Hire (nowadays Santander Cycle Hire) with 745 stations (at that time). Even though this was not studied thoroughly, it may be the case that there is a higher number of neighbouring stations around a London BSS's station than around a Lisbon BSS's station for the same neighbouring Rad, a fact that would have an impact on the best stations chosen by the method, consequently uniformly spreading bikes around a higher number of stations.

Figure 6.9 shows the service level for different cooperation values with NR active. As expected, service level saw a great increase for all methods, with the behaviours between them remaining unchanged. For example, M1-0.5 with CC=0.5 and Rad=0.5 went from 0.6 without repositioning to 0.86 with repositioning, getting a value of 0.71 when repositioning is activated and UI is inactive. These values represent a relevant increase relatively to experiments without repositioning, showing how important is rebalancing stations through trucks, and why this methodology should not be discarded, but instead cooperate with UI methods.

	Scenario			
	M1-0.5	M1-0.7	M1-1.0	M2-0.7
Service Level	0.95	1	1	0.78

Table 6.2: Average service level (without NR; CC=1.0; Rad=1.0 km)

6.6 Average extra effort

Asking customers to change their journey implies an extra effort by cooperative users. This effort was measured by equation 3.7 presented in section 3.2.

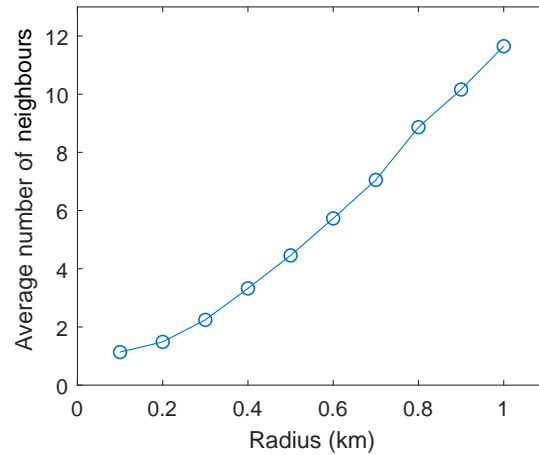


Figure 6.10: Average number of neighbours around a station

The results for the extra effort expression are depicted in figures 6.11, 6.12 and 6.13. An increase on average extra effort over Rad was identified for all scenarios, but not over CC. This was the expected outcome, given that only an increase in the size of the neighbourhood should result in higher efforts due to larger distances from the intended stations. An interesting fact to highlight is that calculated values are, for some cases, higher than the Rad, for example, M1-0.5 has an average extra effort of 0.6. This occurs because the resultant effort is the addition of effort for both start and end stations, which implies a maximum extra effort fixed on 2 times the Rad. When methods apply a Rad of 0.5 km, they are saying that people, at most, will add 1 km (straight line) to its journey.

When it comes to the methods applied, M1 clearly implies a higher effort over M2. This is due to the fact that M2 shows as an alternative station the one closest to the intended; unlike M1 that indicates the one with the best level of occupancy. This way, unlike M2, M1 will have a higher tendency to spread bikes uniformly.

Differences are only visible between M1 and M2, because variations of WPC inside M1 do not seem to have an impact on this metric. The same goes for experiments with repositioning, for which values with and without repositioning appear to be the same, such as the example of (M1-0.5, CC=0.5, Rad=0.5, no repositioning) with a value of 0.53 km, and 0.52 km with repositioning but the same CC and Rad parameters.

	Scenario			
	M1-0.5	M1-0.7	M1-1.0	M2-0.7
Extra Effort (km)	1.13	1.25	1.24	0.53

Table 6.3: Average extra effort (without NR; CC=1.0; Rad=1.0 km)

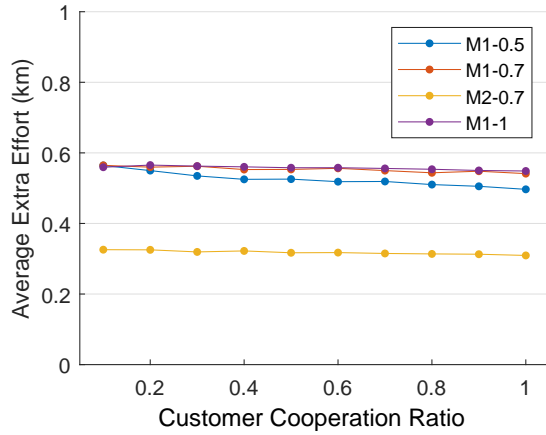


Figure 6.11: Average extra effort (without NR; Rad=0.5)

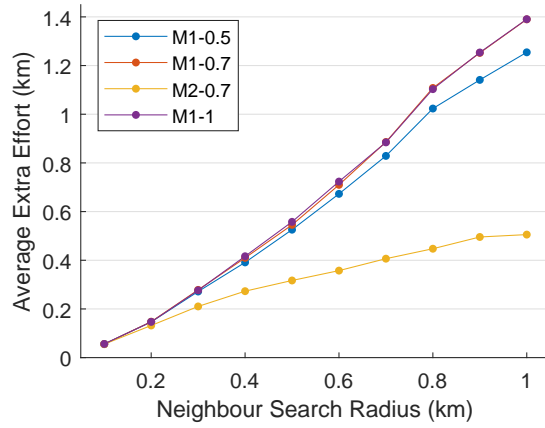


Figure 6.12: Average extra effort (without NR; CC=0.5)

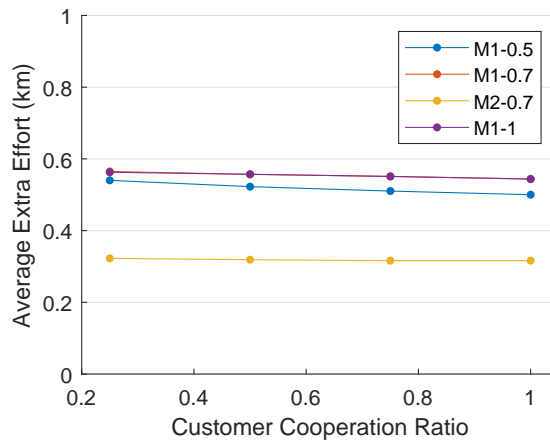


Figure 6.13: Average extra effort (with NR; Rad=0.5 km)

6.7 Lost users

From section 6.3 it is already known that there is always a percentage of users that will be lost when they try to enter the system. As explained in previous sections, there is no data showing how many users are lost in real case scenario; however, according to Gira's management team, if a station is empty/full for more than 15 minutes and all its neighbours within a radius of 100m are in the same situation, this station should be repositioned as fast as possible. Although this information could not be confirmed with the studied data, it indicates a low probability of this type of occurrences for long periods of time. Taking this into account, the whole simulation was modelled considering that there were no more users trying to enter the system beyond the users (trips) registered.

Inversely to service level, as shown by figures 6.14 and 6.15, the percentage of lost users decreases with CC and Rad. As displayed by figure 6.16, NR, in this case, also has a huge impact, decreasing these percentages.

Similarly to the case for previous metrics, M2 was the worst method, always showing higher percentages

of lost users.

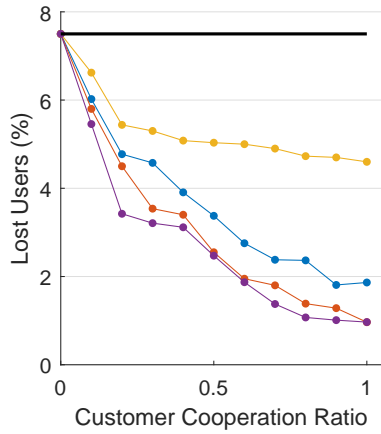


Figure 6.14: Percentage of lost users (without NR; Rad=0.5 km)

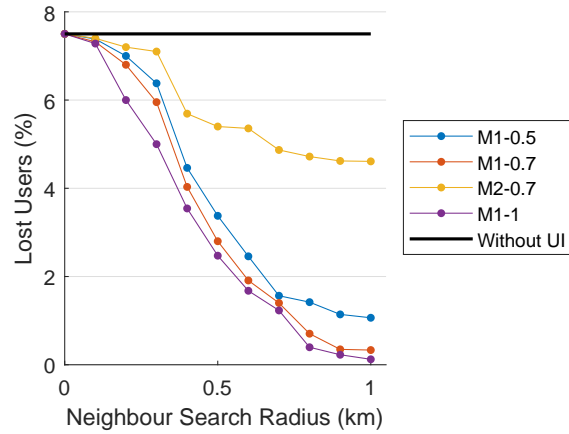


Figure 6.15: Percentage of lost users (without NR; CC=0.5)

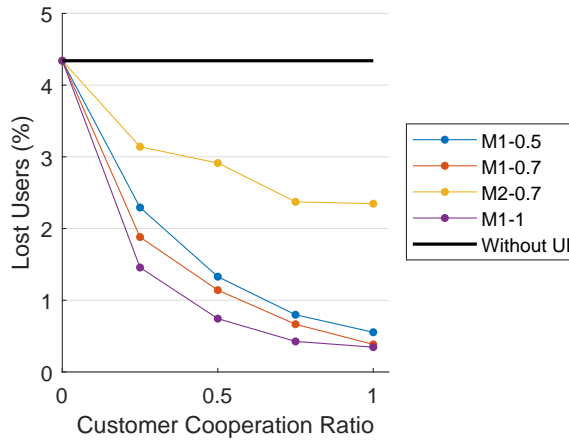


Figure 6.16: Percentage of lost users (with NR; Rad=0.5 km)

Lost Users (%)	Scenario			
	M1-0.5	M1-0.7	M1-1.0	M2-0.7
	0.05	0.04	0.05	1.63

Table 6.4: Average percentage of lost users (without NR; CC=1.0; Rad=1.0 km)

6.8 Repositioning results

Lastly, results related to repositioning are presented in table 6.5. A general analysis shows that, just like previous metrics, M1-1.0 is the best, followed by M1-0.7, then M1-0.5 and finally M2-0.7.

The number of visited stations is reduced with the decrease of cooperation levels for all scenarios, and may be reduced to even less than 11 stations when comparing no repositioning (42 stations) to M1-1.0 with CC=1 (31 stations).

Table 6.5 also shows that the number of bikes transported is on average 2 to 2.5 times the number of visited stations. Considering that almost half of the stations are receiving bikes and the other half losing bikes, on average, there are 4 to 5 bikes dropped or picked up per station visited.

When it comes to the total distance travelled and operation time, the behaviour is the same as previous measures, starting with 143 km for 445 minutes without UI, and dropping to values as low as 119 km travelled for 350 minutes (M1-0.5, CC=1), therefore, showing the influence of UI on repositioning.

<i>Scenario</i>	<i>CC</i>	<i>Visited stations</i>	<i>Bikes transported</i>	<i>Distance traveled (km)</i>	<i>Operation time (minutes)</i>	<i>CPU time (seconds)</i>
No UI		42.43	106.79	143.16	445.04	44.58
M1-0.5	0.25	39.17	91.61	134.57	436.80	20.41
	0.5	37.79	85.30	133.93	422.54	31.48
	0.75	36.46	78.40	131.45	404.89	27.39
	1	35.15	69.63	127.62	382.06	15.52
M1-0.7	0.25	38.79	89.13	135.55	432.13	30.14
	0.5	36.68	79.52	132.38	409.03	28.64
	0.75	34.14	71.96	129.43	388.74	31.31
	1	31.49	61.38	121.08	353.08	23.93
M1-1.0	0.25	37.84	84.37	136.67	426.02	35.47
	0.5	35.96	78.81	130.23	403.46	24.04
	0.75	33.13	69.86	123.97	375.54	22.92
	1	31.05	60.72	118.82	350.06	10.34
M2-0.7	0.25	38.23	90.82	136.14	436.75	27.11
	0.5	37.96	89.61	136.03	435.07	38.61
	0.75	35.70	81.78	130.22	409.85	25.14
	1	35.08	80.45	129.89	406.04	24.77

Table 6.5: Repositioning results with 4 trucks

Chapter 7

Conclusions and future research

7.1 Conclusions

This dissertation consists of the development of a functional simulator of a Gira BSS in Anylogic. This simulation allows the study of repositioning schemes and user incentive methods, and comprises a visual interface where people, trucks and bikes can be seen travelling between stations all day. Besides observing trips on a map, the software user can change determined parameters like the number of trucks, truck's capacity, CC or Rad, as well as the methods applied.

In the simulator a nightly repositioning method (implementation of Schuijbroek et al. [4]'s work) and two user incentive methods are incorporated. The first method follows the work of Aeschbach et al. [6] and the second was originally modelled in the course of this project, in the pursue of reproducing the UI method currently applied to Gira's users.

Real data from Lisbon's BSS has validated the simulator components, such as the number of trips generated at each 20-minutes, trip time and repositioning tasks. The generation of trips was mainly validated by not considering the station's capacity or lack of bikes. Even so, when solely applying nightly repositioning and station constraints, low errors were still reached, with the number of lost users being drastically reduced when compared to the case of no repositioning. Results have shown that nightly repositioning is not enough to keep the system balanced through the entire day, and only a dynamic approach could handle all the demand.

The results from varying CC and Rad with and without repositioning have shown that customers are able to balance the system without trucks, but only if all cooperate, and a Rad of 1 km is applied. Otherwise, no-service events will occur with a much higher frequency. Therefore, in case not all the users cooperate, the first conclusion to take is that truck repositioning is essential and cannot be withdrawn.

M1, presented in Aeschbach et al. [6] work, has shown evidence of a higher service level and lower lost user percentage results than M2, for all WPC ratios. Unlike M2 that only chooses stations with occupation percentages lower/higher than 30/70, M1 indicates the best station in the neighbourhood to

the app user, even if the occupation does not lay in the required interval. Taking this fact into account, M1 will always have more alternative stations to present than the other method, therefore achieving better results. The only factor in which M2 showed better results was in the amount of extra effort. This was the expected behaviour considering that M2 searches for the nearest station that satisfies the requirements, while the other method seeks the best station in the neighbourhood.

Even though repositioning cannot be withdrawn, results have also shown that UI methods have reduced the average number of stations to visit per day, and the number of bikes to transport. This is, therefore, a way of reducing the transportation's costs and increasing the amount of free time to spend on other tasks.

7.2 Future work

Given the complexity of BSSs, there are still several ways to enrich and complement the analysis performed in this dissertation:

- New studies can use this simulator to compare different UI or repositioning methods. The most obvious is to implement a dynamic repositioning method, which would be a useful study to strengthen the conclusions of this work. An alternative UI method should, for example, consider price incentives, therefore allowing the computation of the costs associated.
- Questions such as “How much money is necessary to get people to participate?” or “How much money will be saved on repositioning tasks compared to what is spent with UI?” are yet to be answered. Thus, one way to get more information is by conducting surveys on Lisbon BSS's users, in an attempt to find their willingness to change their journey and what they demand in exchange.
- This work considered a certain number of docks and bikes within the system that have been kept the same for all experiments. As such, future studies could asses how changes in these values would impact the results.
- A differentiation between classic and electric bikes is another path to take.
- In addition, effects of season, weather, events, etc., were not analysed in this thesis and could be subject to further analysis.

Bibliography

- [1] Promoting environmentally sustainable transport (est). *United Nations - Sustainable Development*, (Accessed July, 2021). URL: <https://sustainabledevelopment.un.org/partnership/?p=365>.
- [2] R. B. Nath and T. Rambha. Modelling methods for planning and operation of bike-sharing systems. *Journal of the Indian Institute of Science*, pages 1–26, 2019.
- [3] M. A. Babagoli, T. K. Kaufman, P. Noyes, and P. E. Sheffield. Exploring the health and spatial equity implications of the new york city bike share system. *Journal of transport & health*, 13:200–209, 2019.
- [4] J. Schuijbroek, R. C. Hampshire, and W.-J. Van Hoes. Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, 257(3):992–1004, 2017.
- [5] J. Pfrommer, J. Warrington, G. Schilbach, and M. Morari. Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1567–1578, 2014.
- [6] P. Aeschbach, X. Zhang, A. Georgiou, and J. Lygeros. Balancing bike sharing systems through customer cooperation-a case study on london’s barclays cycle hire. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 4722–4727. IEEE, 2015.
- [7] D. Chemla, F. Meunier, T. Pradeau, R. W. Calvo, and H. Yahiaoui. Self-service bike sharing systems: simulation, repositioning, pricing. 2013. <hal-00824078>.
- [8] E. Maibach, L. Steg, and J. Anable. Promoting physical activity and reducing climate change: Opportunities to replace short car trips with active transportation. *Preventive medicine*, 49(4):326–327, 2009.
- [9] P. Midgley. Bicycle-sharing schemes: enhancing sustainable mobility in urban areas. *United Nations, Department of Economic and Social Affairs*, 8:1–12, 2011.
- [10] S. A. Shaheen, S. Guzman, and H. Zhang. Bikesharing in europe, the americas, and asia: past, present, and future. *Transportation Research Record*, 2143(1):159–167, 2010.
- [11] D. Gutman. Will helmet law kill seattle’s new bike-share program? *The Seattle Times*, 2016, (Accessed July, 2021). URL: <https://www.seattletimes.com/seattle-news/transportation/will-helmet-law-kill-seattles-new-bike-share-program/>.

- [12] I. Otero, M. Nieuwenhuijsen, and D. Rojas-Rueda. Health impacts of bike sharing systems in europe. *Environment international*, 115:387–394, 2018.
- [13] N. McIntyre and J. Kollwe. Life cycle: is it the end for britain's dockless bike schemes. *The Guardian*, February 2019, (Accessed July, 2021). URL: <https://www.theguardian.com/cities/2019/feb/22/life-cycle-is-it-the-end-for-britains-dockless-bike-schemes>.
- [14] G. Paton. Ofo bike-sharing scheme abandoned amid vandalism and rising costs. *The Times*, January 2019, (Accessed July, 2021). URL: <https://www.thetimes.co.uk/article/ofo-bike-sharing-scheme-abandoned-amid-vandalism-and-rising-costs-hwbhkh3mh>.
- [15] N. Menezes. Why bicycling in bengaluru is a cruel joke on cyclists. *The Economic Times*, May 2019, (Accessed July, 2021). URL: <https://economictimes.indiatimes.com/news/politics-and-nation/why-bicycling-in-bengaluru-is-a-cruel-joke-on-cyclists/articleshow/69157348.cms>.
- [16] C. Bonnington. High-tech handlebars and anti-theft alerts. *Slate*, February 2018, (Accessed July, 2021). URL: <https://slate.com/technology/2018/02/how-to-prevent-bike-theft-with-high-tech-handlebars-and-other-anti-theft-gadgets.html>.
- [17] T. Brookes. The 4 best bike trackers for catching thieves red handed. *Make Use Of*, March 2019, (Accessed July, 2021). URL: <https://www.makeuseof.com/tag/bike-tracker-catching-thieves/>.
- [18] M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, 2014.
- [19] M. Rainer-Harbach, P. Papazek, G. R. Raidl, B. Hu, and C. Kloimüller. Pilot, grasp, and vns approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization*, 63(3):597–629, 2015.
- [20] C. Fricker and N. Gast. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *Euro journal on transportation and logistics*, 5(3):261–291, 2016.
- [21] Passes and tariffs. *Gira*, (Accessed July, 2021). URL: <https://www.gira-bicicletasdelisboa.pt/passes-e-tarifarios/>.
- [22] S. Ghosh and P. Varakantham. Incentivizing the use of bike trailers for dynamic repositioning in bike sharing systems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pages 373–381. AAAI Press, 2017.
- [23] S. P. S. Melo. Rebalanceamento estático do sistema de bicicletas públicas partilhadas de lisboa-gira. Master's thesis, Instituto Superior de Economia e Gestão, 2018.
- [24] M. C. Rodrigues. Planning nightly rebalancing in bike sharing systems. Master's thesis, Instituto Superior Técnico, June 2019.

- [25] G. Erdoğan, G. Laporte, and R. W. Calvo. The static bicycle relocation problem with demand intervals. *European Journal of Operational Research*, 238(2):451–457, 2014.
- [26] H. Hernández-Pérez and J.-J. Salazar-González. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1):126–139, 2004.
- [27] H. Hernández-Pérez and J.-J. Salazar-González. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks: An International Journal*, 50(4):258–272, 2007.
- [28] M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, and L. Robinet. Balancing the stations of a self service “bike hire” system. *RAIRO-Operations Research*, 45(1):37–61, 2011.
- [29] S. C. Ho and W. Szeto. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, 69:180–198, 2014.
- [30] J. Schuijbroek, R. Hampshire, and W. van Hoes. Inventory rebalancing and vehicle routing in bike sharing systems. In *Tech. Rep. 2013-E1*. Tepper School of Business, Carnegie Mellon University, 2013.
- [31] D. Chemla, F. Meunier, and R. W. Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146, 2013.
- [32] I. A. Forma, T. Raviv, and M. Tzur. A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transportation research part B: methodological*, 71:230–247, 2015.
- [33] S. Ghosh, P. Varakantham, Y. Adulyasak, and P. Jaillet. Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research*, 58:387–430, 2017.
- [34] C. S. Shui and W. Szeto. Dynamic green bike repositioning problem—a hybrid rolling horizon artificial bee colony algorithm approach. *Transportation Research Part D: Transport and Environment*, 60:119–136, 2018.
- [35] A. Singla, M. Santoni, G. Bartók, P. Mukerji, M. Meenen, and A. Krause. Incentivizing users for balancing bike sharing systems. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, volume 1, pages 723–729, 2015.
- [36] Z. Haider, A. Nikolaev, J. E. Kang, and C. Kwon. Inventory rebalancing through pricing in public bike sharing systems. *European Journal of Operational Research*, 270(1):103–117, 2018.
- [37] L. Pan, Q. Cai, Z. Fang, P. Tang, and L. Huang. A deep reinforcement learning framework for rebalancing dockless bike sharing systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1393–1400, 2019.

- [38] S. J. Patel, R. Qiu, and A. Negahban. Incentive-based rebalancing of bike-sharing systems. In *INFORMS International Conference on Service Science*, pages 21–30. Springer, 2018.
- [39] A. Fernández, H. Billhardt, S. Ossowski, and Ó. Sánchez. Bike3s: A tool for bike sharing systems simulation. *Journal of Simulation*, pages 1–17, 2020.
- [40] L. Caggiani and M. Ottomanelli. A modular soft computing based method for vehicles repositioning in bike-sharing systems. *Procedia-Social and Behavioral Sciences*, 54:675–684, 2012.
- [41] S. Ji, C. R. Cherry, L. D. Han, and D. A. Jordan. Electric bike sharing: simulation of user demand and system availability. *Journal of Cleaner Production*, 85:250–257, 2014.
- [42] T. J. P. Dubernet and K. W. Axhausen. A multiagent simulation framework for evaluating bike redistribution systems in bike sharing schemes. *Arbeitsberichte Verkehrs-und Raumplanung*, 1010, 2014.
- [43] R. M. Saltzman and R. M. Bradford. Simulating a more efficient bike sharing system. *Journal of Supply Chain and Operations Management*, 14(2):36–47, 2016.
- [44] N. Jian, D. Freund, H. M. Wiberg, and S. G. Henderson. Simulation optimization for a large-scale bike-sharing system. In *2016 Winter Simulation Conference (WSC)*, pages 602–613. IEEE, 2016.
- [45] F. Soriguera, V. Casado, and E. Jiménez. A simulation model for public bike-sharing systems. *Transportation Research Procedia*, 33:139–146, 2018.
- [46] A. Waserhole and V. Jost. Pricing in vehicle sharing systems: Optimization in queuing networks with product forms. *EURO Journal on Transportation and Logistics*, 5(3):293–320, 2016.
- [47] F. A. Haight. *Handbook of the Poisson distribution*. John Wiley & Sons, 1967.
- [48] R. D. Yates and D. J. Goodman. *Probability and stochastic processes: a friendly introduction for electrical and computer engineers*. John Wiley & Sons, 2014.
- [49] Expectation of poisson distribution. *Proof Wiki*, (Accessed July, 2021). URL: https://proofwiki.org/wiki/Expectation_of_Poisson_Distribution.
- [50] Variance of poisson distribution. *Proof Wiki*, (Accessed July, 2021). URL: https://proofwiki.org/wiki/Variance_of_Poisson_Distribution.
- [51] Poisson distribution. *Wikipedia*, (Accessed July, 2021). URL: https://en.wikipedia.org/wiki/Poisson_distribution.
- [52] F. Galton. Psychometric experiments. *Brain*, 2(2):149–162, 1879.
- [53] D. McAlister. XIII. the law of the geometric mean. In *Proceedings of the Royal Society of London*, volume 29, pages 367–376. The Royal Society London, 1879.
- [54] N. Balakrishnan and W. Chen. *Handbook of tables for order statistics from lognormal distributions with applications*. Springer Science & Business Media, 1999.

- [55] P.-T. Yuan. On the logarithmic frequency distribution and the semi-logarithmic correlation surface. *The Annals of Mathematical Statistics*, 4(1):30–74, 1933.
- [56] Log-normal distribution. *Wikipedia*, (Accessed July, 2021). URL: https://en.wikipedia.org/wiki/Log-normal_distribution.
- [57] P. Hulot, D. Aloise, and S. D. Jena. Towards station-level demand prediction for effective rebalancing in bike-sharing systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 378–386, 2018.
- [58] J. G. Skellam. The frequency distribution of the difference between two poisson variates belonging to different populations. *Journal of the Royal Statistical Society: Series A*, 109:296, 1946.
- [59] J. Strackee and J. D. van der Gon. The frequency distribution of the difference between two poisson variates. *Statistica Neerlandica*, 16(1):17–23, 1962.
- [60] Sobre a gira - gira - bicicletas de lisboa. (Accessed July, 2021). URL: <https://www.gira-bicicletasdelisboa.pt/sobre-a-gira/>.
- [61] Emel lança bike sharing. September 2017, (Accessed July, 2021). URL: <https://www.gira-bicicletasdelisboa.pt/noticias/emel-lanca-bike-sharing/>.
- [62] Rede de bicicletas partilhadas de lisboa começa a funcionar na terça-feira. September 2017, (Accessed July, 2021). URL: <https://24.sapo.pt/atualidade/artigos/rede-de-bicicletas-partilhadas-de-lisboa-comeca-a-funcionar-na-terca-feira>.
- [63] V. Lucas and A. R. Andrade. Predicting hourly origin-destination demand in bike sharing systems using hurdle models: Lisbon case study. *Case Studies on Transport Policy*, 2021. ISSN 2213-624X. doi: <https://doi.org/10.1016/j.cstp.2021.10.003>. URL <https://www.sciencedirect.com/science/article/pii/S2213624X21001656>.
- [64] C. Etienne and O. Latifa. Model-based count series clustering for bike sharing system usage mining: a case study with the vélib’system of paris. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):1–21, 2014.
- [65] D. Gervini and M. Khanal. Exploring patterns of demand in bike sharing systems via replicated point process models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 68(3): 585–602, 2019.
- [66] F. Sun, P. Chen, and J. Jiao. Promoting public bike-sharing: A lesson from the unsuccessful pronto system. *Transportation Research Part D: Transport and Environment*, 63:533–547, 2018.
- [67] *Anylogic*, (Accessed July, 2021). URL: <https://www.anylogic.com>.
- [68] D. Ljubenkov, F. Kon, and C. Ratti. Optimizing bike sharing system flows using graph mining, convolutional and recurrent neural networks. In *2020 IEEE European Technology and Engineering Management Summit (E-TEMS)*, pages 1–6. IEEE, 2020.

- [69] What is a tour de france pro's average speed and how do your stats compare? *Gira*, (Accessed July, 2021). URL: https://www.gira-bicicletasdelisboa.pt/termos_condicoes.pdf.
- [70] What is a tour de france pro's average speed and how do your stats compare? *Gira*, (Accessed July, 2021). URL: <https://www.gira-bicicletasdelisboa.pt/perguntas-frequentes/>.
- [71] W. Yost. What is a tour de france pro's average speed and how do your stats compare? *bicycling*, (Accessed July, 2021). URL: <https://www.bicycling.com/racing/a20037750/you-versus-a-tour-de-france-pro-cyclist/>.

Appendix A

A.1 Trip time figures

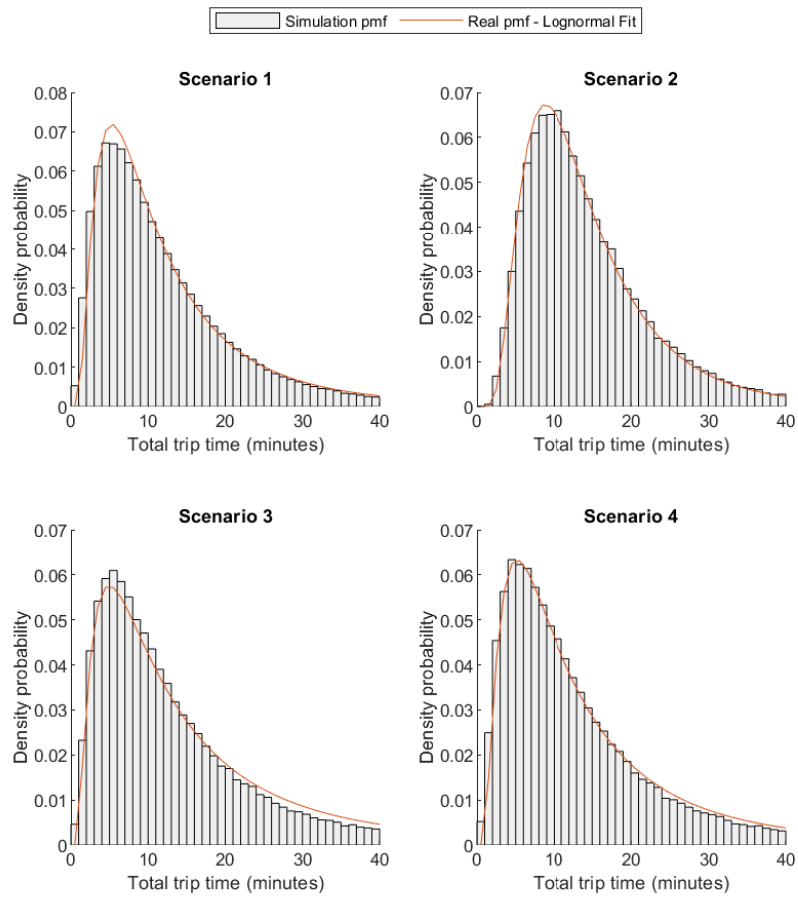


Figure A.1: Real PMF (Lognormal fit) versus simulation PMF of trip time (all trips)

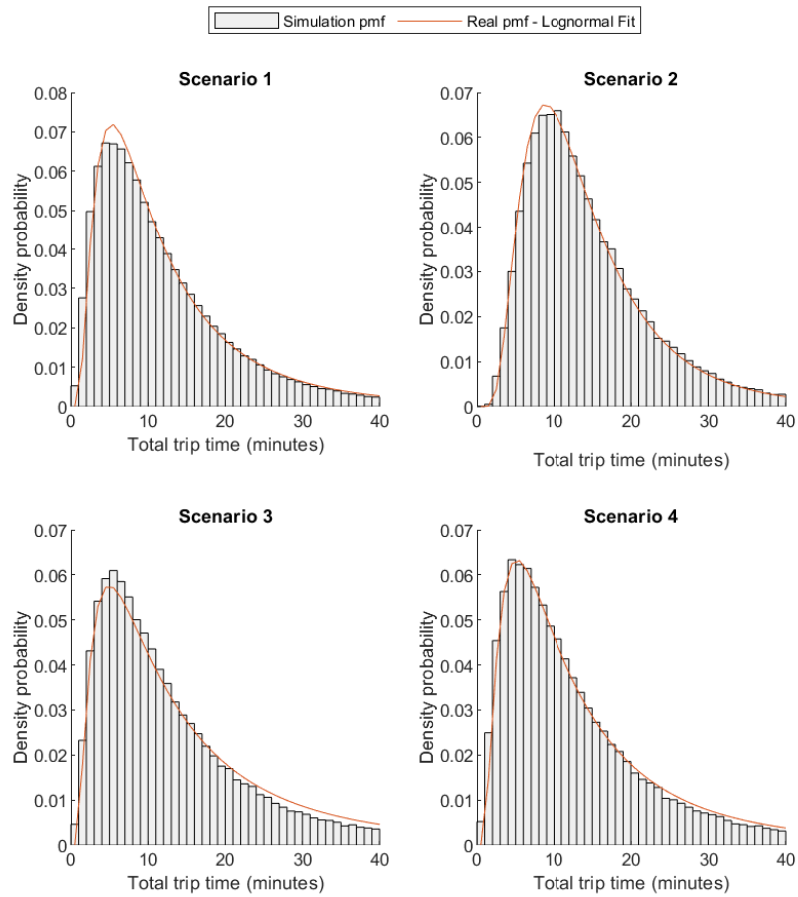


Figure A.2: Real PMF (Lognormal fit) versus simulation PMF of trip time (trips from 101 to 104)

A.2 Repositioning results

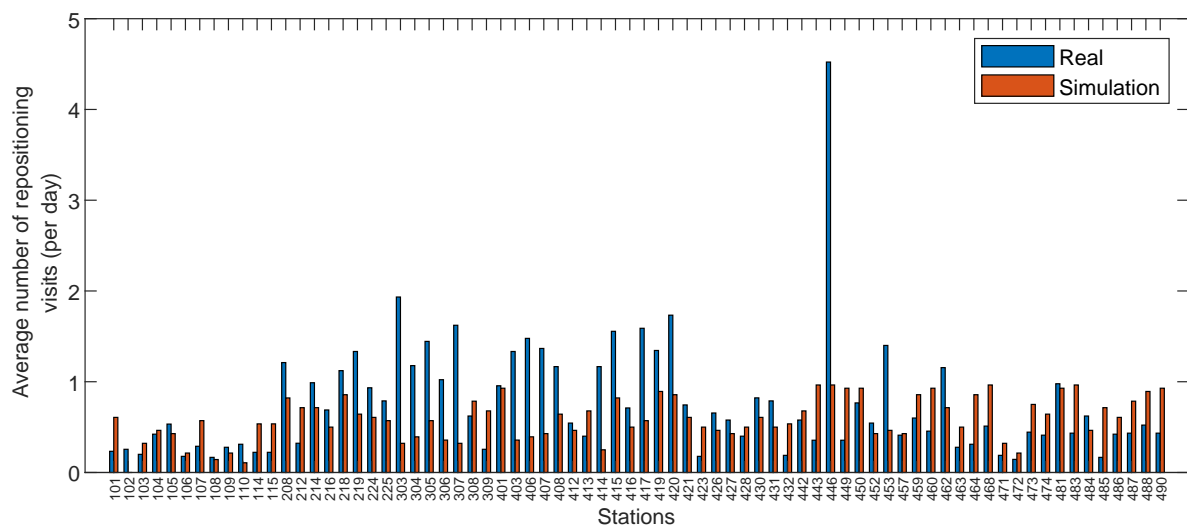


Figure A.3: Real versus simulation average number of repositioning visits per day at each station

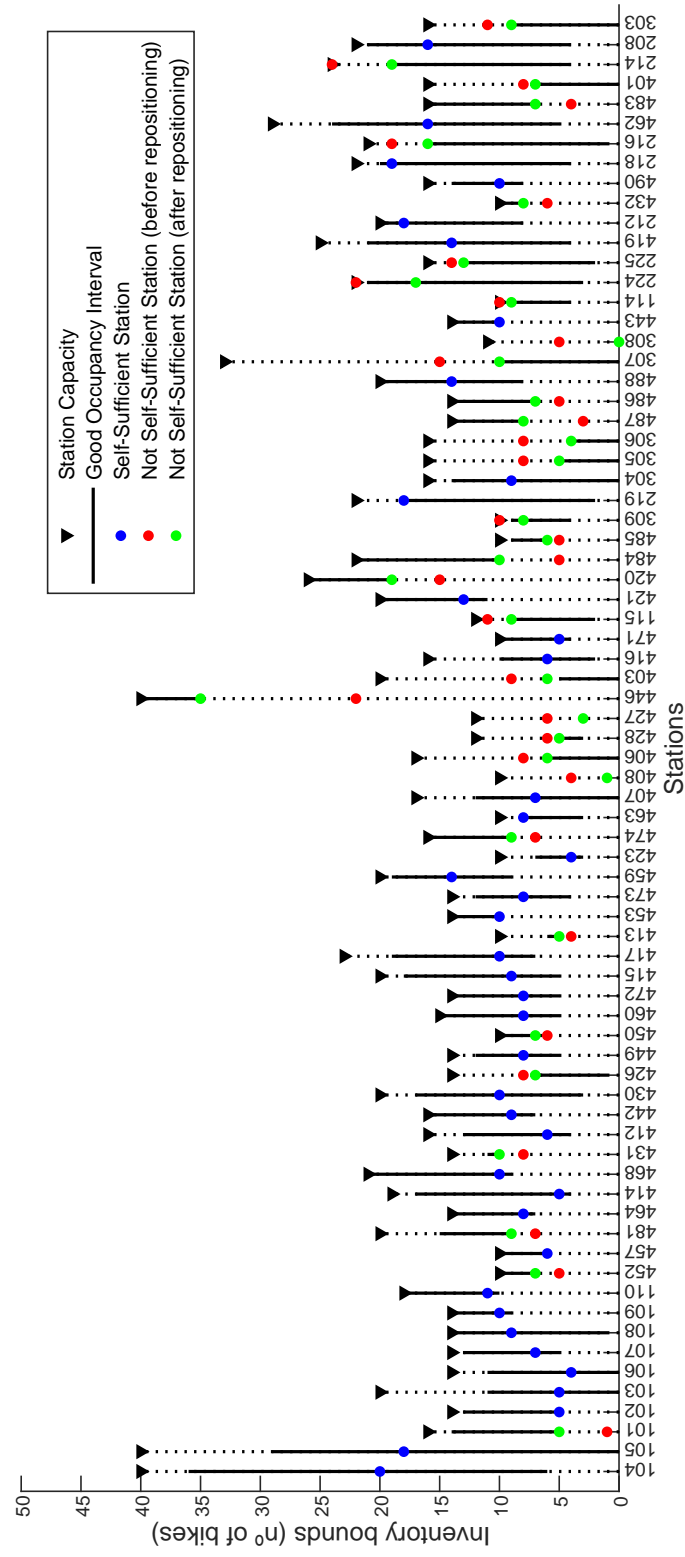


Figure A.4: Example of stations' level before and after repositioning

A.3 Agents

Main



Figure A.5: Agent Main in Anylogic

Parameter	Description
TruckSpeed	Truck speed
PersonSpeed	Person speed
UI	Indicates which user incentive method is used: 0 - no UI 1 - method 1 2 - method 2
NR	Indicates if nightly repositioning is activated
SR	Indicates if station restrictions are activated or not
Data	Indicates which file is used as input data ($\lambda_w(i, j, t)$)
Radius	Neighbourhood radius
CustCoop	Customer cooperation ratio
wpcPerc	Ratio that indicates if stations primarily chosen by customers could be changed or not. When a UI method is active and a customer insert a start/end station with an occupation ratio higher/lower than wpcPerc/1-wpcPerc, than customer's choice is fine and the app will not give an alternative station. If it is lower/higher than wpcPerc/1-wpcPerc than the method will try to find a better station within specified "Radius".
NbrVehc	Number of trucks used on tasks of repositioning
TruckCapacity	Maximum number of bikes truck is able to transport

Table A.1: Principal parameters of agent Main

Station

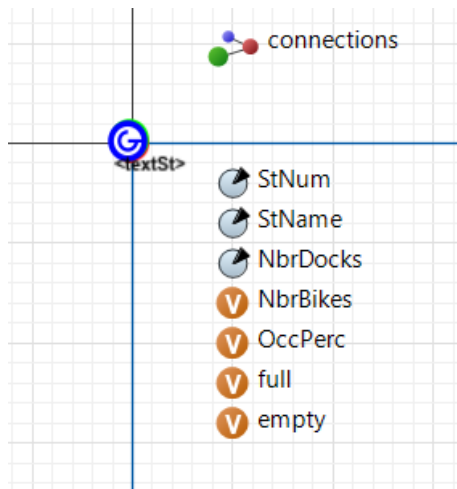


Figure A.6: Agent Station in Anylogic

Parameter	Description
StNum	Station index number
StName	Station code
NbrBikes	Number of bikes at the station
OccPerc	Station occupation percentage
full	Boolean variable that indicates if the station is full
empty	Boolean variable that indicates if the station is empty

Table A.2: Principal parameters of agent Station

Depot

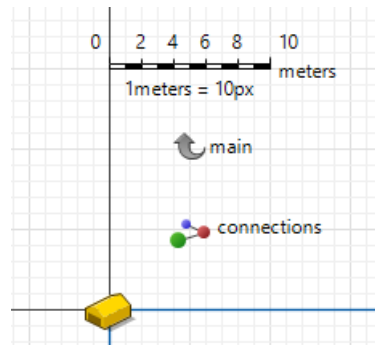


Figure A.7: Agent Depot in Anylogic

Person

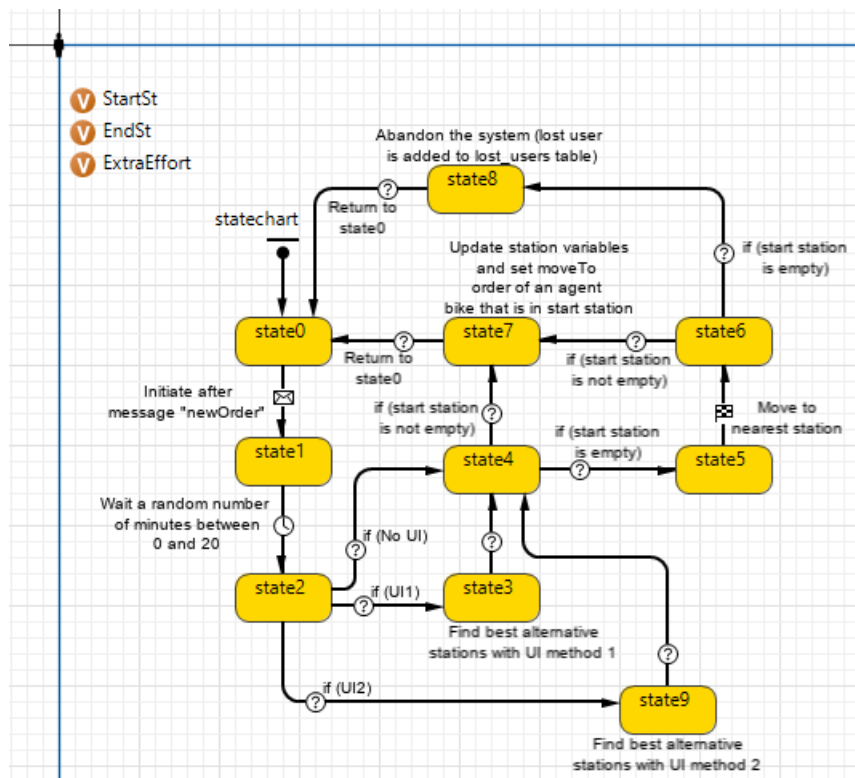


Figure A.8: Agent Person in Anylogic

Parameter	Description
StartSt	Trip start station
EndSt	Number of bikes at the station
ExtraEffort	Extra meter travelled by customer due to UI methods

Table A.3: Principal parameters of agent Person

Bike

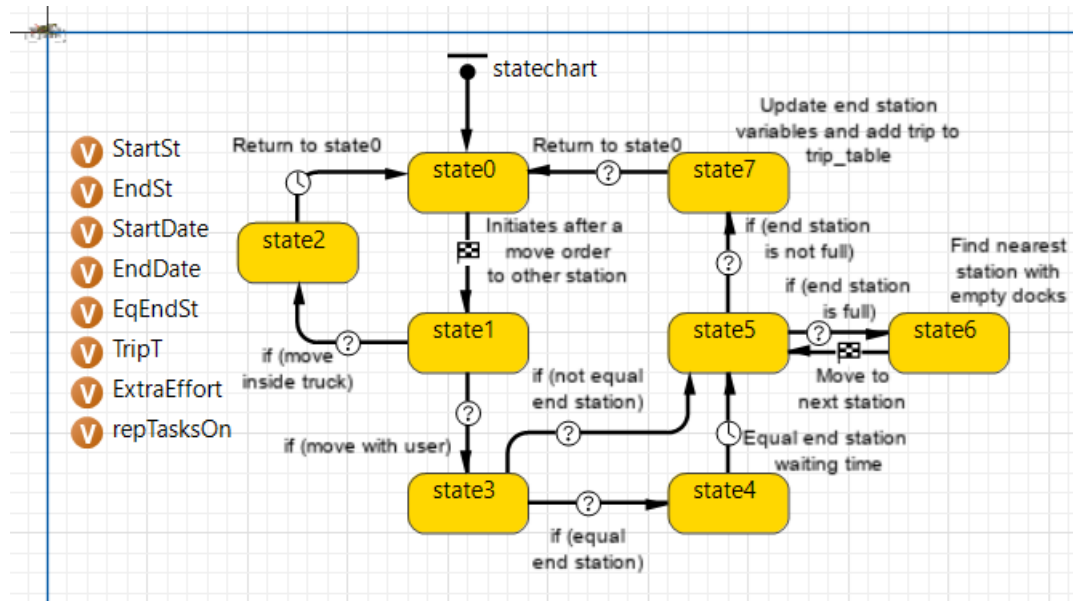


Figure A.9: Agent Bike in Anylogic

Parameter	Description
StartSt	Trip start station
EndSt	Trip end station
StartDate	Trip start date
EndDate	Trip end date
EqEndSt	Boolean variable that indicate if start and end station are the same
TripT	Trip time
ExtraEffort	Extra meter travelled by customer due to UI methods
repTasksOn	Boolean variable that indicates if trucks are repositioning bikes

Table A.4: Principal parameters of agent Bike

Truck

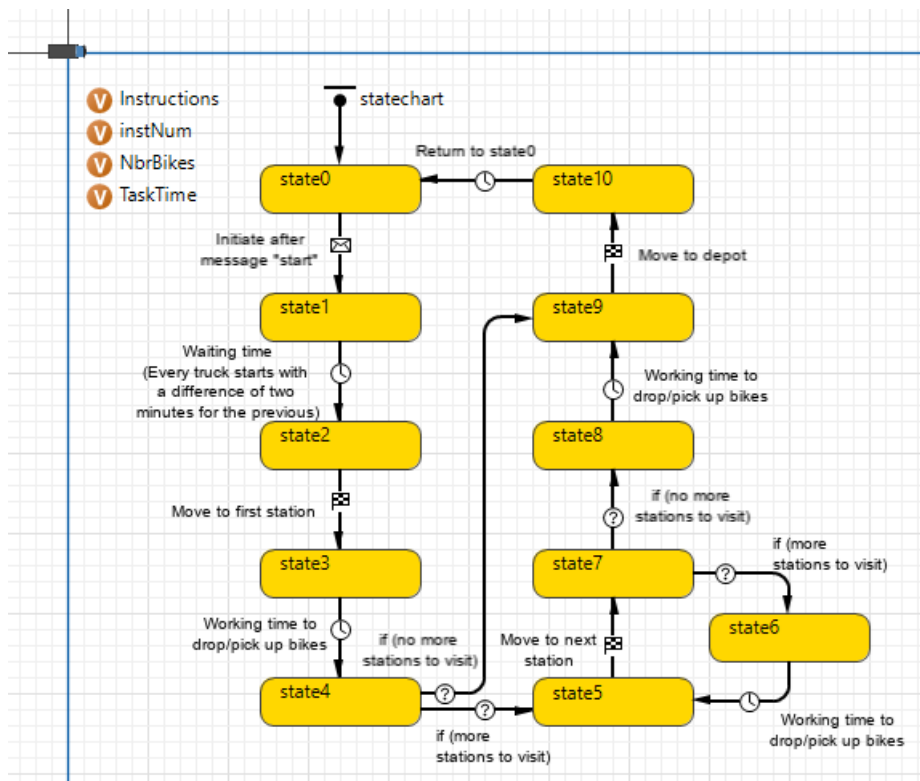


Figure A.10: Agent Truck in Anylogic

Parameter	Description
Instructions	Matrix that indicates stations to visit and number of bikes to pick up or drop at that station
instNum	Instruction number
NbrBikes	Number of bikes of bikes to pick up or drop at the current station
TaskTime	Task time. It is equal to one minute times the number of bikes to drop or pick up.

Table A.5: Principal parameters of agent Truck

A.4 Principal simulator codes in Java

Create Trips

Code implemented in agent Main and inside event: *CreateTrips*.

```
// Iterate through all stations combinations
for (int StartSt = 0; StartSt < NbrSt; StartSt++) {
    for (int EndSt = 0; EndSt < NbrSt; EndSt++) {

        // Get number of trips poisson value for current weekday,
        // time interval and from station i to j
        double meanValue = WeekArray.get(WeekArrIndex).get(countLamb);

        // Exterior iteration variable
        countLamb = countLamb + 1;

        // Get random number of trips according to current poisson distribution
        int nTrips = roundToInt(poisson(meanValue));

        if (nTrips > 0) {
            for (int k = 0; k < nTrips; k++) {

                // Add person to agent Person
                add_people();
                countPp = people.size() - 1;
                people(countPp).setLocation(depot);

                // Add new order to yes/no service events
                StEndYesNot[StartSt][4] = StEndYesNot[StartSt][4] + 1;

                // Initiate new order
                people(countPp).StartSt = StartSt;
                people(countPp).EndSt = EndSt;
                depot.send("newOrder", people.get(countPp));

                // Exterior iteration variable
                countPp = countPp + 1;
            }
        }
    }
}
```

Clustering

Code implemented in agent Main inside function: *Clustering*.

```
try {
    // Define an empty model
    IloCplex model = new IloCplex();

    // Define the binary decision variables
    IloNumVar[][] z = new IloNumVar[NbrSt][NbrVehc];
    for (int i = 0; i < NbrSt; i++) {
        z[i] = model.boolVarArray(NbrVehc);
    }

    IloNumVar[] h = model.numVarArray(NbrVehc, 0,
    Double.MAX_VALUE);

    IloNumVar H = model.numVar(0, Double.MAX_VALUE);

    // Define the objective function
    model.addMinimize(H);

    // Define the constraints
    // (1)
    for (int i = 0; i < NbrSt; i++) {
        if (sSuff[i] != true) {
            IloLinearNumExpr e1 = model.linearNumExpr();
            for (int v = 0; v < NbrVehc; v++) {
                e1.addTerm(1, z[i][v]);
            }
            model.addEq(e1, 1);
        }
    }

    // (2)
    for (int i = 0; i < NbrSt; i++) {
        if (sSuff[i] == true) {
            IloLinearNumExpr e2 = model.linearNumExpr();
            for (int v = 0; v < NbrVehc; v++) {
                e2.addTerm(1, z[i][v]);
            }
            model.addLe(e2, 1);
        }
    }

    // (3)
    for (int v = 0; v < NbrVehc; v++) {
        IloLinearNumExpr e3 = model.linearNumExpr();
        for (int i = 0; i < NbrSt; i++) {
            if (s0[i] > 0) {
                e3.addTerm(s0[i], z[i][v]);
            }
            if (sMin[i] > 0) {
                e3.addTerm(-sMin[i], z[i][v]);
            }
        }
        if (q0[v] > 0) {
            model.addGe(e3, -q0[v]);
        }
        else {
            model.addGe(e3, 0);
        }
    }

    // (4)
    for (int v = 0; v < NbrVehc; v++) {
        IloLinearNumExpr e4 = model.linearNumExpr();
        for (int i = 0; i < NbrSt; i++) {
            if (s0[i] > 0) {
                e4.addTerm(s0[i], z[i][v]);
            }
            if (sMax[i] > 0) {
                e4.addTerm(-sMax[i], z[i][v]);
            }
        }
        if (abs(Q[v] - q0[v]) > 0) {
            model.addLe(e4, (Q[v] - q0[v]));
        }
        else {
            model.addLe(e4, 0);
        }
    }

    // (5)
    for (int v = 0; v < NbrVehc; v++) {
        for (int i = 0; i < NbrSt; i++) {
            IloLinearNumExpr e5 = model.linearNumExpr();
            double k = 0;
            for (int j = 0; j < NbrSt; j++) {
                if (d[i][j] > 0) {
                    k = k + d[i][j];
                    e5.addTerm(d[i][j], z[i][v]);
                    e5.addTerm(d[i][j], z[j][v]);
                }
                if (dp * sMax[j] > 0) {
                    e5.addTerm(dp * sMax[j], z[j][v]);
                }
                if (dm * sMin[j] > 0) {
                    e5.addTerm(dm * sMin[j], z[j][v]);
                }
                e5.addTerm(-1, h[v]);
            }
            if (k > 0) {
                model.addGe(k, e5);
            }
            else {
                model.addGe(0, e5);
            }
        }
    }

    // (6)
    for (int v = 0; v < NbrVehc; v++) {
        for (int i = 0; i < NbrSt; i++) {
            IloLinearNumExpr e6 = model.linearNumExpr();
            double k = 0;
            for (int j = 0; j < NbrSt; j++) {
                if (d[i][j] > 0) {
                    k = k + d[i][j];
                    e6.addTerm(d[i][j], z[i][v]);
                    e6.addTerm(d[i][j], z[j][v]);
                }
                if (dp * sMax[j] > 0) {
                    e6.addTerm(dp * sMax[j], z[j][v]);
                }
                if (dm * sMax[j] > 0) {
                    e6.addTerm(dm * sMax[j], z[j][v]);
                }
                e6.addTerm(-1, h[v]);
            }
            if ((k + dm * q0[v]) > 0) {
                model.addGe(k + dm * q0[v], e6);
            }
            else {
                model.addGe(0, e6);
            }
        }
    }

    // (7)
    for (int v = 0; v < NbrVehc; v++) {
        for (int i = 0; i < NbrSt; i++) {
            IloLinearNumExpr e7 = model.linearNumExpr();
            double k = 0;
            for (int j = 0; j < NbrSt; j++) {
                if (d[i][j] > 0) {
                    k = k + d[i][j];
                    e7.addTerm(d[i][j], z[i][v]);
                    e7.addTerm(d[i][j], z[j][v]);
                }
                if (dp * sMin[j] > 0) {
                    e7.addTerm(dp * sMin[j], z[j][v]);
                }
                if (dm * sMin[j] > 0) {
                    e7.addTerm(dm * sMin[j], z[j][v]);
                }
                e7.addTerm(-1, h[v]);
            }
            if ((k + dp * (Q[v] - q0[v])) > 0) {
                model.addGe(k + dp * (Q[v] - q0[v]), e7);
            }
            else {
                model.addGe(0, e7);
            }
        }
    }

    // (8)
    IloNumExpr e8 = model.max(h);
    model.addEq(H, e8);

    // Set Model Params
    //model.setParam(IloCplex.IntParam.Simplex.Display,
    0);
    model.setParam(IloCplex.Param.MIP.Tolerances.MIPGap,
    0.01);
    model.setParam(IloCplex.Param.TimeLimit, 60); //
seconds
}
```

```

double t1 = model.getCplexTime();
// Solve Model
boolean isSolved = model.solve();
double t2 = model.getCplexTime();
T = t2 - t1;

// Save optimal objective values, decision variables
and GAP
ArrayList results = new ArrayList();
double[][] zFinal = new double[NbrSt][NbrVehc];
double[][] hFinal = new double[NbrVehc][1];
double[][] HFinal = new double[1][1];
double[][] gapFinal = new double[1][1];

if (isSolved) {
    for (int v = 0; v < NbrVehc; v++) {
        for (int i = 0; i < NbrSt; i++) {
            zFinal[i][v] =
(double)(model.getValue(z[i][v]));
        }
        hFinal[v][0] = (double)(model.getValue(h[v]));
    }
    HFinal[0][0] = model.getValue(H);
    gapFinal[0][0] = model.getMIPRelativeGap();
} else {
    // Print error if model not solved
    PrintStream ps = System.out;
    ps.print("Model not Solved");
}

// Set output array
model.end();
model.close();
results.add(zFinal);
results.add(hFinal);
results.add(HFinal);
results.add(gapFinal);
return results;
} catch (IloException ex) {
    ex.printStackTrace();
}

return null;

```

Routing

Code implemented in agent Main inside function: *Routing*.

```
try {
    // Define an empty model
    IloCplex model = new IloCplex();

    // Define binary decision variables
    IloNumVar[][][] x = new IloNumVar[NbrStv][NbrStv
+ 1][[]];
    for (int i = 0; i < NbrStv; i++) {
        for (int j = 0; j < NbrStv + 1; j++) {
            x[i][j] = model.boolVarArray(NbrStv + 1);
        }
    }

    // Define integer decision variables
    IloNumVar[][] ym = new IloIntVar[NbrStv][[]];
    for (int i = 0; i < NbrStv; i++) {
        ym[i] = model.intVarArray(NbrStv + 1, 0,
Integer.MAX_VALUE);
    }

    // Define integer decision variables
    IloNumVar[][] yp = new IloIntVar[NbrStv][[]];
    for (int i = 0; i < NbrStv; i++) {
        yp[i] = model.intVarArray(NbrStv + 1, 0,
Integer.MAX_VALUE);
    }

    // Define decision variables
    IloNumVar h = model.numVar(0, Double.MAX_VALUE);

    // Define the objective function
    model.addMinimize(h);

    // Define the constraints
    // (1)
    for (int i = 0; i < NbrStv; i++) {
        IloLinearNumExpr e1 = model.linearNumExpr();
        for (int t = 0; t < NbrStv + 1; t++) {
            e1.addTerm(1, yp[i][t]);
            e1.addTerm(-1, ym[i][t]);
        }
        model.addGe(e1, sMinv[i] - s0v[i]);
    }

    // (2)
    for (int i = 0; i < NbrStv; i++) {
        IloLinearNumExpr e2 = model.linearNumExpr();
        for (int t = 0; t < NbrStv + 1; t++) {
            e2.addTerm(1, yp[i][t]);
            e2.addTerm(-1, ym[i][t]);
        }
        model.addLe(e2, sMaxv[i] - s0v[i]);
    }

    // (3)
    IloLinearNumExpr e3 = model.linearNumExpr();
    for (int i = 0; i < NbrStv; i++) {
        for (int j = 0; j < NbrStv + 1; j++) {
            e3.addTerm(1, x[i][j][0]);
        }
    }
    model.addLe(e3, 1);

    // (4)
    for (int i = 0; i < NbrStv; i++) {
        for (int t = 1; t < NbrStv + 1; t++) {
            IloLinearNumExpr e4 = model.linearNumExpr();
            for (int j = 0; j < NbrStv + 1; j++) {
                e4.addTerm(1, x[i][j][t]);
            }
            for (int j = 0; j < NbrStv; j++) {
                e4.addTerm(-1, x[j][i][t - 1]);
            }
            model.addLe(e4, 0);
        }
    }

    // (5)
    IloLinearNumExpr e5 = model.linearNumExpr();
    for (int i = 0; i < NbrStv; i++) {
        for (int t = 0; t < NbrStv + 1; t++) {
            e5.addTerm(1, x[i][i][t]);
        }
    }

    model.addEq(e5, 0);

    // (6)
    for (int i = 0; i < NbrStv; i++) {
        for (int t = 0; t < NbrStv + 1; t++) {
            IloLinearNumExpr e6 = model.linearNumExpr();
            e6.addTerm(1, ym[i][t]);
            for (int j = 0; j < NbrStv + 1; j++) {
                if (Qv > 0) {
                    e6.addTerm(-Qv, x[i][j][t]);
                }
            }
            model.addLe(e6, 0);
        }
    }

    // (7)
    for (int i = 0; i < NbrStv; i++) {
        for (int t = 0; t < NbrStv + 1; t++) {
            IloLinearNumExpr e7 = model.linearNumExpr();
            e7.addTerm(1, yp[i][t]);
            for (int j = 0; j < NbrStv; j++) {
                e7.addTerm(-Qv, x[j][i][t]);
            }
            model.addLe(e7, 0);
        }
    }

    // (8)
    for (int i = 0; i < NbrStv; i++) {
        IloLinearNumExpr e8 = model.linearNumExpr();
        for (int t = 0; t < NbrStv + 1; t++) {
            e8.addTerm(1, ym[i][t]);
        }
        model.addLe(e8, s0v[i]);
    }

    // (9)
    for (int i = 0; i < NbrStv; i++) {
        IloLinearNumExpr e9 = model.linearNumExpr();
        for (int t = 0; t < NbrStv + 1; t++) {
            e9.addTerm(1, yp[i][t]);
        }
        model.addLe(e9, Cv[i] - s0v[i]);
    }

    // (10)
    for (int t = 0; t < NbrStv + 1; t++) {
        IloLinearNumExpr e10 = model.linearNumExpr();
        for (int i = 0; i < NbrStv; i++) {
            for (int t2 = 0; t2 <= t; t2++) {
                e10.addTerm(1, ym[i][t2]);
                e10.addTerm(-1, yp[i][t2]);
            }
        }
        model.addGe(e10, -q0v);
    }

    // (11)
    for (int t = 0; t < NbrStv + 1; t++) {
        IloLinearNumExpr e11 = model.linearNumExpr();
        for (int i = 0; i < NbrStv; i++) {
            for (int t2 = 0; t2 <= t; t2++) {
                e11.addTerm(1, ym[i][t2]);
                e11.addTerm(-1, yp[i][t2]);
            }
        }
        model.addLe(e11, Qv - q0v);
    }

    // (12)
    IloLinearNumExpr e12 = model.linearNumExpr();
    for (int i = 0; i < NbrStv; i++) {
        for (int t = 0; t < NbrStv + 1; t++) {
            e12.addTerm(dm, ym[i][t]);
            e12.addTerm(dp, yp[i][t]);
            for (int j = 0; j < NbrStv; j++) {
                e12.addTerm(dv[i][j], x[i][j][t]);
            }
        }
    }
    model.addEq(h, e12);

    // Set Model Params
}
```

```

//model.setParam(IloCplex.IntParam.Simplex.Display
, 0);

model.setParam(IloCplex.Param.MIP.Tolerances.MIPGap, 0.01);
model.setParam(IloCplex.Param.TimeLimit, 60); //
seconds

double t1 = model.getCplexTime();
// Solve Model
boolean isSolved = model.solve();
double t2 = model.getCplexTime();
Tv[v] = t2 - t1;

// Save optimal objective values, decision
variables and GAP
ArrayList results = new ArrayList();
double[][][] xFinal = new double[NbrStv][NbrStv
+ 1][NbrStv + 1];
double[][][] ymFinal = new double[NbrStv][NbrStv
+ 1][1];
double[][][] ypFinal = new double[NbrStv][NbrStv
+ 1][1];
double[][][] hFinal = new double[1][1][1];
double[][][] gapFinal = new double[1][1][1];

if (isSolved) {
    for (int i = 0; i < NbrStv; i++) {
        for (int t = 0; t < NbrStv + 1; t++) {
            ymFinal[i][t][0] =
(double)model.getValue(ym[i][t]);
            ypFinal[i][t][0] =
(double)model.getValue(yp[i][t]);
            for (int j = 0; j < NbrStv + 1; j++) {
                xFinal[i][j][t] =
(double)model.getValue(x[i][j][t]);
            }
        }
        hFinal[0][0][0] = model.getValue(h);
        gapFinal[0][0][0] = model.getMIPRelativeGap();
    } else {
        // Print error if model not solved
        PrintStream ps = System.out;
        ps.print("Model not Solved");
    }
}

// Set output array
model.end();
model.close();
results.add(xFinal);
results.add(ymFinal);
results.add(ypFinal);
results.add(hFinal);
results.add(gapFinal);
return results;
} catch (IloException ex) {
    ex.printStackTrace();
}
return null;

```


Repositioning

Code implemented in agent Main inside function: *Repositioning*.

```
// Reset variables
InstructArray = new ArrayList < int[][] > ();
ymArray = new ArrayList < double[][][] > ();
ypArray = new ArrayList < double[][][] > ();
xvArray = new ArrayList < double[][][] > ();
H = 0;
GAP = 0;
s0 = new int[NbrStv];
sMin = new int[NbrStv];
sMax = new int[NbrStv];
sSuff = new boolean[NbrStv];
sp = new int[NbrStv];
sm = new int[NbrStv];
hv = new double[NbrVehc];
GAPv = new double[NbrVehc];

int WeekDay = getDayOfWeek();
WeekArrIndex = 0;
if (WeekDay == 1 || WeekDay == 7) {
    WeekArrIndex = 1;
}

// Update variables
for (int i = 0; i < NbrStv; i++) {
    s0[i] = roundToInt(stations(i).NbrBikes);
    sMin[i] = IntervMinArr[i][WeekArrIndex];
    sMax[i] = IntervMaxArr[i][WeekArrIndex];
    sp[i] = max(sMin[i] - s0[i], 0);
    sm[i] = max(s0[i] - sMax[i], 0);
    if (s0[i] >= sMin[i] & s0[i] <= sMax[i]) {
        sSuff[i] = true;
    } else {
        sSuff[i] = false;
    }
}

// Cluster stations
ArrayList < double[][] > ClusterResults =
    Clustering(C, DistMat, dm, dp, NbrStv, NbrVehc, q0,
    Q, s0, sMin, sMax, sSuff, sp, sm);

// Results from clustering
double[][] zF = ClusterResults.get(0);
H = ClusterResults.get(2)[0][0];
GAP = ClusterResults.get(3)[0][0];

// Run through all vehicles
for (int v = 0; v < NbrVehc; v++) {

    // Number of stations to visit
    int NbrStv = 0;
    for (int j = 0; j < NbrStv; j++) {
        NbrStv = NbrStv + roundToInt(zF[j][v]);
    }

    if (NbrStv > 0) {

        vOn[v] = true;

        // Initiate and rearrange variables
        int[] Sv = new int[NbrStv];
        int[] Cv = new int[NbrStv];
        double[][] dv = new double[NbrStv][NbrStv];
        int q0v = q0[v];
        int Qv = Q[v];
        int[] s0v = new int[NbrStv];
        int[] sMinv = new int[NbrStv];
        int[] sMaxv = new int[NbrStv];
        int n = 0;
        for (int j = 0; j < NbrStv; j++) {
            if (zF[j][v] >= 0.99) {
                Sv[n] = j;
                Cv[n] = C[j];
                s0v[n] = s0[j];
                sMinv[n] = sMin[j];
                sMaxv[n] = sMax[j];
                n = n + 1;
            }
        }
        for (int j = 0; j < Sv.length; j++) {
            for (int k = 0; k < Sv.length; k++) {
                dv[j][k] = DistMat[Sv[j]][Sv[k]];
            }
        }

        // Routing Instructions
        ArrayList < double[][][] > RoutingResults =
            Routing(Cv, dv, dm, dp, NbrStv, q0v, Qv, s0v,
            sMinv, sMaxv, v);

        // Results from routing
        double[][][] xFv = RoutingResults.get(0);
        double[][][] ym = RoutingResults.get(1);
        double[][][] yp = RoutingResults.get(2);
        hv[v] = RoutingResults.get(3)[0][0][0];
        GAPv[v] = RoutingResults.get(4)[0][0][0];

        // Instructions from routing results
        int[][] Instructions = new int[NbrStv + 1][4];
        for (int t = 0; t < NbrStv + 1; t++) {
            int StartSt = 0;
            int EndSt = 0;
            for (int i = 0; i < NbrStv; i++) {
                for (int j = 0; j < NbrStv + 1; j++) {
                    if (xFv[i][j][t] >= 0.99) {
                        Instructions[t][0] = Sv[i] + 1;
                        if (j == NbrStv) {
                            Instructions[t][2] = 0;
                        } else {
                            Instructions[t][2] = Sv[j] + 1;
                        }
                        StartSt = i;
                        EndSt = j;
                    }
                }
            }
            if (EndSt == NbrStv) {
                Instructions[t][1] = roundToInt(-
                ym[StartSt][t][0] +
                yp[StartSt][t][0]);
                Instructions[t][3] = 0;
            } else {
                if (ym[StartSt][t][0] > 0 ||
                yp[EndSt][t][0] > 0) {
                    Instructions[t][1] = roundToInt(-
                    ym[StartSt][t][0]);
                    Instructions[t][3] =
                    roundToInt(yp[EndSt][t][0]);
                } else if (ym[EndSt][t][0] > 0 ||
                yp[StartSt][t][0] > 0) {
                    Instructions[t][3] = roundToInt(-
                    ym[EndSt][t][0]);
                    Instructions[t][1] =
                    roundToInt(yp[StartSt][t][0]);
                } else {
                    Instructions[t][1] = 0;
                    Instructions[t][3] = 0;
                }
            }
        }

        // Update instructions array lists
        InstructArray.add(Instructions);
        ymArray.add(ym);
        ypArray.add(yp);
        xvArray.add(xFv);

        // Set trucks variables
        trucks(v).Instructions = Instructions;
        trucks(v).Stations = Sv;
    } else {
        // Update instructions array lists
        int[][] Instructions = new int[1][1];
        InstructArray.add(Instructions);
        double[][][] ym = new double[1][1][1];
        ymArray.add(ym);
        double[][][] yp = new double[1][1][1];
        ypArray.add(yp);
    }
}
```

UI Method 1

Code implemented in agent Person inside flowchart block: *state3*.

```
// Initiate variables
int ss = StartSt;
int es = EndSt;
double fullestPerc = 0;
double emptyPerc = 1;
int newSS;
int newES;
ExtraEffort = 0;
boolean b0 = randomTrue(main.CustCoop);
boolean b01 = false;
boolean b02 = false;
boolean b1 = false;
boolean b2 = false;
boolean StChange = false;
boolean EndChange = false;

UserType = 0;
if (b0 == true) {
    b01 = randomTrue(0.333);
    if (b01 == true) {
        b1 = true;
        b2 = true;
        UserType = 1;
    } else {
        b02 = randomTrue(0.5);
        if (b02 == true) {
            UserType = 5;
            b1 = true;
            b2 = false;
        } else {
            UserType = 7;
            b1 = false;
            b2 = true;
        }
    }
}

if (b1) {
    // New better departure station
    if (main.stations(ss).OccPerc < main.wpcPerc) {
        for (int k = 0; k < main.NbrSt; k++) {
            // Consider only neighbors
            if (main.NeighbMat[1][ss][k] <= main.Radius)
            {
                newSS =
                roundToInt(main.NeighbMat[0][ss][k]);
                if (main.stations(newSS).OccPerc >
                fullestPerc) {
                    StartSt = newSS;
                    fullestPerc =
                    main.stations(newSS).OccPerc;
                    if (StartSt != ss) {
                        StChange = true;
                        ExtraEffort = ExtraEffort +
                        main.DistMat[StartSt][ss];
                    }
                }
            }
        }
    }
}

if (b2) {
    // New better arrival station
    if (main.stations(es).OccPerc > 1 -
    main.wpcPerc) {
        for (int k = 0; k < main.NbrSt; k++) {
            // Consider only neighbors
            if (main.NeighbMat[1][es][k] <= main.Radius)
            {
                newES =
                roundToInt(main.NeighbMat[0][es][k]);
                if (main.stations(newES).OccPerc <
                emptyPerc) {
                    EndSt = newES;
                    emptyPerc =
                    main.stations(newES).OccPerc;
                    if (EndSt != es) {
                        EndChange = true;
                        ExtraEffort = ExtraEffort +
                        main.DistMat[EndSt][es];
                    }
                }
            }
        }
    }
}

}
}

// Define User Type
if (b1 & b2 & StChange == true && EndChange ==
true) {
    UserType = 2;
} else if (b1 & b2 & StChange == true && EndChange
== false) {
    UserType = 3;
} else if (b1 & b2 & StChange == false &&
EndChange == true) {
    UserType = 4;
} else if (b1 & !b2 & StChange == true &&
EndChange == false) {
    UserType = 6;
} else if (!b1 & b2 & StChange == false &&
EndChange == true) {
    UserType = 8;
}
}
```

UI Method 2

Code implemented in agent Person inside flowchart block: *state9*.

```
// Initiate variables
int ss = StartSt;
int es = EndSt;
double fullestPerc = 0;
double emptyPerc = 1;
int newSS;
int newES;
ExtraEffort = 0;
boolean b0 = randomTrue(main.CustCoop);
boolean b01 = false;
boolean b02 = false;
boolean b1 = false;
boolean b2 = false;
boolean StChange = false;
boolean EndChange = false;

UserType = 0;
if (b0 == true) {
    b01 = randomTrue(0.333);
    if (b01 == true) {
        b1 = true;
        b2 = true;
        UserType = 1;
    } else {
        b02 = randomTrue(0.5);
        if (b02 == true) {
            UserType = 5;
            b1 = true;
            b2 = false;
        } else {
            UserType = 7;
            b1 = false;
            b2 = true;
        }
    }
}

if (b1) {
    // New better departure station
    if (main.stations(ss).OccPerc < main.wpcPerc) {
        int k = 0;
        newSS = ss;
        while (main.NeighbMat[1][ss][k] <= main.Radius
        &
            main.stations(newSS).OccPerc < main.wpcPerc)
        {
            newSS =
            roundToInt(main.NeighbMat[0][ss][k]);
            k = k + 1;
            if (main.stations(newSS).OccPerc >=
            main.wpcPerc) {
                StartSt = newSS;
                if (StartSt != ss) {
                    StChange = true;
                    ExtraEffort = ExtraEffort +
                    main.DistMat[StartSt][ss];
                }
            }
        }
    }

    if (b2) {
        // New better arrival station
        if (main.stations(es).OccPerc > 1 -
        main.wpcPerc) {
            int k = 0;
            newES = es;
            while (main.NeighbMat[1][es][k] <= main.Radius
            &
                main.stations(newES).OccPerc > 1 -
                main.wpcPerc) {
                newES =
                roundToInt(main.NeighbMat[0][es][k]);
                k = k + 1;
                if (main.stations(newES).OccPerc <= 1 -
                main.wpcPerc) {
                    EndSt = newES;
                    if (EndSt != es) {
                        EndChange = true;
                        ExtraEffort = ExtraEffort +
                        main.DistMat[EndSt][es];
                    }
                }
            }
        }
    }
}

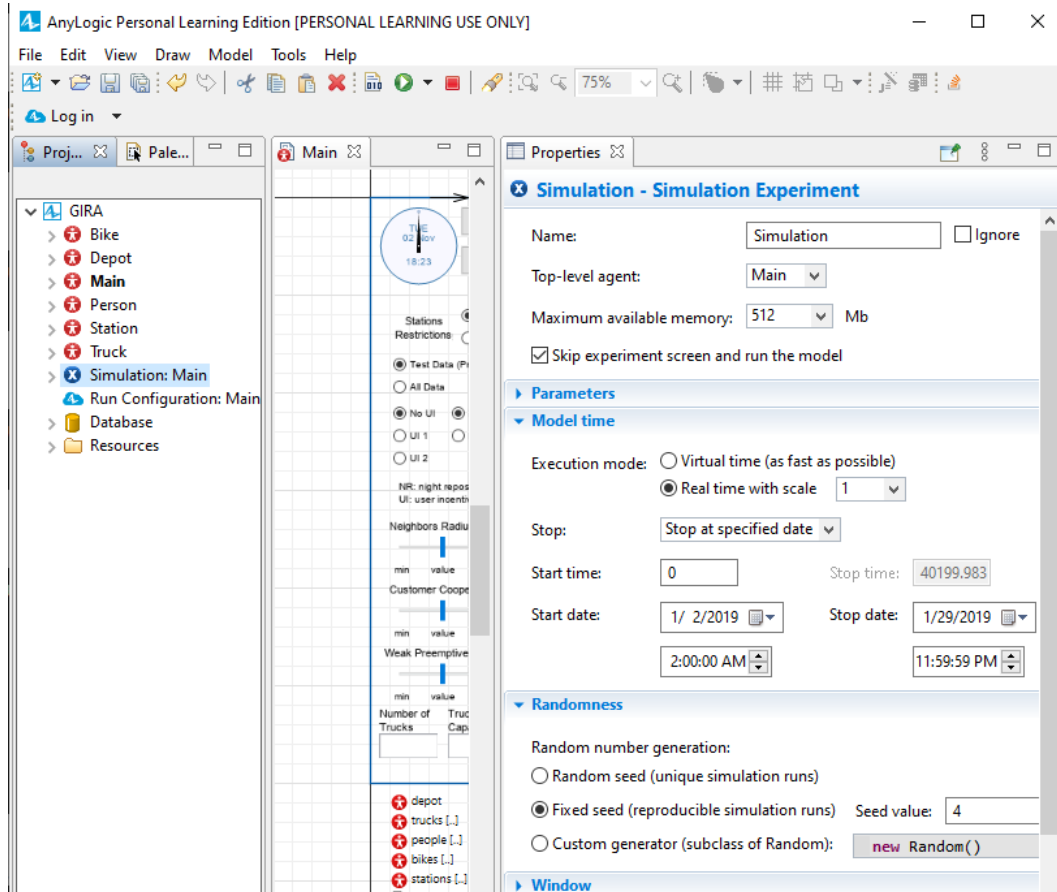
}

// Define User Type
if (b1 & b2 & StChange == true && EndChange ==
true) {
    UserType = 2;
} else if (b1 & b2 & StChange == true && EndChange
== false) {
    UserType = 3;
} else if (b1 & b2 & StChange == false &&
EndChange == true) {
    UserType = 4;
} else if (b1 & !b2 & StChange == true &&
EndChange == false) {
    UserType = 6;
} else if (!b1 & b2 & StChange == false &&
EndChange == true) {
    UserType = 8;
}
```

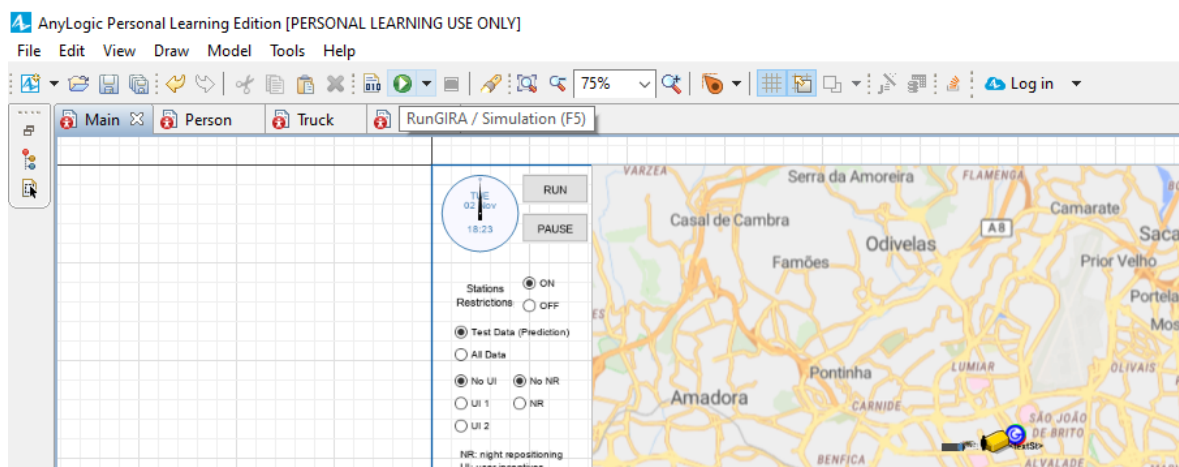
A.5 Simulator user manual

To properly use this simulator, the sequence of procedures to make an experiment is presented below.

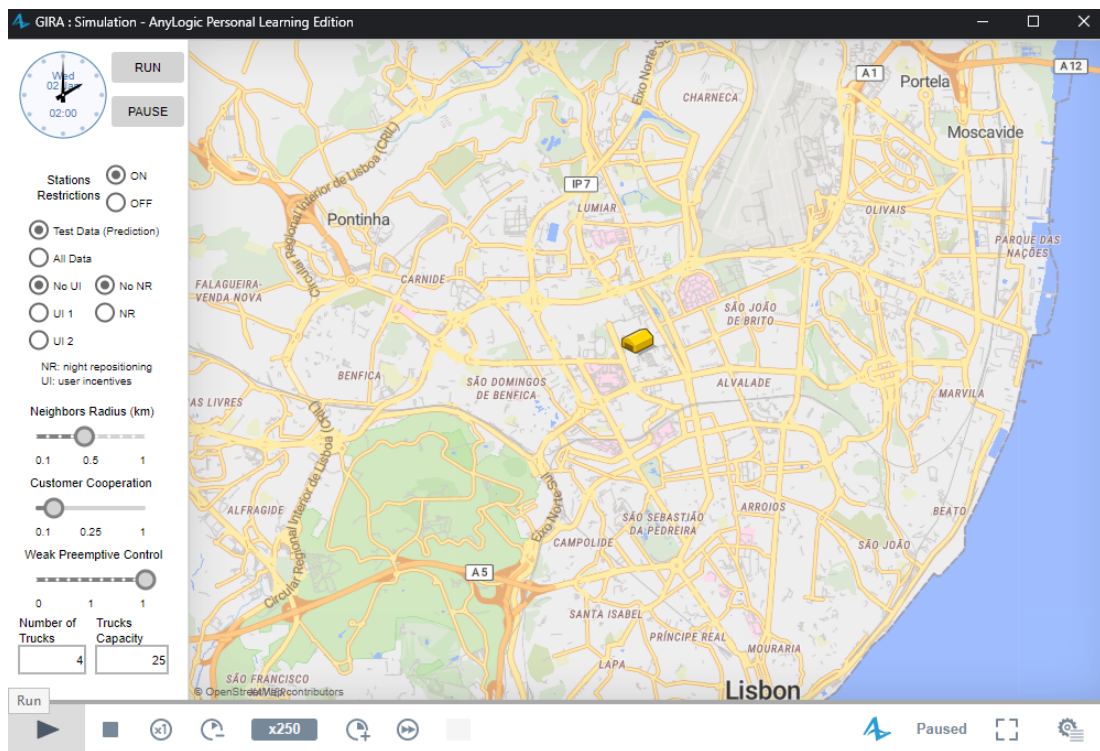
(1) Define model time on "Simulation" tab;



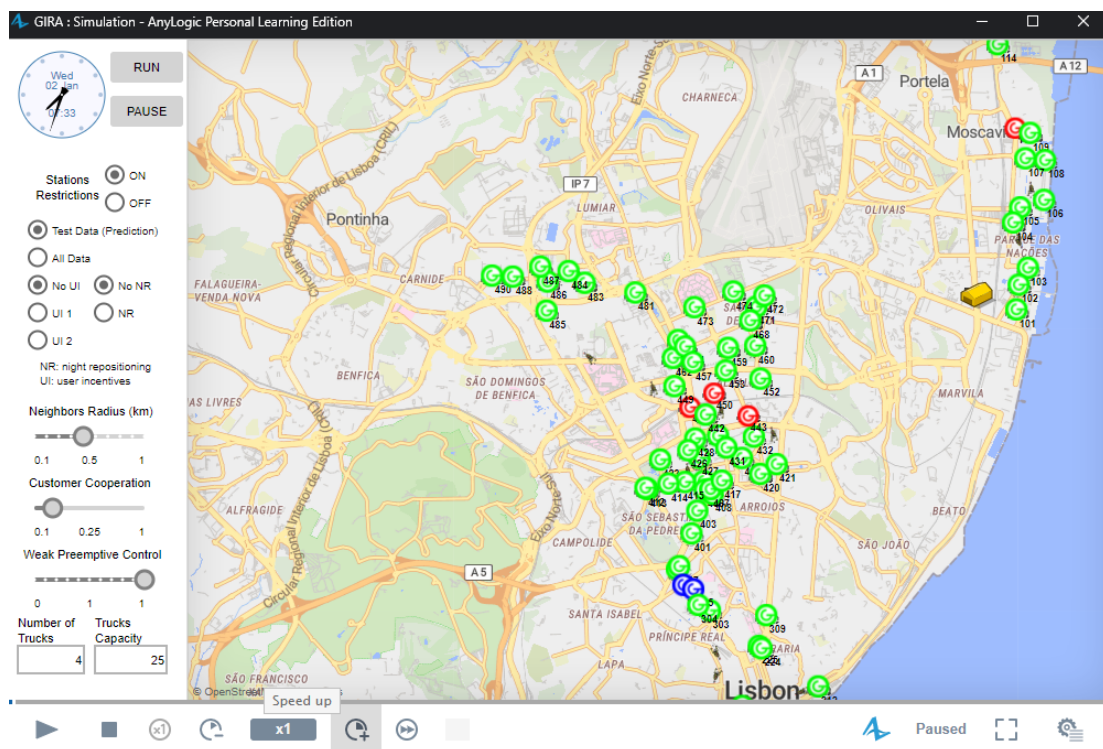
(2) Hit green "RunGIRA / Simulation (F5)" button;



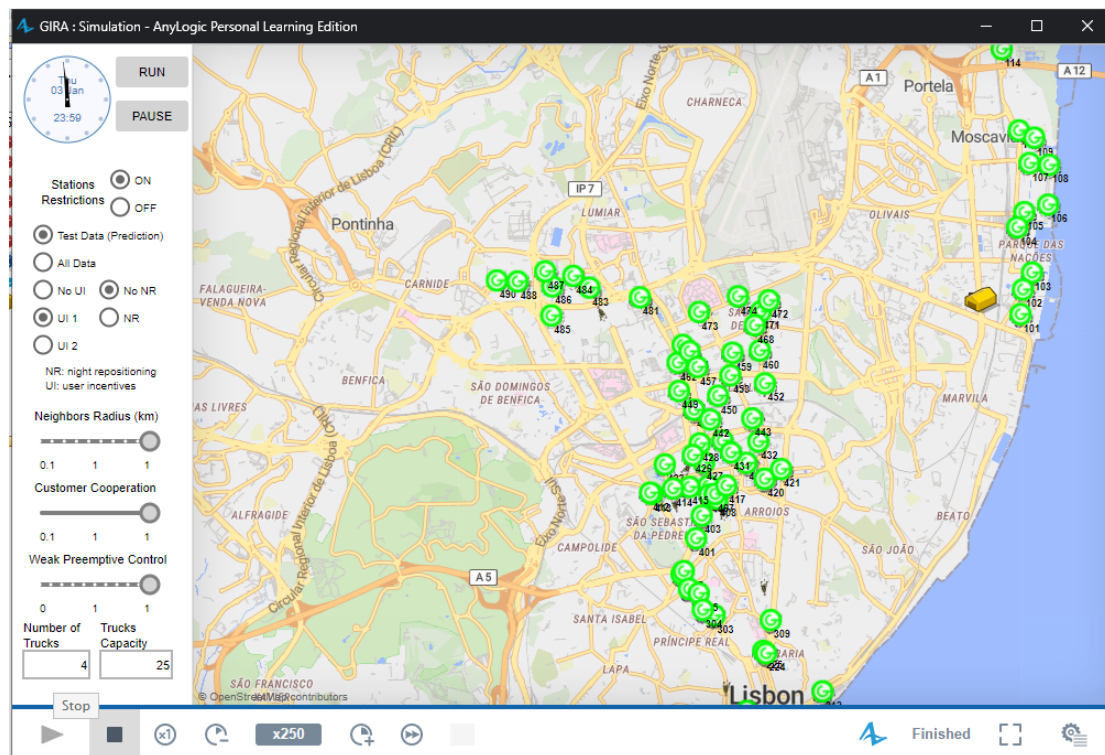
(3) Set input variables and hit "Run" button;



(4) Simulation speed can be increased or decreased. It will finish at "Stop model date/time".



(5) Hit the "Stop" Button;



(6) Check trip results on tables: *trip_table*, *lost_user* and *ok_table*; and repositioning results on the excel file: *RepResults.xlsx*.

AnyLogic Personal Learning Edition [PERSONAL LEARNING USE ONLY]

File Edit View Model Tools Help

Log in

Properties

lost_users - Data

trip_table

	startlocation	endlocation	startdate	enddate	triptime	extraeffort	usertype
1	39	56	02-01-20				
2	39	40	02-01-20				
3	63	63	02-01-20				
4	45	39	02-01-20				
5	72	61	02-01-20				
6	53	12	02-01-20				

lost_users

	date	st1	st2
1	02-01-2019 06:08:11	56	57
2	02-01-2019 07:22:32	51	52
3	02-01-2019 07:44:06	51	52
4	02-01-2019 07:56:07	52	51
5	02-01-2019 08:59:33	10	23
6	02-01-2019 09:27:09	18	39

ok_table

	station	orders	startok	start
1	1	92	57	
2	2	137	90	
3	3	54	47	
4	4	58	78	
5	5	52	64	
6	6	19	58	
7	7	46	62	
8	8	14	32	

RepResults.xlsx

	A	B	C	D	E	F	G	H	I	J	K
154	214	72	4	19	19	19					
155	208	73	4	21	16	16					
156	303	74	0	9	10	9					
157											
158	849.26	0.01	2.75								
159	8533.4	0.002	0.329	33082	83.578						
160	8585	0.0013	0.533	25555	72.932						
161	12875	0.0012	0.266	31691	99.576						
162	16430	0.0004	0.078	18113	84.423						
163											
164	104	1	6	36	16	16					
165	105	2	0	29	21	22					
166	101	3	5	14	8	8					
167	102	4	5	13	6	6					
168	103	5	0	11	4	4					
169	106	6	0	11	8	8					
170	107	7	5	13	13	13					
171	108	8	1	14	13	13					
172	109	9	9	14	10	10					
173	110	10	10	18	12	12					
174	452	11	7	10	4	7					
175	457	12	6	10	1	6					
176	481	13	9	15	11	11					
177	464	14	7	14	4	7					
178	414	15	4	17	0	4					
179	468	16	9	21	8	9					
180	431	17	10	11	5	10					
181	412	18	4	13	1	4					
182	442	19	7	16	2	7					
183	430	20	3	17	8	8					
184	426	21	1	7	0	1					