UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

Tráfico de bicicletas a través de un servicio de enrutamiento basado en OpenStreetMap.

Bicycle traffic using OpenStreetMap routing service

Alberto López Santiago
2019

TRABAJO DE FIN DE GRADO

Título:	Tráfico de bicicletas a través de un servicio de enrutamiento basado en OpenStreetMap
Título (inglés):	Bicycle traffic using OpenStreetMap routing service
Autor:	Alberto López Santiago
Tutor:	Carlos Ángel Iglesias Fernández
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE GRADO

Bicycle traffic using OpenStreetMap routing

Alberto López Santiago

Junio 2019

Resumen

Los sistemas de alquiler de bicicletas, que permiten su uso por parte de una persona a cualquier momento del día es un complejo sistema de comportamiento. Este comportamiento puede verse influenciado por diferentes factores como la hora del día o el origen.
El estudio de diferentes políticas para aumentar el balanceo del servicio permite analizar diferentes situaciones y sus consecuencias a la carga del sistema y por lo tanto a la disponibilidad del servicio.
En este trabajo se propone una comparativa, a través de un modelo basado en agentes, de diferentes políticas para balancear la ocupación de las estaciones del servicio de alquiler de bicis, BiciMAD.
Las rutas para los usuarios se obtienen en función de la probabilidad de los viajes reales. Las estaciones actuarán como agentes pasivos y reactivos. Además será analizada la distribución de las estaciones a lo largo de la geografía de Madrid, obteniendo zonas con necesidad de políticas para regular su carga.
Como resultado se obtiene un 32.2% de mejora en el número de estaciones con baja ocupacion y un 44.7% en estaciones cargadas. También se consigue un aumento en el número de viajes realizados de un 23.25%.
Se observa mucha saturación en la zona colindante al Paseo de la Castellana y una falta de bicis en estaciones más alejadas.

Palabras clave: Sistemas multi-agente, Simulación de ocupación, Políticas de rebalanceo, Disponibilidad de servicio

Abstract

Bicycle rental systems, which allows a person to use a bike at any time of day is a complex behavioral system. This behavior can be influenced by different factors like the time of day or the origin.

The study of different policies to increase the availability of the service allows analyzing different situations and their consequences for the load of the system.

This paper proposes a comparison, through an agent-based model, of different policies to balance the occupancy of BiciMAD bike rental service stations and therefore to the availability of the service.

Routes for users are obtained based on the probability of the real trips. The stations will act as passive and reactive agents. In addition, the distribution of the stations throughout the geography of Madrid will be analyzed, obtaining areas in need of policies to regulate their load.

The result is a 32.2% improvement in the number of stations with low occupancy and 44.7% in heavily loaded stations. An increase in the number of trips made of 23.25% is also achieved. The simulations show saturation in the stations of the area adjoining Paseo de la Castellana and a lack of bikes in more distant stations.

Keywords: Multi-agent systems, occupation simulation, rebalancing policies, service availability

Agradecimientos

A mis padres, por estar ahí aunque hubiese problemas en el camino.
A mi hermana Irene, como inspiración para sacar una mejor parte de mi.
A mis abuelos, por confiar en mi ciegamente como solo hace un abuelo.
A mi abuela Chicha, que aunque no estés aqui para verlo te he tenido conmigo todo este tiempo.
A Marta, por aguantar mis neuras y obsesiones con el trabajo sin volverse loca y hacerme más felices los días.
A mis amigos, por hacer este viaje entre risas y cervezas uno inolvidable.
A mi tutor Carlos Angel por exprimirme al máximo y proponerme retos.

Muchas gracias a todos.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

In a globalized society where the need to move through the city is daily and where the main solution is motor vehicles that, as indicated in [10] assumed in 2004 75.5%, 91% and 34% of the total emissions of NOx, CO and NMVOC respectively in Madrid. This, added to European climate control policies, is shown as a crucial point of action.
An emerging alternative to this issue would consist in the implementation of bike sharing systems (BSS). Bike share technology has evolved over the course of decades exponentially. As of 2014, this option has already been implemented in 700 cities with more than 800,000 bicycles riding [16].
Their benefits for cities are multiple: from a greener image due to more eco-friendly means of transportation to the reduction of traffic congestion, noise and air pollution, they provide an alternative to private motorized vehicles, especially for short-distance trips. According to Paul DeMaio [4], these systems can be categorized into 4 generations:

- **First generation**
  Established in Amsterdam in 1965, bicycles were available on the street and could be used for free by anyone. This system has the lowest psychological barrier for a potential user but suffered loss rates from theft and vandalism.
- **Second generation**

Designed in Denmark in 1991, bicycles were installed in stations throughout the city. Making a previous deposit of money, the users could ride from one station to the other. However, due to the anonymity of the users, bicycle thefts continued to occur.

- **Third generation**
  In contrast to the previous generation, anonymity is no longer a problem thanks to the implementation of authentication through a single user card. This system is widely used in cities, such as Madrid (BiciMAD), Barcelona (SalenBici) or Amsterdam (OV-Fiets).

- **Fourth generation**
  In this case, the mobile phone plays a key role: stations are no longer necessary, as the lock is integrated onto the frame of the bicycle. This system is usually implemented by private companies such as Mobike which operates in more than 20 European cities such as Hanover, Amsterdam or Zaragoza.

Figure 1.1: Third generation BiciMAD station, Madrid.

1.2 Project goals

The objective of this project is to create a web application where Agent-based Social Simulations could be visualized. Also, the analysis of different policies to prevent inequality in occupation between different areas, applied to the bike shared system BiciMAD.
In addition, the distribution of the occupation of the stations in the geography of Madrid will be analyzed in order to analyze patterns of use.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document.

1. Introduction. The background and the project goals are presented.
2. Enabling technologies. Previous projects are analyzed and the tools that are going to be used are explained
3. Dataset processing. The realization of the dataset is explained, formed from the actual data of use.
4. Model description. A description is made about the components of the system and about the different models developed.
5. Architecture. The connection between the different modules is explained.
6. Case study. The environment used for the study will be explained and the results of the simulation will be presented.
7. Conclusions. The consequences of the results are valued.

CHAPTER 2

Enabling Technologies

This chapter shows us the main technologies used or involved in the use of our related theme, introducing firstly the social environment and the issues we are going to try to solve in this project.

2.1 State of the art

One of the main obstacles that third generation faces is the forecast of service. The mismatch of occupation between stations need to be avoided. It is not uncommon to see situations where there is station without any free docks to park the bicycle (check-out), while there is another one with no bicycles for rent (check-in); the user would have to move or wait to be able to travel. Having a predictive model to balance the occupation of the stations can benefit both the user and the BSS provider.
As used in [13] several review questions were formulated before locating and selecting relevant studies. These questions are the following:

- **Q1. Does the work deal with bicycle traffic?**
- **Q2. Does it study a bicycle renting service case?**
- **Q3. Does it involve a Spanish city?**
- **Q4. Is real data employed in the study?**

Table 2.1: State of the art papers. Check mark: yes, empty space: No

References							
Ref.	Target				Method		
	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Yu Zheng et. al [7]	✓	✓		✓			✓
Junming Liu et al. [8]	✓	✓		✓			✓
Longbiao Chen et al. [3]	✓	✓		✓			✓
Divya Singhvi et al. [14]	✓	✓		✓			
Rodrigo Meza et. al [5]	✓	✓	✓	✓	✓		✓
Alvaro Lozano et. al [9]	✓	✓	✓	✓	✓		
Marco Santoni et. al [15]	✓	✓		✓			
Mieke Massink et. al [11]	✓	✓		✓			✓
Martin Loidl et al. [17]	✓				✓	✓	
Simone Z. Leao [6]	✓			✓	✓	✓	✓

- **Q5. Does it include a visualization of the traffic simulated?**
- **Q6. Is there an agent-based social simulation?**
- **Q7. Does it contrast different policies results?**

These questions can be broken down into two types: type of target (Q1–Q4) and method used (Q5–Q7). The summary about the chosen papers can be seen on the Table 2.1.
The traffic of the system and therefore the occupation of the stations can be influenced by a wide variety of factors. For instance, Yu Zheng et al.[7] uses the collected data of the stations as well as the time, the wind speed and the temperature as variables of his demand prediction system. However Junming Liu et al.[8] runs the analysis in terms of weather, wind speed, temperature, humidity and visibility.
Several studies have developed occupancy simulation systems for these stations in different cities such as New York or Washington DC [3, 14] where a database of events, traffic data and environmental factors trained a model to forecast the state of demand of the stations

clusters based on their occupation characteristics.

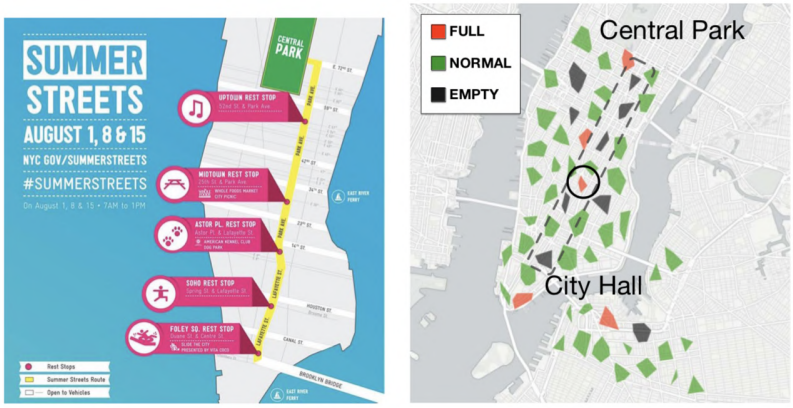
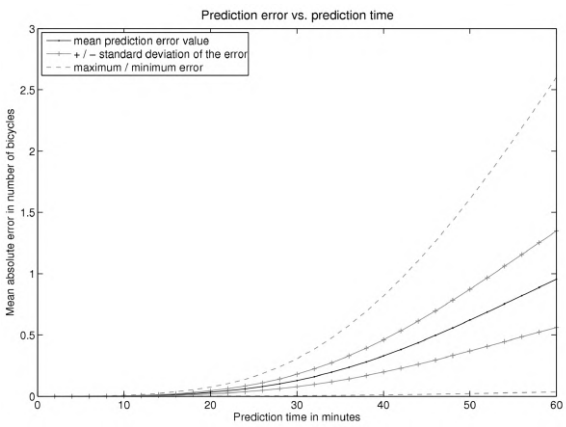
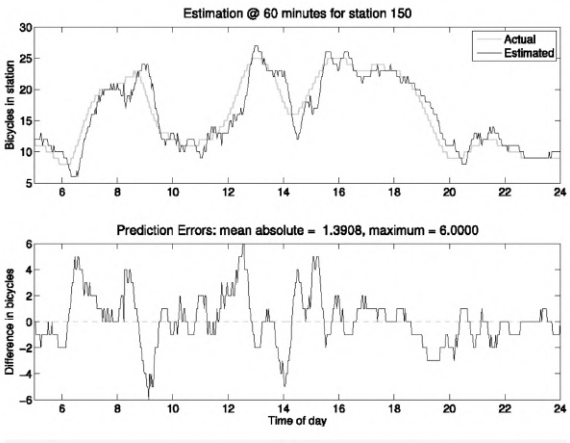


Figure 2.1: Simulation of occupancy when an event occurs, New York. [3]

Predictive models have also been developed in Spanish cities such as Barcelona [5], where different simulations were performed taking into account a set of variables. As can be seen in Figure 2.2(a), the mean error obtained in the prediction is less than 1 bicycle per station in a time windows of 60 minutes. However, it was also noted that the error depends on the hour of the day in which the analysis is made, obtaining peaks of up to 6 bikes of difference in the central hours of the day where a more abnormal behaviour is observed.



(a) Mean prediction error depending on time



(b) Estimation error @ 60 minutes

Figure 2.2: Estimation error from Barcelona´s simulation. [5]

Another example of a system used to overcome this casuistry is the one developed by Álvaro Lozano et. al. [9], where they choose the Random Forest regression algorithm to train the model with the data of the trips made in the Salamanca BSS between 2013-2017.

These data obtained from the analysis is presented in a graphical interface in a web environment, so that the user can see what the state of a station will be like depicted at a certain moment in time. When a prediction of demand in a certain station is made, different actions can be implemented to achieve a better load balance between their stations, such as the use of economic incentives to the user if they deposit the bicycle in a station with a lower occupation; or trucks that redistribute bicycles from stations with an excess load to others that are emptier, thus participating actively in the ecosystem.

The agent-based models are useful for the spatial distribution of agents and their interaction with the environment, as seen by Simone Z Leao et al. [6] in their study on traffic flows of people going to work. The different routes that agents carry out can be viewed in a map, as shown in Fig. 2.3. These paths can be compared the ones done by 16 real cyclists. This can be very useful to reroute traffic at certain times of the day or react to an unexpected event such as an accident.

This utility has also been studied by Martin Loidl et al. through a simulation to observe

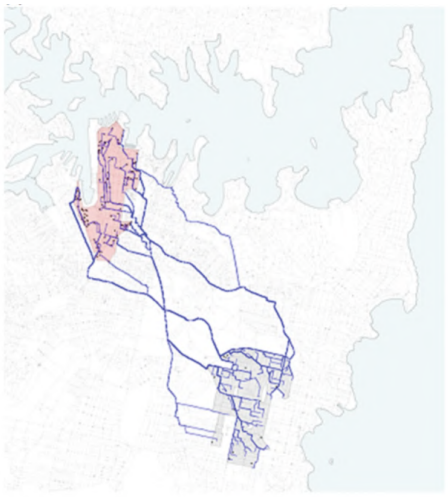


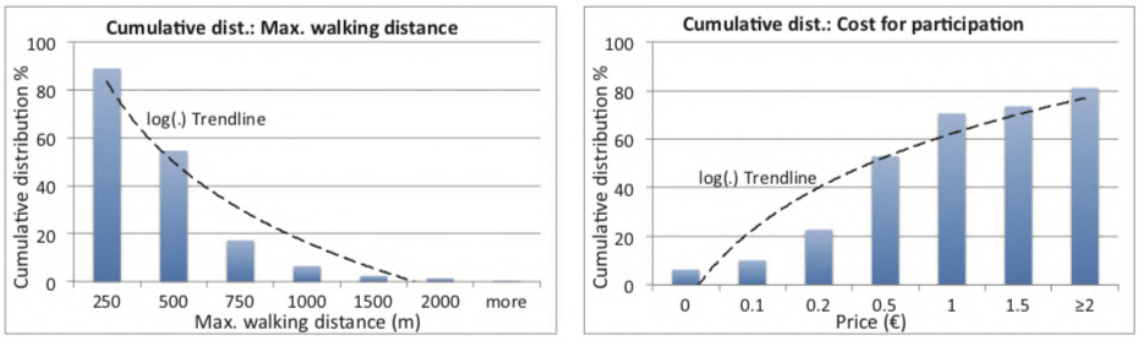
Figure 2.3: Simulation of routes between Randwick and Sydney centre [6] .

traffic in the city of Salzburg [17]. Similarly to the previous case the purpose was to study the routes used by the agents and the impact that outlying areas have on traffic. The simulation, executed in NetLogo, generates agents of different types; workers, students and leisure cyclists. Then a destination is generated to specific areas of the city depending on the agent type, such as enterprise zones or university campuses, as shown in Fig. 2.4.

According to Adish Singla et. al. [15] through a study on a BSS in a European city about the involvement of users in the techniques of balancing the stations through economic

Figure 2.4: Simulation of "Salzburg City" (left) and "Salzburg City Region" (right) [17].

rewards. In the figure 2.5 (a) the distribution of participation can be seen according to the walking distance. In (b) the probability distribution can be seen, it depends on the economic retribution of the incentive.



(a) Dist. of max. walking distance

(b) Dist. of costs

Figure 2.5: Study on participation in incentive policies [15].

2.2 Technologies used

The objective of this study is to carry out a simulation based on bike traffic agents of BiciMAD by the city of Madrid and contrast it with real usage data, that is why the technologies to be used can be divided into two groups, those related to data analysis and graphic representation and on the other hand those that have to do with the simulation and obtaining of the routes. For the simulation we will use an agent-based model (ABM), which is a class of computational models for simulating the actions and interactions of autonomous agents. The basis of this system is that the agent has initial conditions that change autonomously in each step of the model.

One of the most famous examples of ABM is the simulation of wealth distribution. In this simulation, an N number of agents is created with an equal amount of initial wealth for all of them and distributed on a map. Each agent moves along the map and gives 1 wealth to another nearby random agent. In this way, after a number M of steps, wealth is concentrated in a small percentage of the agents, while the great majority have lost their wealth. In the example, the following aspects are identifiable, which are characteristics of an ABM:

- Number of agents.
- Decision-making heuristics.
- Adaptive processes.
- Interaction topology.
- An environment.

Therefore, these aspects must be defined at the moment of creating the system so that it can work correctly.

2.2.1 Data analysis and visualization libraries

The following libraries are responsible for two functions: the analysis and processing of the data and the creation of all the visual elements shown in the project. These are the graphs that draw the results and the interactive map.

- **Seaborn**
  Seaborn[1] is a Python data visualization library based on matplotlib which expands its potential use with different new representations such as **lineplot**, **boxenplot** or **jointplot**. It's the main library used to represent and visualize the data.

- **Numpy**
  Numpy[2] is a Python library for scientific computing. It's used to calculate the correlation between the simulated data and the real one.

- **Pandas**
  Pandas[3] is the Python library used to analyze and process the real usage data in order

[1] https://seaborn.pydata.org
[2] http://www.numpy.org/
[3] https://pandas.pydata.org/

to get the probability matrix or mean usage of the service.
The data generated by the ABM is also stored in a DataFrame to its subsequent analysis.

- **Folium**
  Folium[4] is the library used to draw the data processed by the ABM on a Leaflet map. Folium is avaliable on Python and Javascript (JS) and allows different types of maps and layers,such as the openstreetmap layer or a heatmap.
  Folium implements different functionalities such as markers, which can be customized with a descriptive text that is displayed once they are clicked, allowing their differentiation.

2.2.2 Agent-based simulator and routing

In this block you will find the libraries whose function is included in the simulation processing. These functions are to obtain the routes of the agents, based on real geographic information; and the execution of the functions associated with the behavior of each agent.

- **OpenStreetMap**
  OpenStreetMap (OSM)[5] is a wiki that allows users to keep an updated map through their contributions, these are checked by collaborators to check their reliability. Its access is free and allows the export of maps in different formats, in this case OSM will be used.

- **OpenRouteService**
  OpenRouteService (ORS)[6] is a Python library used to get the routes between the origin and the destination of bike agents. The route is presented as an array of positions (latitude,longitude) that contains the points to make the route.
  To deploy ORS service a docker container is mounted, which can be obtained from its repository at github [7]. This container generates the route files for different profiles from an OSM map file:
  - **cycling-regular**
  - **vehicles-car**
  - **vehicles-hgv**

[4]https://github.com/python-visualization/folium
[5]https://www.openstreetmap.org/
[6]https://openrouteservice.org
[7]https://github.com/GIScience/openrouteservice

Each of these profiles has associated different paths that can travel and different speeds. These speeds are also variable depending on the type of road to travel, as can be seen in the figure C.3.

This library offers many customization options for the routes, allowing to avoid certain points that for example have a large traffic, or even get the routes to the nearest entertainment bars. An example of routes where areas with high traffic are avoided can be seen in their example on the web[8], in this example the flooded zone is get through data obtained from Twitter, figure 2.6

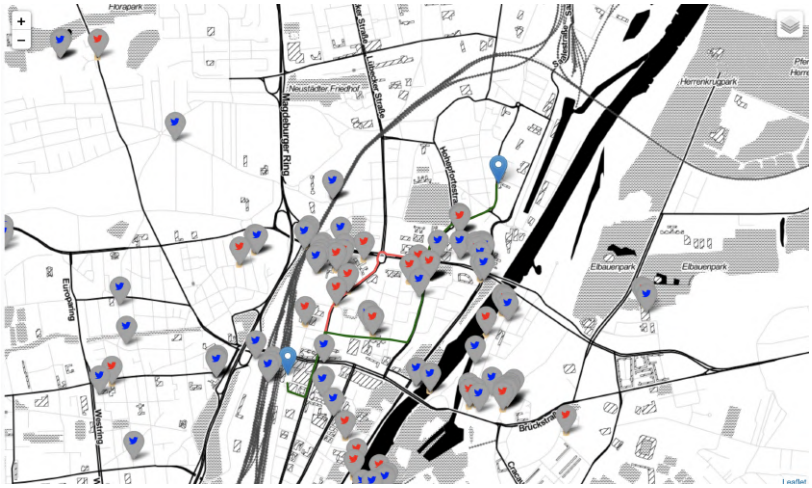


Figure 2.6: Regular route (red) and alternative route (green) avoiding flood.

- **Mesa**

  Mesa[9] is an agent-based modeling framework in Python which allows the execution of an ABS using built-in core components. It's an open source project that pretends to replace NetLogo. Mesa launches a Tornado server with which it communicates through JSON messages and allows visualization in a web environment. This can be used to implement different libraries on the server, such as folium. In Figure C.2 the execution of the wealth distribution model can be observed in the web server. In this model a grid map is used where the agents are placed and assigned a visualization pattern.

[8] https://openrouteservice.org/example-avoid-flooded-areas-with-ors/
[9] https://github.com/projectmesa

CHAPTER 3

Dataset processing

3.1 Introduction

For the realization of the project we need to obtain real data of use of BiciMAD, that is why a processing is necessary. The data has been obtained from the official website of the Municipal Transport Company of Madrid (EMT)[1]. There you can obtain data on the situation of the stations and on the use, that is, the routes carried out, all these data are recorded every hour. In order to carry out this study, the usage data were obtained between August 2018 and November 2018, that is, 4 months of using the BSS service.

3.2 Processing

The nature of the data is different so, these should be treated independently. For its processing and subsequent use the csv format has been chosen, to facilitate the reading and analysis of the data the Pandas library will be used.

3.2.1 Routes data

The data about the routes is provided in json format according to the structure of the figure 3.1, where we can see a record of a trip. The first step is to select which fields we want to

[1] http://opendata.emtmadrid.es/Datos-estaticos/Datos-generales-(1)

keep from the data.

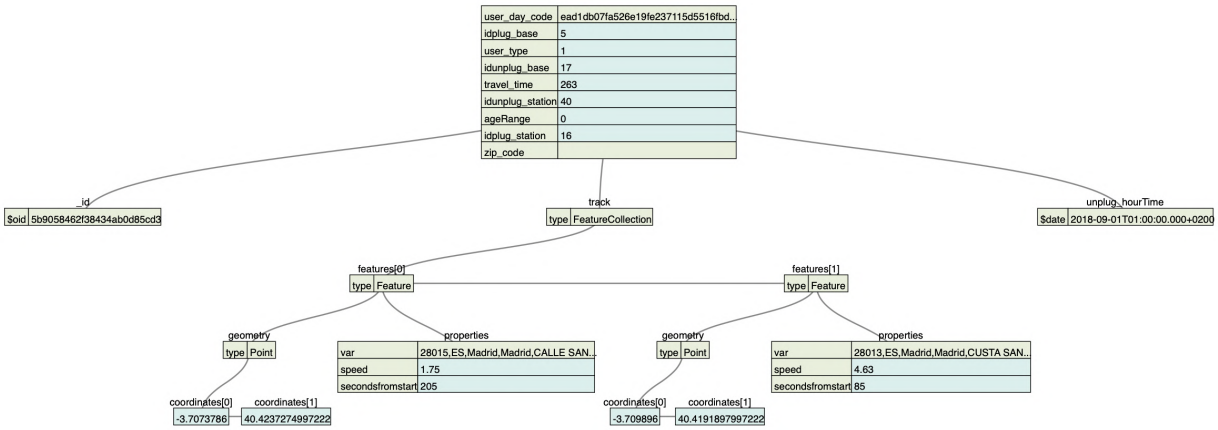


Figure 3.1: Route json structure.

- The field **_id** is a unique identifier of the trip, so it does not provide information.

- In the variable **track** we can obtain a record measured every minute of the position of the bicycle, however, we can not assure which route it has taken until reaching the new position, so we discard this field.

- In **unplug_hourTime** we can obtain information about the day in which the rent of the bike was made. However, due to privacy reasons the EMT provides only the hour, without giving access to the minutes in which it was made. On top of that the time is in a timezone **GMT +2**.

- The field **user_day_code** contains a hexadecimal code that identifies the user during a time cycle of 24 hours, which may allow us to analyze the assiduity of use of the service by the same user but always within the margin of one day. Since the window is very small and does not produce clear data, we leave aside this field.

- The other fields provide information about the station and the base, both origin and destination and the duration of the trip, as well as the duration, which is in

travel_time in seconds. It is important to preserve the field **user_type**, since it will allow us to later eliminate the routes associated with the workers of the EMT and thus avoid contaminated data.

Therefore the final json would be as you can see in the figure 3.2. This will be the one that will later be analyzed and stored in a dataframe.

In this study, we will focus on working days, Monday through Friday. To obtain the day of

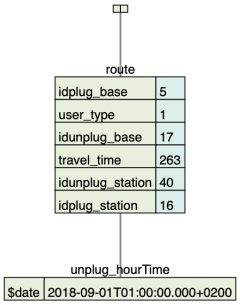


Figure 3.2: Chosen fields to analyze the data.

the week based on the date we use the calendar library that, through the weekday method, returns an integer between 0 and 6, being 0 on Monday and so on progressively. The date and time have been separated, so that we have 3 fields, **Day**, **Month** and **Hour**, obtaining a dataframe like the figure 3.3.

	ageRange	idplug_base	idplug_station	idunplug_base	idunplug_station	travel_time	user_type	Day	Hour	Month
0	0	11	43	7	1	228	1	1	1	8
1	0	11	43	7	1	228	1	1	1	8
2	0	22	11	11	163	378	1	1	1	8
3	0	14	80	2	46	380	1	1	1	8
4	4	17	135	20	52	452	1	1	1	8

Figure 3.3: Route dataframe sample of August 2018 routes.

3.2.1.1 ageRange

According to EMT data the field **ageRange** includes information about the age ranges, as shown in the table 3.1

Table 3.1: Description of the ageRange values.

ageRange value	Age (years old)
0	Indeterminate
1	0-16
2	17-18
3	19-26
4	27-40
5	41-65
6	66+

However, these data are not reliable because, as can be seen in the figure C.1, most routes are performed by the user with ageRange value 0, that is, of undetermined age.

3.2.1.2 user_type

This is a variable that tells us the type of user that made the movement. According to the information of the EMT the possible values that can have are seen in the table 3.2.

Table 3.2: Description of the user_type values.

user_type value	Type of user
0	Indeterminate
1	Anual user
2	Ocasional user
3	EMT employee

The trips performed by the employees of the company should be dismissed and not by the means of communication by the users, therefore for the study we discard all the users_type 3.

3.2.2 Stations data

This data provides information every hour on the status of bicycle stations. The data is provided in a json containing the hourly identifier and an array with the information of all the stations, figure 3.4 The data stored in the json will be processed and saved in a

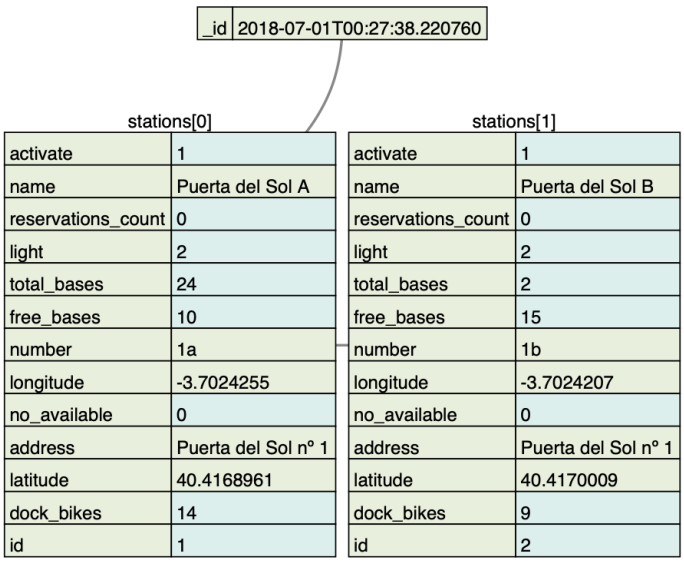


Figure 3.4: Json structure of the station dataset.

dataframe. In a similar way to the procedure with the routes, the fields that are going to be useful must be selected.

- From the variable **_id** you can obtain information regarding the time and day in which the data was checked. This data will be stored in **Hour**, **Day** and **Month**.

- The variable **activate** tells us if the station is active for its use, in our case we will suppose that all of them are so it is **discarded**.

- The **address** value contains the name and number of the Madrid street where the station is at.

- The value of **reservations_count** is the number of bikes reserved by users through the BiciMAD application, since our model will not have this functionality is not useful.

- **total_bases**, **free_bases** and **dock_bikes** indicate the total number of bases available to the station, those available at the moment and the free bikes to use respectively.

These data will then be used to indicate the initial situation of the model based on the real usage data.

- The **latitude** and **longitude** values show the geographical position of the station, which will be used for the generation of the routes through the ORS API; and the visualization of the system.
- The **id** value indicates the unique code associated with each BSS, it will be used as an identifier.
- The rest of the data does not provide information to the system, so they will not be used and will not be included in the DataFrame.

The final result is a dataframe similar to that of the figure 3.5, which contains information about 167 stations. [2]

	id	address	dock_bikes	free_bases	total_bases	Hour	Day	Month	latitude	longitude
8829	3	Calle Miguel Moya nº 1	0	24	24	4	1	9	40.4205886	-3.7058415
8830	3	Calle Miguel Moya nº 1	1	23	24	3	1	9	40.4205886	-3.7058415
8831	3	Calle Miguel Moya nº 1	0	24	24	2	1	9	40.4205886	-3.7058415
8832	3	Calle Miguel Moya nº 1	2	22	24	1	1	9	40.4205886	-3.7058415
8833	3	Calle Miguel Moya nº 1	5	19	24	0	1	9	40.4205886	-3.7058415

Figure 3.5: Structure of the station dataframe.

3.2.3 Probability array and matrix creation

To adjust the behavior of the agents, the behavior of the real users must be studied. This will be done through the data obtained on the routes above. However, the behavior is not homogeneous throughout the day.

In the figure 3.6 we can distinguish different peaks of use by users. These peaks are mainly associated with the use of the service as a method of transport to and from work or university, and three cases can be differentiated:

- From 06:00 to 10:00
- From 13:00 to 16:00
- From 18:00 to 21:00

[2] 171 starting stations, 4 were discarded due to they were inoperative during the study period.

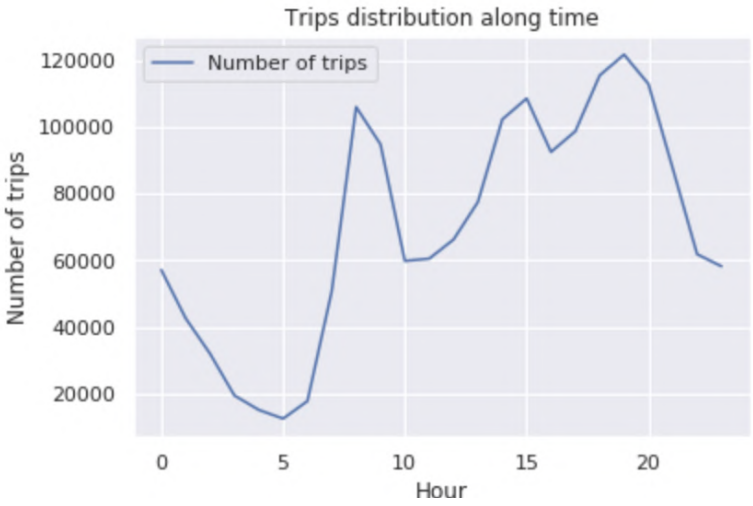


Figure 3.6: Trips distribution.

In addition to these peaks, a default behavior is assumed in the rest of the hours. As we are only going to take into account the behavior of the users in the working days we filter the data of the routes according to their value from the column **"weekday"** so that only use lower values between 0 and 4. To obtain the origin and destination station of the users, it will consist of two steps, the first one obtaining the origin to later obtain the destination address based on the obtained origin station.

3.2.3.1 Origin probability array

The probability of being the origin for station i will be obtained based on the following formula:

$$Pi = Ni/Nt$$

Where **Ni** is the number of routes originating in i and **Nt** the total number of routes performed. The final result is an array for each time group where in each column is the probability of each station.

3.2.3.2 Destination probability matrix

The probability that station j is chosen as the destination is obtained through:

$$Pij = Nij/Nt$$

Where **Nij** is the number of routes originating in i and destination in j and **Nt** the total number of routes performed. This returns a matrix where each column contains the probabilities of going from a certain station to the rest.

3.2.4 Initial state creation

An initial state of the model is needed in order to display an average situation of the system. To calculate this starting point, only the values of the days from Monday to Friday and taking into account that our system starts the simulation at 00:00 hours just this time will be selected. To obtain the initial situation of the stations, an average value of the dataset of the stations calculated above will be obtained based on the following formula.

$$Ndmean = Ndi/Nd$$

Where **Ndmean** is the initial state of the dock bikes variable, Ndi the sum of all the values of the records of station i under the conditions above. Nt is the number of days meeting the filters.
As calculated above the number of free bases would be given by:

$$Nfmean = Nfi/Nd$$

Where **Nfmean** is the initial state of the dock bikes variable, Nfi the sum of all the free bases of the records of station i under the conditions above. Nt is the number of days meeting the filters.
The amount of total bases will consist of the sum of free bases and bikes available.
To calculate the number of trips that will be simulated every hour, the dataset of the real routes will be used, according to the following formula.

$$Ntrips = Ntripsi/Nd$$

Where **Ntrips** is the number of trips to simulate each hour, Ni the count of all the real trips performed at each hour. Nt is the total number of days under the conditions.

As a summary, in this chapter three different data groups have been obtained that are necessary for the execution of the model:

- The set of probabilities for the choice of routes according to the different temporal groupings.
- The initial state of the stations.
- The number of trips to be deployed.

CHAPTER 4

Model description

4.1 Introduction

A brief introduction will be made about the structure of the agents, explaining each one of their variables. The operation of the three models and their actions will also be explained. The elements that will be seen are the following:

- **Model**. The model is the main orchestrator of the system, it is in charge of adding, deleting or modifying the agents as well as initializing and controlling the schedule and the space.
- **BikeAgent**. It is the main agent of the system, representing the BSS users who try to make a trip through the service.
- **StationAgent**. This agent represents the stations, they are passive and reactive.
- **TruckAgent**. They are the agents that represent the rental company workers who travel throughout the city relocating the bicycles between stations.

In addition, we also have to comment on the function of the **space** and **schedule** modules. The first contains information about a continuous map in which the agents are located. There are different implemented methods that allow: add an agent, move it, delete it or get a list of all the agents of each type.

As for the schedule, it is the model's clock. Indicates the order of execution of the agents, in this case the "RandomActivation" module of MESA has been selected, which makes a random execution among all.
These modules are connected as the structure shown in Fig. 4.1

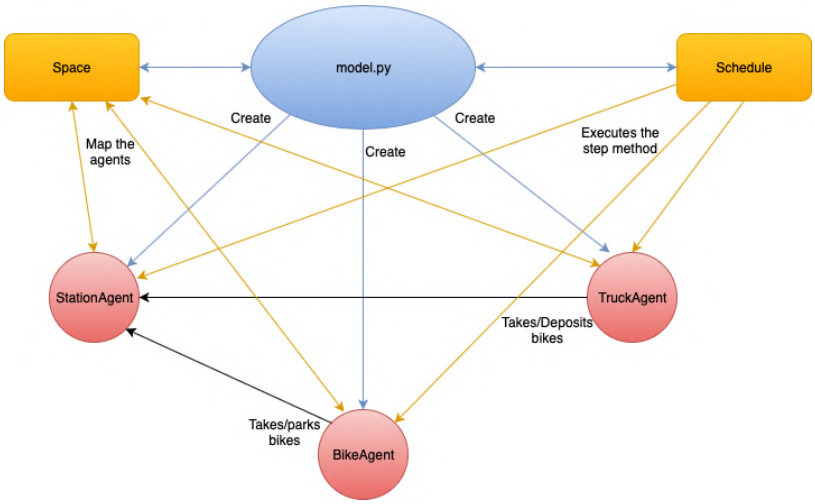


Figure 4.1: Model flow diagram.

4.2 ABM modules

4.2.1 Model

The first thing it does is loading the dataframe of the initial situation of the stations in order to create the StationAgent.
He is in charge of creating BikeAgents each model step, calculating its destination and origin station. Once created the agent is added to the schedule and to the space.
It is also responsible for obtaining data about the status of the simulation that will later be shown in the visualization. Its internal variables are those that are defined in the table. 4.1.

In each step of the model functions are executed achieving the following changes:

- Execute all the elements that are added to the model's schedule, calling the step method of all the agents.
- Create a number of BikeAgent agents based on the hour and add them to the schedule and space. It also assigns a source station and another as a destination.
- The time is updated.

Table 4.1: Internal variables of the model

Variable	Default Value	Meaning
running	True	**Boolean** which shows if the model is running.
number_trips	[ ]	**Array** containing the number of trips realized each hour. These data is generated from the route dataset, using a 25% of the mean real trips.
day	0	**Integer** defining the day of the simulation environment.
hour	0	**Integer** defining the hour of the simulation environment.
minutes	0	**Integer** defining the minutes of the simulation environment.
failures	0	**Integer** indicating the number of trips failed
finished_routes	0	**Integer** indicating the number of succeeded trips
cnt_agents	0	**Integer** defining the number of agents created
stations_revised	[ ]	**Array** containing the id of the stations where a truck had been deployed.
walking_max_dist	250	**Integer** defining the maximum distance walking by foot to another station without incentive.
riding_max_dist	1000	**Integer** defining the maximum distance to ride the bike to another station without incentive.
model_type	1	**Integer** defining the model to execute.

4.2.2 BikeAgent

Its variables represent its status, position and route to follow and are explained in greater detail on the table 4.2

There are three different types of behavior for the agent depending on the variables

Table 4.2: Internal variables of the BikeAgent

Variable	Default Value	Meaning
unique_id	0	**Integer** defining the unique identifier of the agent
id_dest	0	**Integer** of the destination station id.
id_orig	0	**Integer** of the origin station id.
route	[ ]	**Array** containing the route generated by the ORS application.
cnt_route	0	**Integer** used to control the next point of the route to move.
checkin	False	**Boolean** indicating if the agent has already taken a bike
checkout	False	**Boolean** indicating if the agent has already parked the bike
pos	[0,0]	**Array** containing the actual latitude and longitude.
duration	0	**Integer** defining the amount of meters traveled.
distance	0	**Integer** defining the number of seconds it has taken to travel.

checkin and **checkout**, as shown in Fig. 4.2, and described below:

- If both are false: the user tries to get a bike. As soon as he gets it the value of moving is changed to true.
- If checkin is true: the agent moves towards his destination.
- If checkout is true: Once reached the destination tries to park the bike and then the agent is eliminated.

The behavior of the agent is explained in more detail according to the model in the following section, focusing on the differences between them.

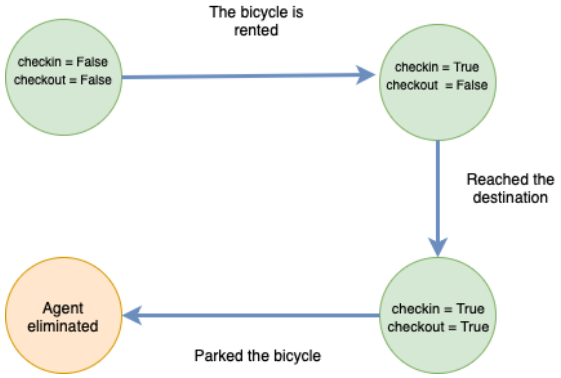


Figure 4.2: State flow of BikeAgent.

4.2.3 StationAgent

This agent represents the stations, in each step it updates its priority. This variable is an indicator about the state of charge, which will be used to indicate priority for a balancing action. The meaning of the values of the priority variable is explained in the table 4.4 and the internal variables of the agent are explained in the Table 4.3.

4.2.4 TruckAgent

The TruckAgent is responsible for redistributing bikes between stations. Its main mission is to balance the load and get fewer stations out of service for lack of bikes or docks. Its variables are broken down on the table 4.5. The flow between agent states based on the value of working and moving variables can be seen in Fig. 4.3.

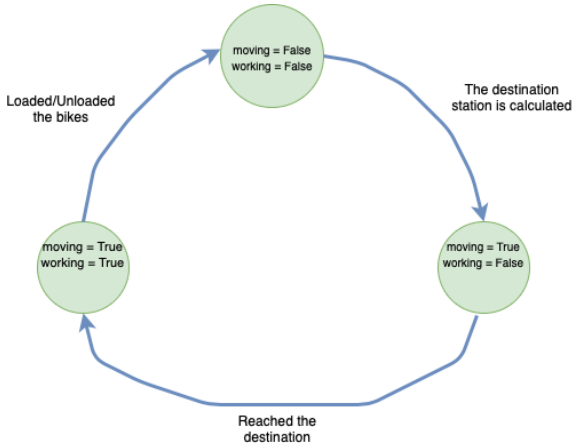


Figure 4.3: State flow of TruckAgent.

Table 4.3: Internal variables of the StationAgent

Variable	Default Value	Meaning
index	0	**Integer** defining the unique identifier of the agent
available	True	**Boolean** representing the state of the station.
address	“ ”	**String** indicating the name of the station address.
free_bases	0	**Integer** defining the quantity of free bases.
dock_bikes	0	**Integer** indicating the number of bikes available.
total_bases	0	**Boolean** indicating the total amount of bases.
pos	[0,0]	**Array** containing the latitude and longitude of the station.
priority	0	**Integer** that defines the situation of the station based on its load.

Table 4.4: Priority of the StationAgent

Priority	Value	Trigger
Low	2	<5 bikes avaliables
High	1	<5 bases avaliables
Normal	0	≥5 bikes avaliables & ≥5 bases avaliables

Table 4.5: Internal variables of the TruckAgent

Variable	Default Value	Meaning
index	0	**Integer** defining the unique identifier of the agent
moving	False	**Boolean** representing if the truck is moving to a station.
working	False	**String** representing if the agent is loading or unloading bikes from the truck.
pos	[0,0]	**Array** containing the actual latitude and longitude.
id_station_dest	0	**Integer** indicating the id of the target station.
route	[]	**Array** containing the route of the truck to reach the destination.
cnt_route	0	**Integer** used to get the next route point.
returning	False	**Boolean** that indicates if the truck has to finish its job.
cnt_st	0	**Integer** that counts the number of steps that the truck is waiting.

4.3 Models

Three models have been created that are governed by different policies with the intention of reaching conclusions on how to improve the service.

- **Base model**: It is a base model of BSS, agents perform their journeys from their initial station **i** to the final station **j**. If it is not possible to make the trip you can move a distance to another station to either take a bike or park it.
- **Incentive model**: The agents, in exchange for an economic retribution, change their origin and/or destination stations so that, the bikes are rented in high load stations and are returned at the low ones.
- **Truck and incentive model**: In addition to the previous incentive model, the TruckAgent is included, which is actively responsible for the relocation of the bikes, distributing the load.

4.3.1 Base Model

This will be the model that will represent the behavior of cyclists. It would be the case of an environment in which rebalancing measures are not carried out.

4.3.1.1 BikeAgent behavior

When an agent is created the model has already assigned a destination station and an origin. In this model according to checkin and checkout values:

- If checkin is false it means that the agent has not yet rented a bike. This checks if bikes are available at the origin station if so, it is rented. However, if not, the agent checks which station is closest to the original one and, if the distance is less than the one we have parameterized in the slider **Max. distance to walk** updates its origin station and the new route is calculated. The walking route between the old and the new origin station is not taken into account since during this time there is no use of the BSS.

- If checkin is true but checkout is false the agent is in the realization of its route, so it moves, following the calculated route.
  Once it reaches the end of its journey it checks if there are any base available at the destination station.If there were, the agent parks the bike, otherwise he would look for the nearest station with available bases and, if this was a distance less than the one

introduced in **Max. distance to ride** would become the new destination station. The agent recalculates a route between its current position and the new destination.

- When checkout is true, the agent has parked the bike, so the information of the station is updated and the BikeAgent is eliminated, both from the space and from the schedule.

4.3.2 Incentive Model

Now incentives will be introduced to the users that help to improve the load balance. This means that they will receive economic rewards as a discount on the use of the service in exchange for renting bikes from a heavily loaded station or parking them in one with a very low load.
In this model, incentives of 0.5 euros will be used, so that according to [15] a participation is achieved by half of the users.

4.3.2.1 BikeAgent behavior

The agent before starting his trip gets the situation of the stations, obtaining which is the closest with high load. It is calculated if the agent participates in the incentives policy and depending on the distance you decide if the origin station is changed according to the curve in ref fig: incentives.
On the other hand, the destination station is also checked, obtaining the closest one with low load. As the speed of the bike is substantially higher we assume a greater predisposition of the users to park in more distant places. This is why we consider the maximum distance of the users to accept an incentive of 1.5 times higher.
Once the route has been established, whether there have been changes or not, the trip is made in the same way as the base model.

4.3.3 Truck+Incentive model

An active balancing element are the trucks, these help to improve the distribution of bicycles by relocating them from stations with excess bicycles to others that lack them. In addition, there is the passive system of incentives explained above, working at the same time.

4.3.3.1 Model

In order to optimize the work and with a foresighted object before large peaks of use, the number of trucks will follow an hourly distribution similar to the number of trips. In

addition they will also be deployed at night to reposition the bikes during the hours of lower traffic and use of the service.

4.3.3.2 TruckAgent behavior

The first to be checked by the agent is its variable **returning**, if **true** the truck understands that it must stop its service, and there are two options depending on whether they have bicycles loaded or not:

- If the truck still has bicycles, look for a station with low load to go to download them.
- If not the agent is removed from the space and the schedule.

If **false**, the behavior of the agent can be divided into three blocks, depending on the value of its variables **moving** y **working**.

- If both are false you get a destination based on your load. If you have less than 3 bikes, look for one with high priority, otherwise you will go to one with low priority. If both are false, a destination is obtained according to its load. If it has less than 3 bikes, one with high occupancy is sought, otherwise it will travel to one with low load.
  The choice of station is based on the number of bikes or free bases, depending on the case, and the distance, according to the following formula.

$$P = D + Nv * 200$$

  P being the score, D the distance to that station and Nv the number of free bases or bikes available. The one with the lowest score will be chosen.

- If moving is true, the truck will move to its destination at the determined speed. Once it reaches its destination, the working variable is activated.
- When working is true there are two possible scenarios for the agent.
  If the priority of the station is high take a bike by step until the priority changes or the truck is full. If the priority is low, leave one bike per step until the station regains a normal load or the truck is empty.

CHAPTER 5

Architecture

5.1 Introduction

In this chapter we will explain the architecture of the system, explaining the operation of the ORS routes service and the connection with the Tornado web server. First we will proceed to explain the interconnection of the model with its different elements and then a general scheme including the new modules explained.

5.2 System modules

5.2.1 ORS

This library, developed by GIScience Research Group [1] in Java, offers a service on demand routes. Through its API, executable in a docker, we can make requests with the parameters. For our model we have two different types of routing, that of the bikes and the trucks. The users, usually choose a **recommended** route to avoid stretches with a very high slope or with great traffic. Also, not all roads are available for **bicycle** traffic.
In the case of trucks, their operation is different. These agents seek the realization of the largest possible number of routes, which is why the preference variable will take a value **faster**. Also to be a motorized vehicle will use the profile **driving-car**.

[1] https://github.com/GIScience

Table 5.1: Parameters of the bike route request

Variable	Default Value	Meaning
Profile	cycling-regular	Cycling profile, varies the speed and the roads to travel
coordinates	[x,y][x,y]	Origin and destination coordinates
preference	recommended	Type of route (recommended, faster, shorter)

These GET requests are made by the agents involved (TruckAgent and BikeAgent) to the server deployed by the docker at the localhost address [2]. The server responses are in json format, and contain:

- An array of points representing the route to follow.
- The total distance to travel in meters.
- The total duration of the trip, taking into account the speed according to the sections to be covered.

The information about distance and duration will be used to study the variability of these two factors in each model.

5.2.2 Tornado Web Server

Tornado [3] is a Python web framework and asynchronous networking library that is responsible for creating a communication between different systems, in this case the **server**, where the model is executed, and the **client**, formed by the web interface.
In this web interface you can include different visual elements, in this case we will use the library Chart.js [4] to show graphs about the situation of the model. In addition you can also include objects to allow the user to vary the data entered in the model, some types are:

- Sliders.
- Checkbox.
- Choice selection.

[2] http://localhost:8080/ors/routes
[3] https://github.com/tornadoweb/tornado
[4] https://www.chartjs.org

5.2.2.1 Server

The server renders an HTML page and includes the code js to initialize the graphic objects.

- When the sockets are created, a message **get_params** is received, with a request to obtain the parameters. The server returns a message **model_params** with the array of elements.
- The value of these elements is received through a message of type **submit_params** and, they are loaded into the model when it is started.
- When it is received a **get_step** type message, a new step of the model is executed and the collected information is sent in a **get_situation** type message.

The information sent about the situation has the structure of the figure 5.1. In this message is provided the state of the time and day of the model, the stations, the bikes and the trucks.

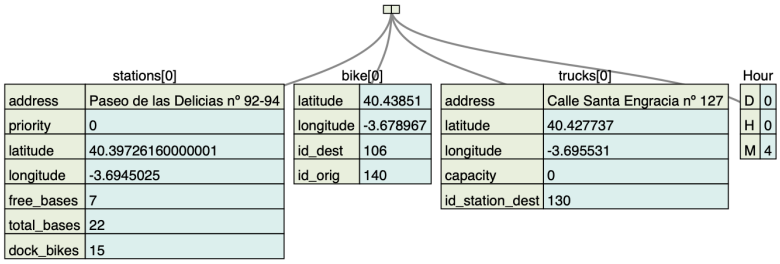


Figure 5.1: Structure of the model situation message.

5.2.2.2 Client

The client is in charge of rendering all the elements of the folium map, this is explained through a set of methods that allow you to update the data, delete and add the markers. When started, the client sends a message **get_params**, as explained before, requesting the interactible elements to render to the server. Each time these elements are modified they send a message **submit_params** with the values, which are loaded in the model once it is restarted, through the **Reset** button.
When the **Step** button is pressed, a message is sent to the server asking for a new step to be executed, this message type is **get_step**. To automate this process the **Start** button

is avaliable where, when you press it, a **setInterval()** method sends the message every so often. The time of this interval can be modified through the slider **Frames per second**. In reception we can have different types of data, so we categorize the behavior according to the type of messages:

- If **model_params** is received, the parameterizable elements are rendered.
- In the case of being **get_situation** the data of the display module is updated
- If the message is of type **viz_state**, which contains the to plot information of that step, the graphs are drawn with the new values.

The main message that the client receives is of type **get_situation**, since it contains all the information of the model to be loaded in the DOM. To do this, a visualization module is used that performs the following functions:

- Create the Leaflet map.
- Update data of the agents.
- Set and update the stations', bikes' and trucks' markers.
- Set and update the hour and day info.

For the visualization of the data in real time, 4 graphs have been added, indicating a comparison of the following data:

- Comparison of high and low load stations.
- Successful trips.
- Check-in and check-out incentives count and number of trip fails.
- Number of current active bikers.

In addition to modifying the variables of the model, different modifiable elements have been added, in the table 5.2 each of them is detailed.

With this we get an interface like the one that can be seen in the figure 5.2. In the upper part we find the navigation bar, where we have the buttons **Start**, **Step** and **Reset**. After the bar we can see the map as the main element, in which are the agents' markers, which will be modified with each step of the model. Below the map we find the modifiable elements.
On the right we find the graphs along with their legends.

Table 5.2: Variable elements in the web interface

Element name	Values	Meaning
Frames per seconds	0-20	Speed to execute the model, in each frame a model step is ran.
Model to execute	Base, Incentive and Truck+Incentive models	Choice input where the model to use can be selected, if none selected the base model is executed.
Percentage of real trips (%)	0-100	Percentage of the real users mean to use in the model, recommended 25%.
Max. distance to walk	0-1000	Maximum distance to move by-foot when the origin station is empty.
Max. distance to ride	0-2000	Maximum distance to move by-bike when the destination station is full.
Number of initial trucks	0-10	Number of trucks to be created when the model initiates.
Truck speed (km/h)	0-80	Truck speed, considered as average.
Bike speed (km/h)	0-30	0-30 Bike speed.

5.2.3 Module interconnection

Now that the different modules have been described, the connection between them will be explained, for that purpose it will be divided into two modules, **execution of the model** and **visualization**.

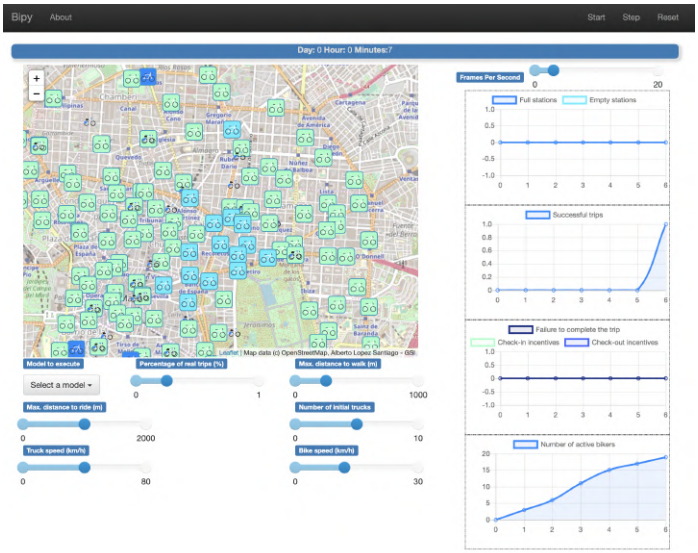


Figure 5.2: Web interface visualization.

5.2.3.1 Model execution

The actors in this part are the model and the different agents along with the space and the schedule. In addition, as already explained before, the BikeAgent and the TruckAgent communicate with the ORS service to obtain the routes, the final architecture is the one seen in fig. 4.1

5.2.3.2 Visualization

In the visualization part all the elements that have been commented before in the Tornado subsection, the objective of this architecture is to visualize the data generated during the execution of the model. The diagram can be seen in the fig. 5.3.

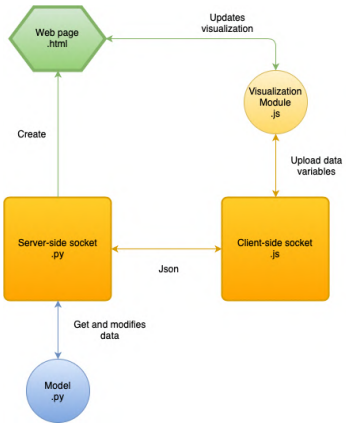


Figure 5.3: Visualization interconnection diagram.

CHAPTER 6

Case study

6.1 Introduction

The main objectives of this project are the study of the influence that have different behaviors for the stability of BSS, as well as how this instability affects the geographic distribution of Madrid.
To obtain results, we will perform 50 simulations of a 7-day use situation. Since the ABM represents 1 minute per step this would give us a total of 10080 steps per simulation.
We will also set some variables to be able to compare the results, table 6.1.

6.1.1 Results of the simulation

For the analysis and the conclusions we will study about:

- The distribution of distance traveled and duration of the trips of the BikeAgents.
- The evolution of the stations load.
- The percentage of trips completed successfully.

One of the points of study is the deviation of the duration of the trips with respect to the real data. This will help us determine the degree of similarity between the model and the real scenario.

Table 6.1: Variables values

Variable	Value
Bike speed	18 km/h
Truck speed	60 km/h
Number of trips	25% of real trips
Number of trucks	5
Probability	Mon-Fri prob.
Stations start-up	Mon-Fri average

We have a collection of over 1.5 million real routes and approximately 2,375 million simulated trips.

6.1.1.1 Distance and duration

In the comparison between the duration of the simulated routes and the real ones, shown in fig. 6.1 a decrease in the average time of travel is observed. Although the normal distribution is very similar, the simulation reaches a prob peak. of **0.0015** in its mode, while the real data records a prob. from **0.0012**. The evolution of the duration for each model, fig. 6.1 shows how in the base model the duration undergoes a smaller variance than in the other two models. This means that in this model trips suffer less variations. In the other two models we observe how the normal one is displaced slightly to the left, which means a decrease in the average time of the trip.

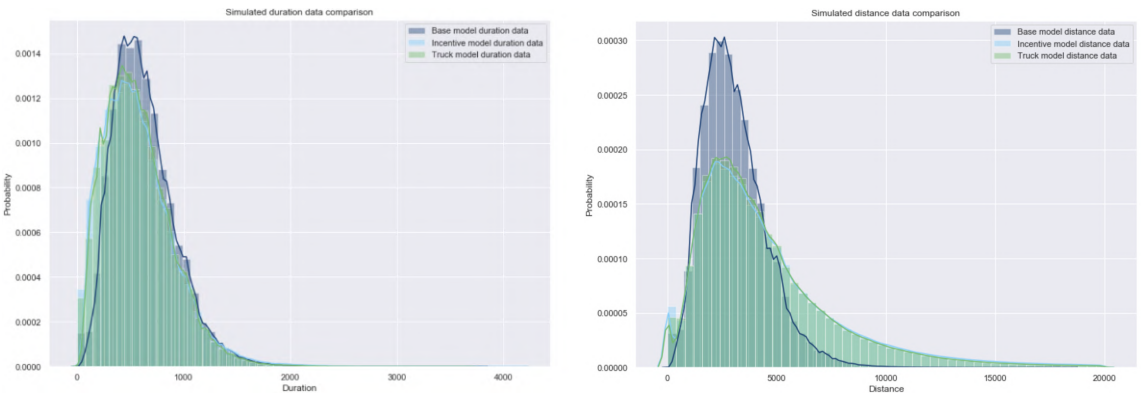


Figure 6.1: Trips duration (left) and distance (right) distribution.

In the table 6.2 we can see the average duration of the trips for each model and for the real data.

In the case of distance, there are no records of the use of the BiciMAD service, so the difference between the three simulated models will be analyzed.
In the figure 6.5 we find the distribution of the distance traveled by the agents. As we

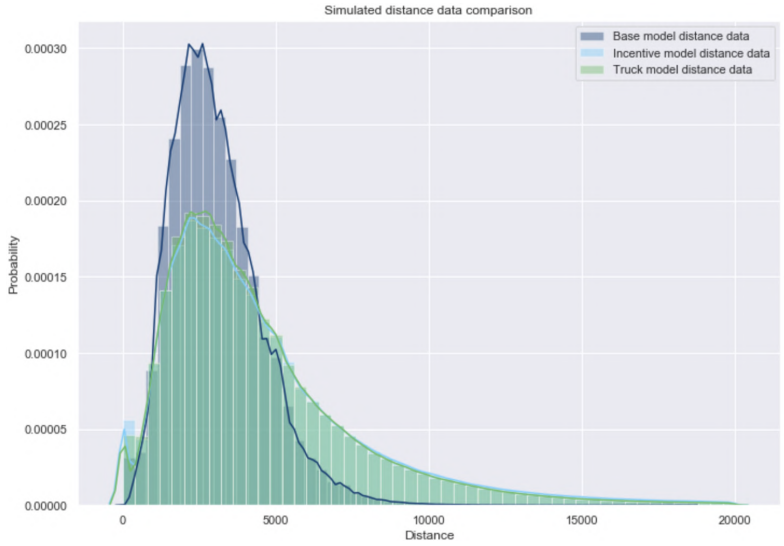


Figure 6.2: Models distance comparison.

can see in the figure, analogous to what happened with the duration, the base model gets smaller value range than the other two models, which have a much higher variance. In the first case the distance reaches maximum values of 10,000 meters , being doubled by the models with incentives and with trucks. You can find in the table 6.2 information about the average duration in each model.

From the data analyzed above, we can assume that although the trips are of a greater distance in the models with balancing, the agents take a smaller amount of time to travel them, so we assume a better optimization in the routes.

6.1.1.2 Station load

For the study of the load of the stations we will take into account the same condition as in the model, if there are less than 5 bikes the station is with low load, while if there are less than 5 free bases it is with a high load.
In order to study the behavior correctly, we will analyze it once it has reached a more stable behavior. In the figure 6.3 we find the evolution of the 3 models.

Table 6.2: Average distance and duration

Model	Mean distance (meters)	Mean duration (sec.)
Base model	3031.7	623.02
Incentive model	4771.93	565.85
Incentive+Truck model	4541.25	564.33
Real data	-	831.5

As can be seen during the first 24 hours of the model, the amount, both of loaded and empty stations, increases linearly. Once these hours have passed, the model becomes more stable and fluctuations in the number of stations of both characteristics begin to appear. Therefore for the calculation of the improvement between models we will use the data from the second day.

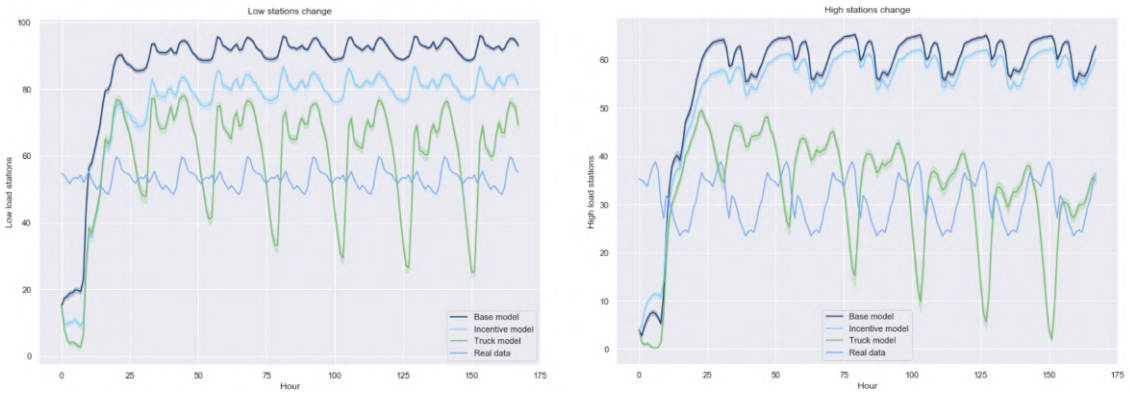


Figure 6.3: Number of low (left) and high (right) occupancy stations.

In the table 6.3 we can find the average values, for the three models, where the improvement in the load of the stations can be observed.
Taking as reference the data obtained in the base model we get:

- In model 2, a decrease of 13.7% in the number of stations with low load and, a 4.86% in the number of stations high loaded.

- In the case of the third model we have a reduction of 32.2% in the number of almost empty stations and 44.7% in those that are very busy.

Table 6.3: Low and high load stations average

Model	Low stations average	High stations average
Base model	91.80	61.40
Incentive model	80.14	58.42
Incentive+Truck model	62.23	33.98
Real usage data	53.18	30.95

6.1.1.3 Completed trips and failures

The study of this variable will allow us to analyze with what percentage of success a user can use the service. An increase in the number of trips will indicate an improvement in the situation of the system, which impacts on a better user experience. We can find the evolution, both of the trips made and those that could not be made due to failures, in fig. 6.4

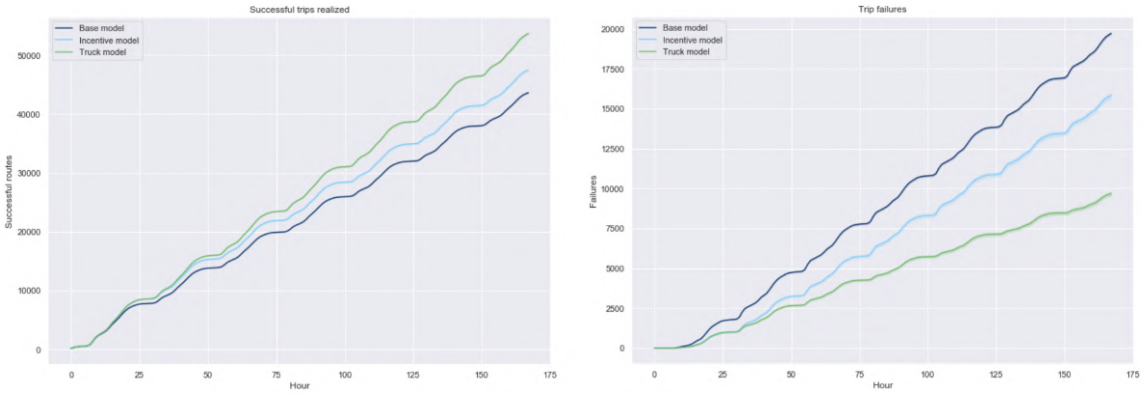


Figure 6.4: Successful (left) and failed (right) trips evolution.

An average of 64,300 trips are estimated for each execution of the model, so according to the data, the success and failure percentage in each of the models can be found in the table 6.5.

A great improvement can be observed in the number of trips satisfactorily completed, obtaining an improvement, in the third model, of up to 23.25% on the base model.

6.1.1.4 Incentives given

The number of incentives indicates the participation of the people. Due to the distance difference mentioned above, the number of incentives given to agents who have parked their

Table 6.4: Successful and failed trips percentage

Model	Success rate	Failure rate
Base model	68.58%	32.42%
Incentive model	74.47%	25.53%
Incentive+Truck model	84.40%	15.60%

bike in a station with low occupancy is greater.
In the latter case, we obtain an average of 21,800 incentives, representing a 33.9% share compared to the 64,300 total trips. In the other we obtain 12,500, which implies 19.44% of the total.

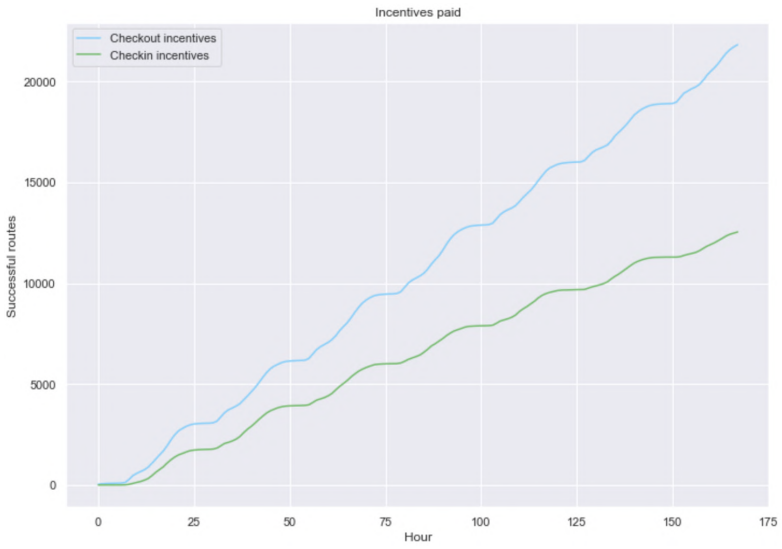


Figure 6.5: Incentives paid evolution.

6.2 Impact on Madrid

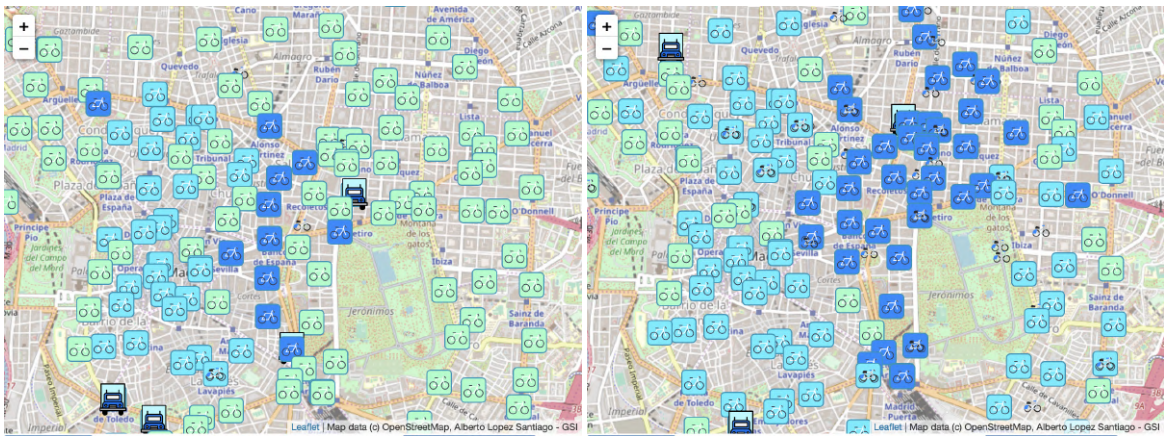
To analyze the impact on the city of Madrid, and to be able to reach a solution that improves the system, we must set different moments of time. To carry out this test, three different points will be checked at 4:00 AM, at 10:00 AM and at 20:00 AM. For the analysis we will use the values to execute the system seen in the table 4.5

Table 6.5: Variable values in the simulation

Variable	Value
Model	Truck+Incentive model
Perc of real trips	25%
Max. distance to walk	250 (m)
Max. distance to ride	1000 (m)
Number of initial trucks	5
Truck speed	40 (km/h)
Bike speed	12 (m/s)

6.2.1 South zone

In the figure 6.6(a) a stable initial situation is observed, at 4:00 AM, where the number of very charged or empty stations is very low.
In the image (b) there is a high concentration of full stations around Paseo de la Castellana, Cibeles, Colón business concentration and tourist zone. We also have a large concentration of stations loaded in the area of Callao, which can be caused by the use of tourists.
In the figure 6.8 (a) we find the situation at 20:00 where we see how the area of Paseo de la Castellana is still saturated, there being no significant changes in the area.

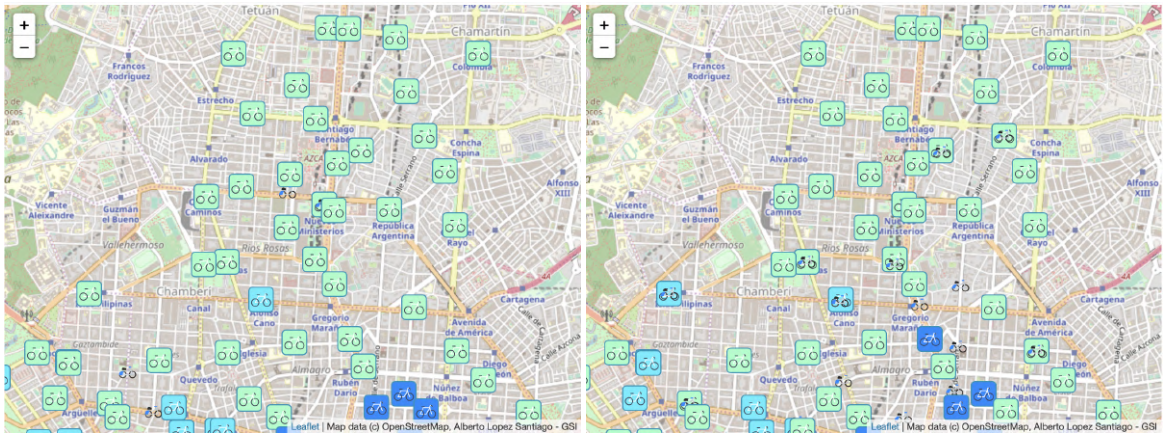


(a) 4:00 (b) 10:00

Figure 6.6: 4:00 AM (left) and 10:00AM (right) situation. South zone.

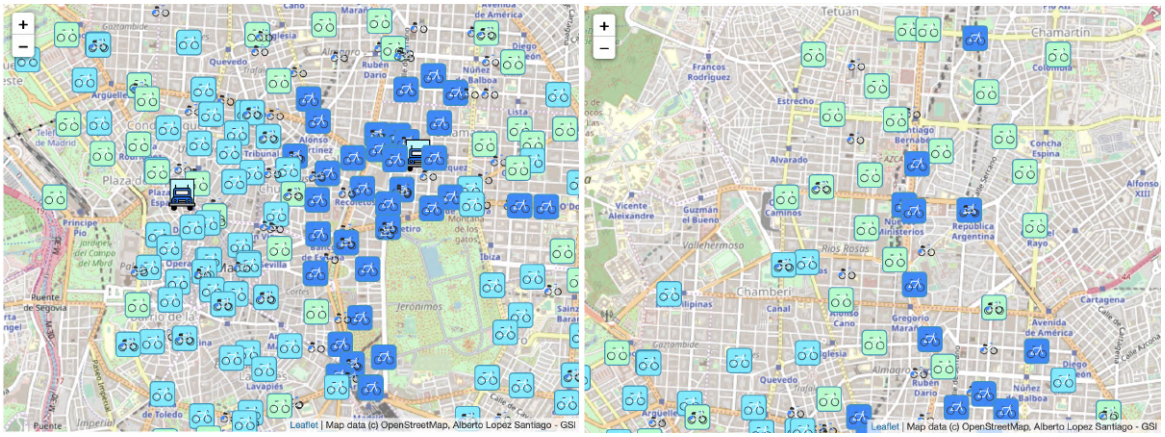
6.2.2 North zone

In the northern area of Madrid there are areas such as Ríos Rosas, Tetúan and Chamartín. In these areas we can find both business and residential buildings.
In the same way, starting from an initial situation at 4:00 6.7 (a) we there are hardly any changes in the situation at 10:00. However we see how at 20:00, fig. 6.8 (b) all along the area of Paseo de la Castellana street, we find several full stations. In addition, the neighborhood of Gaztambide has increased the number of stations with low load.



(a) 4:00 (b) 10:00

Figure 6.7: 4:00 AM (left) and 10:00AM (right) situation. North zone.



(a) North zone. (b) South zone.

Figure 6.8: 20:00 situation, north zone (left) and south zone (right).

CHAPTER 7

Conclusions and future work

Finally we will explain the conclusions that we have reached through the study. The objectives that have been made will also be commented on. By last, different future research lines will be proposed for the project.

7.1 Conclusions

In this thesis a web application has been developed, where Agent-based Social Simulations (ABSS) could be visualized. The application allows the user to see an overall state of the system, as well as analyze the data. It is also posible to adjust the value of some model variables.

The application uses Leaflet to draw the map and Chart.js to plot the graphs. Python was used to create the server and perform the simulations, along with the Mesa framework. This software could be used for other projects, since the visualization of the data facilitates the analysis of the results, being used for the simulation in Madrid or adapted to another city.

In order to verify the functionalities of the system, a multi-agent social model has been developed, which seeks to simulate the traffic produced in the BSS. The motivation arises from the growth in interest in a renewal of the transport sector, as well as the large number of companies in the sector that have appeared.

Understanding the flows of the system helps to know what improvements can be imple-

mented in order to achieve greater availability in the service. The public companies of bicycle rental usually publish in an open way the data of use, reason why the program can be adapted for the analysis in another city.
The inequality in the occupation of the stations has been reduced, reaching average values similar to the real ones. Taking the system as an approximation to reality we can establish a good starting point for the development and analysis of new policies that can be implemented.

During the accomplishment of this project we have achieved the following objectives:

- Development of an open-source software that allows the execution of the model and the editing of parameters in a WEB environment.
- Development of an autonomous ABSS for the simulation of bicycle traffic in a BSS.
- Load balancing policy study.
- Study of load distribution throughout the geography of Madrid.

7.2 Future work

As possible lines to investigate in the future the following have been remarked:

- Dataset increase. Currently only 4-month data is available for the creation of probability matrices; increasing the number of data would allow the creation of a more accurate pattern of behavior to the real.
- Analysis of other cities. Currently the study is done on the geography of Madrid, adapting the project to other cities can generate an improvement in the system
- Creation of new models. Applying new policies to rebalance traffic as gamification.
- New types of agents. At the moment only the traffic of the bicycles is contemplated, nevertheless to add other vehicles and to study its action with the BSS could be implemented.
- Incorporation of an algorithm of machine learning for the forecast of traffic that will suffer the system according to: date, events, place, etc.

APPENDIX A

Impact of this project

A.1 Introduction

Bicycle rental services have increased their numbers exponentially in recent years, reaching more than 4 different companies operating in a city like Madrid, so this type of transport has become a common means of transport. In this appendix we will try to analyse the impacts related to the realization of this project from a social, economic, environmental, ethical and professional point of view.

A.2 Social impact

Due to this type of systems are being implemented in greater numbers each time this project could be used as a starting point for future researchers who want to adapt it to other cities or BSS or want to increase the functionalities to obtain greater conclusions.

A.3 Economic Impact

In this section we will assess the possible economic impacts that users and companies using our system may experience. This also helps the renting companies, which can use the

software to study new policies in order to improve their service. Likewise, users will benefit from these new policies, obtaining a better user experience.

A.4 Environmental Impact

This section aims to define the main environmental impact of the development of our system. The main energy expenditure is derived from the computer in which the model is executed, due to the high processing capacity the energy consumed is large. This results in polluting emissions due to the generation of electricity.

A.5 Ethical Implications

In this section we will evaluate the ethical implications. This project develops functions that were previously carried out by analysts. This does not necessarily mean a destruction of jobs, but a transformation. Due to the fact that users are simulated and the limitations imposed on their BiciMAD usage data, no sensitive data of real users is used.

APPENDIX B

Cost of the System

B.1 Introduction

Now we will make a forecast of the expenses subject to the realization of this project, breaking it down into the following four aspects.

B.2 Physical Resources

The only electronic device needed to carry out this project is a computer, which has minimum processing and memory requirements. These basic requirements are mainly focused on three aspects:

- **CPU**: Intel Core I5 3.2GHz, 4 cores
- **RAM**: 8 GB
- **Disk**: 250 GB

The price of the device is variable, but over a price range of approximately 800 euros.

B.3 Human Resources

Now we will introduce the expenses related to the workers in charge of the development of the system.
For this, assuming a total of 360 hours of dedication for its realization, as stipulated by the UPM Collaboration Scholarship, at a salary of 5e per hour it results in 1800 euros.

B.4 Licences

For the realization of this project has been used only open-source software, so the expenses associated with the payment of licenses is zero.

B.5 Taxes

In the event that the software is sold, a tax of 15% will be applied on the price of the product as defined in Statute 4/2008 of Spanish law.

APPENDIX C

Annex of Figures

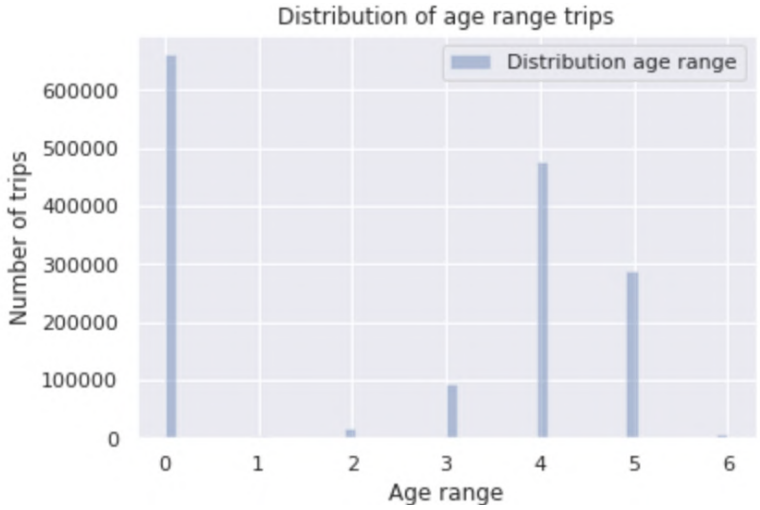


Figure C.1: ageRange value distribution, August-November 2018.

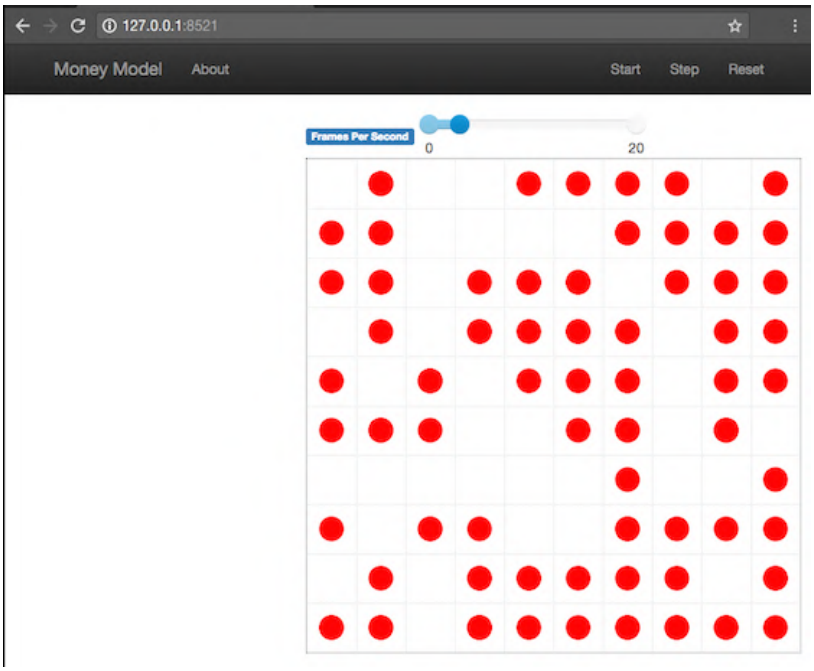


Figure C.2: Tornado server executing the Money Model.

Waytype \ Profile ->	driving-car	driving-hgv	cycling-regular
motorway	100	85	-
motorway_link	60	50	-
motorroad	90	80	-
trunk	85	60	18
trunk_link	60	50	18
primary	65	60	18
primary_link	50	50	18
secondary	60	60	18
secondary_link	50	50	18
tertiary	50	50	18
tertiary_link	40	40	18
unclassified	30	30	16
residential	30	30	18
living_street	10	10	6
service	20	20	14
road	20	20	12
track	15	15	12
path	-	-	12
footway	-	-	6
pedestrian	-	-	6
cycleway	-	-	18

Figure C.3: Speed depending on profile and way type.

Bibliography

[1] Oscar Araque. Design and Implementation of an Event Rules Web Editor. Trabajo fin de grado, Universidad Politécnica de Madrid, ETSI Telecomunicación, July 2014.

[2] Leonardo Caggiani and Michele Ottomanelli. A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems. *Procedia-Social and Behavioral Sciences*, 87:203–210, 2013.

[3] Longbiao Chen, Daqing Zhang, Leye Wang, Dingqi Yang, Xiaojuan Ma, Shijian Li, Zhaohui Wu, Gang Pan, Thi-Mai-Trang Nguyen, and Jérémie Jakubowicz. Dynamic cluster-based over-demand prediction in bike sharing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 841–852. ACM, 2016.

[4] Paul DeMaio. Bike-sharing: History, impacts, models of provision, and future. *Journal of public transportation*, 12(4):3, 2009.

[5] Andreas Kaltenbrunner, Rodrigo Meza, Jens Grivolla, Joan Codina, and Rafael Banchs. Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing*, 6(4):455–466, 2010.

[6] Simone Z Leao and Chris Pettit. Mapping bicycling patterns with an agent-based model, census and crowdsourced data. In *International Workshop on Agent Based Modelling of Urban Systems*, pages 112–128. Springer, 2016.

[7] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 33. ACM, 2015.

[8] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. Rebalancing bike sharing systems: A multi-source data smart optimization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1014. ACM, 2016.

[9] Álvaro Lozano, Juan De Paz, Gabriel Villarrubia González, Daniel Iglesia, and Javier Bajo. Multi-agent system for demand prediction and trip visualization in bike sharing systems. *Applied Sciences*, 8(1):67, 2018.

[10] J Lumbreras, M Valdés, R Borge, and ME Rodríguez. Assessment of vehicle emissions projections in madrid (spain) from 2004 to 2012 considering several control strategies. *Transportation Research Part A: Policy and Practice*, 42(4):646–658, 2008.

[11] Mieke Massink and Rytis Paskauskas. Model-based assessment of aspects of user-satisfaction in bicycle sharing systems. In *ITSC*, pages 1363–1370, 2015.

[12] J. Fernando Sánchez-Rada. Design and Implementation of an Agent Architecture Based on Web Hooks. Master's thesis, ETSIT-UPM, 2012.

[13] Emilio Serrano and Carlos A Iglesias. Validating viral marketing strategies in twitter via agent-based social simulation. *Expert Systems with Applications*, 50:140–150, 2016.

[14] Divya Singhvi, Somya Singhvi, Peter I Frazier, Shane G Henderson, Eoin O'Mahony, David B Shmoys, and Dawn B Woodard. Predicting bike usage for new york city's bike sharing system. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[15] Adish Singla, Marco Santoni, Gábor Bartók, Pratik Mukerji, Moritz Meenen, and Andreas Krause. Incentivizing users for balancing bike sharing systems. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[16] Adish Singla, Marco Santoni, Gábor Bartók, Pratik Mukerji, Moritz Meenen, and Andreas Krause. Incentivizing users for balancing bike sharing systems. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[17] Gudrun Wallentin and Martin Loidl. Agent-based bicycle traffic model for salzburg city. *GI_Forum J. Geogr. Inf. Sci*, 2015:558–566, 2015.