The Version of Record of this manuscript has been published and is available in
Journal of Simulation 14 (4), January 2020
https://www.tandfonline.com/doi/10.1080/17477778.2020.1718022

Bike3S: A Tool for Bike Sharing Systems Simulation

Alberto Fernández*, Holger Billhardt, Sascha Ossowski, Óscar Sánchez

CETINIA, University Rey Juan Carlos, Madrid, Spain

Tulipán s/n, 28933, Móstoles, Madrid, Spain {alberto.fernandez, holger.billhardt, sascha.ossowski, oscar.sanchezsa}@urjc.es

ORCID Alberto Fernández: 0000-0002-8962-6856,

ORCID Holger Billhardt: 0000-0001-8298-4178

ORCID Sascha Ossowski: 0000-0003-2483-9508

Bike3S: A Tool for Bike Sharing Systems Simulation

Vehicle sharing systems, such as bike, car or motorcycle sharing systems, are becoming increasingly popular in big cities as they provide a cheap and green means of mobility. The effectiveness and efficiency, and thus, the quality of service of such systems depends, among other factors, on different strategic and operational management decisions and policies, like the dimension of the fleet or the distribution of vehicles. In particular, the problem of agglutination of available vehicles in certain areas whereas no vehicles are available in other areas is a common problem that needs to be tackled by the operators of such sharing systems. Recently, research works have been focusing on adaptive strategies to reduce such imbalances, mainly through dynamic pricing policies. However, there is no best operational management strategy for all types of bike sharing systems, so it is of foremost importance to be able to anticipate and evaluate the potential effects of such operational management strategies before they can be successfully deployed in the wild. In this paper we present Bike3S, a simulator for a station-based bike sharing system. The simulator performs semi-realistic simulations of the operation of a bike sharing system in a given area of interest and allows for evaluating and testing different management decisions and strategies. In particular, the simulator has been designed to test different station capacities, station distributions, and balancing strategies. The simulator carries out microscopic agent-based simulations, where users of different types can be defined that act according to their individual goals and objectives which influences the overall dynamics of the whole system.

Keywords: Bike sharing, Agent-based simulation, Smart transportation, Smart mobility, Multi-agent systems.

1 Introduction

Nowadays, in urban mobility there is a trend towards limiting the use of vehicles with combustion engine, especially if they are used for private transportation. The aim is to reduce their environmental impact as well as the occupation of public spaces. At the same time, citizens are demanding flexible, individualised mobility solutions that adapt to their specific needs at any moment, and that are aligned with their environmental,

health and cost concerns. As a result, there is a growing deployment of sustainable mobility systems, with zero or low emissions and shared vehicles.

In this context, many big cities around the world are encouraging cycling mobility among their citizens, not only by improving the cycling infrastructure (bicycle lanes etc.), but also by installing bike-sharing systems (BSS). Some BSS (e.g. *BikeShare* in Seattle or DB *Callabike* in Munich) are free-floating systems that allow citizens to pick up and return a bike at any location within a certain area in town, but most systems are station-based, i.e. they rely on a set of rental stations with fixed locations. Several BSS are quite sizeable, reaching about 20000 rental bicycles in Paris (Vélib) and more than 78000 in Hangzhou or 90000 inWuhan, China (Soriguera & Jiménez-Meroño, 2020). Studies have shown that the adoption of new BSS depends strongly on the perception of their efficiency and positive effects, individually and within the community.

Both the quality of service offered to the citizens, as well as the (economic and environmental) cost of running a BSS depends strongly on taking proper management decisions. This does not only include *strategic* choices regarding the positioning and dimensioning of rental stations, the selection of adequate bicycle models, etc., but also on *operational* decisions. Resources in a BSS are limited, and not being able to find an available bike at some stations, or not being able to return it to another one due to the lack of free slots, are events that can severely deteriorate user experience in a BSS. ICT-based solutions that allow users to *reserve* bicycles or free slots at some stations palliate this problem, but proper bike balancing mechanisms are needed to attack the problem at its core.

Several research works have been done so as to optimise the use of trucks to keep the bike sharing fleet as balanced as possible, either statically (typically at night)

or dynamically (during operation) trying to match the expected demand with the available resources.

Still, it is known that there is no best management strategy for all types of BSS, so it is of foremost importance to be able to anticipate and evaluate the potential effects of such operational management strategies in a particular BSS before they can be successfully deployed in the wild. For this purpose, firstly, a particular BSS needs to be modelled at sufficient level of detail, including the positions and size of rental stations, a town cycling and street network, different user demand patterns, etc. And, secondly, the action space of BSS users and their behavioural choices need to be realistically modelled, in particular with regard to economic and social stimuli. To the best of our knowledge, there are currently no BSS simulators that fully account for these requisites.

In this paper we present Bike3S[1], a simulator for a station-based bike sharing system. The simulator performs semi-realistic simulations of the operation of a bike sharing system in a given area of interest and allows for evaluating and testing different management decisions and strategies. In particular, the simulator has been designed to test different stations capacities, station distributions, and balancing strategies. The simulator carries out microscopic agent-based simulations, where users of different types can be defined that act according to their individual goals and objectives which influences the overall dynamics of the whole system.

The rest of the paper is organised as follows. In Section 2 we discuss related work on bicycle sharing systems simulation. Sections 3 focuses on station-based BSS that we are particularly concerned with. Section 4 discusses the general architecture of the Bike3S simulator and its different elements in detail. We evaluate our proposal in

[1] https://github.com/gia-urjc/Bike3S-Simulator

Section 5, where we put forward different use cases, including the application of Bike3S to BSS scenarios for the cities of Madrid and London. Section 6 summarises our work, outlines the lessons we have learnt, and points to future lines of work.

2 Related works

Microscopic simulation has been successfully used in the transportation community for many years. Specially, several well-known tools were created for traffic simulation, such as SUMO (Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, 2012), MATSim (Horni, Nagel & Axhausen, 2016) or PTV Vissim (Fellendorf, 1994).

The state of the art on simulating bike sharing systems is not as advanced as those works on traffic simulation mentioned above. Nevertheless, the increase of popularity of this kind of transportation means is provoking that the research community is paying attention to this area.

Chemla et al. (2013) presented a discrete-events open-source simulator (OADLIBSim) for evaluating their proposed algorithms for dynamic balancing. The user behaviour for choosing stations combines walking and riding distances. If there are no available bikes at a station the user may try in a different one up to a limited number of times. The number of satisfied users is taken as a quality measure to compare different balancing strategies. Unfortunately, the simulator is no longer available.

Caggiani, and Ottomanelli (2012) proposed a model and simulator with the goal of optimising bikes repositioning and routes of carrier vehicles. In their model, a day is divided into discrete time intervals. At each interval the O-D demand is considered, and new states of stations are calculated. A user that fails to hire a bike (empty station) is removed and, if the destination is full, she has to wait to return her bike.

Romero, Moura, Ibeas and Alonso (2015) combined the use of cars, buses and bikes into an integrated transportation model. Their focus is on modelling users' transportation choices based on urban transportation infrastructures so as to analyse their effect on urban mobility. Our objective is more focused on bike sharing infrastructures, mainly oriented towards balancing strategies.

Soriguera, Casado and Jiménez (2018) developed a very complete simulation model and tool with the goal of supporting the decisions regarding deploying a BSS. The tool is based on Matlab. They include repositioning trucks and the possibility of using electric bikes. Users take and return bikes as near as possible to their locations (it is assumed that users know about resource availability). This work is probably the closest to ours. We keep electric bikes as part of our future lines. Our simulator is more general leaving most of the users' decisions (e.g. next station) up to the specific user implementation, as we will see later.

Dubernet and Axhausen (2014) created a multiagent simulation framework in which they focus on modelling users' behaviour, especially in reaction to changes in the relocation strategy. Agents have daily plans, which can be updated according to the bike sharing system state. The authors point out that agent capabilities can be easily extended.

Ji, Cherry, Han and Jordan (2014) developed a Monte Carlo simulation model to evaluate the efficiency of an electric bike sharing systems. Their goal is to obtain the number of e-bikes and batteries needed to meet the demand and recharge rates. They focus on trip generation, length and duration demand variables. Only round trips are considered in the model, although the authors state that it can be easily extended to one-way trips. In general, the overlap between this simulation model and ours is not too big: they do not deal with balancing strategies whereas the management of electric bikes is

part of our future work. Nevertheless, this model is a good reference for the future extension of our simulator.

Lin and Liang (2017) built a simulation model on top of the Arena[2] simulation software. Their goal is to obtain the optimal number of repositioning vehicles to minimise users' waiting time at stations (users wait at stations until a bike/slot is available, they do not walk to another station). The initial state of stations, arrival rate, O-D probability matrix, rental time and waiting probability can be configured. However, the repositioning strategy is fixed.

Saltzman and Bradford (2016) proposed a simulation model to analyse how the change of the initial state of the system affects the shortage of bikes or slots during its operation. In their model, users wait up to one or two minutes in case they do not find a bike, then they leave the system. They created a simulator with visual representation for the city of San Francisco.

Gámez-Pérez, Arroyo and Puente-Rivera (2019) used a simulation model to evaluate the performance of bike sharing systems in terms of number of trips and, differently to other approaches, the environmental effect of cycling (CO2 emissions). Their model includes variables such as socioeconomic growth, transportation infrastructure and urban mobility patterns. Their approach is a rather long-term simulation (10 years) aimed at a global analysis, so it targets a different application from ours.

Jian, Freund, Wiberg and Henderson (2016) proposed a method for optimising the initial allocation of bikes and docks at the beginning of the day. Their approach is based on a discrete-event simulation model (every minute). They randomly generate users at stations (time varying Poisson process) and trips according to an O-D probability

[2] https://www.arenasimulation.com/

distribution matrix, as well as trip durations. If a user cannot take a bike because the station is empty, she abandons the system, while she goes (at most three attempts) to the nearest station in case of trying to return a bike at a full station.

One of the main uses of BSS simulators is often the evaluation of bike balancing strategies, e.g., strategies to (re-)distribute bikes in the area in order to adapt to demand variations. Traditionally, bike balancing has been done by trucks that transport bicycles from some stations to others. Some research has focused on optimizing the static balancing problem (Chemla, Meunier, Pradeau, Calvo & Yahiaoui, 2013; Forma, Raviv & Tzur, 2015), where the routes of trucks at night or off-peak periods are optimised. More recently, the dynamic version of the problem has been considered, which involves predicting the demand at each station in the next period and optimizing the distribution of bikes among stations so as to maximise the number of trips (i.e. reduce the number of "no-service" situations) (Contardo, Morency, & Rousseau, 2012; O'Mahony, Shmoys, 2015; Schuijbroek, Hampshire, & Van Hoeve, 2017). While those approaches only consider trucks as the means to rebalance the bike-sharing system, there are other approaches that try to incentivise bike users to contribute to system rebalancing (Chemla, Meunier, Pradeau, Calvo & Yahiaoui, 2013; Fricker & Gast, 2012; Pfrommer, Warrington, Schildbach & Morari, 2014; Waserhole & Jost, 2016). For this purpose, prices are commonly used as incentives. For example, in the city of Madrid BiciMAD[3] grants discounts over the usual rental price if a user picks up a bike at a station that is almost full with parked bicycles or if she returns it to a station with many empty slots. Experiments with approaches that modify rental prices dynamically, based on the BSS load situation, and with social stimuli to achieve voluntary travel behaviour change of BSS clients

[3]Public bike sharing system of Madrid (Spain): https://www.bicimad.com/

have also been reported (Chemla, Meunier, Pradeau, Calvo & Yahiaoui, 2013; Haider, Nikolaev, Kang & Kwon, 2018; Pfrommer, Warrington, Schildbach & Morari, 2014).

In summary, most existing works on BSS simulation created simulators for evaluating their particular balancing approaches rather than general tools. Almost none of the tools has a visual interface. They all use Poisson probability distributions for generating demand. User behaviour on empty/full stations varies among leaving, waiting or going to another station, but usually only one of those criteria is used. By contrast, the approach presented in this paper is flexible and admits all those different behaviours within the same simulation. While most existing proposals generate demand at stations, we are more general and allow specifying points of demand everywhere, which of course includes docking stations.

In general, we present in this paper a BSS simulator that is highly configurable with many parameters at different levels: global, user models, user generation, balancing strategies. User types and strategies can be easily extended by developers. In addition, it includes user interfaces for configuring, simulating and visualisation. To the best of our knowledge, there are not any other BSS simulators as complete as the one presented in this paper.

3 Station-based Bike Sharing Systems

In this section we describe the general functioning of a station-based bike sharing system. The description is inspired by the functioning of the BICIMAD system in the city of Madrid in Spain. However, it is very general and fits many similar systems in other cities.

A station-based bike sharing system consists of a set of docking stations distributed in an area of interest (typically a city) and a set of bikes that can be taken from or

returned to a station. Stations are at fixed positions and have a set of slots where bikes can be plugged into. The number of operative slots of a station represents its capacity. At a given moment, a station may have broken slots, or some slots may be occupied by broken bikes. We do not deal with such cases. However, they could be easily modelled as a decrease in the station's capacity.

In general, when a user wants to use the system she would go to a station, take a bike and return it after some time at another station. A user may go directly to a station close to her position or she could consult some kind of information or recommendation service to find the closest station. Furthermore, we assume that there is some kind of registry of users (e.g., long-term contracts or occasional users) and also that there is some type of payment involved in the use of the system. However, these aspects are not included in our simulation tool.

Besides the possibility of simply taking an available bike at a station, we consider that it is possible to reserve a bike (or a slot for returning a bike) at a station. Usually, systems that allow for reservations block the reserved bike (or slot) for a fixed amount of time, such that no other user can use it. In the BICIMAD system in Madrid, this reservation time is 20 minutes. Thus, any bike in the system would be either "in use", "available at a station" or "reserved at a station". Regarding the empty slots, they may be either "available for use" or "reserved".

Depending on the situation of the infrastructure (e.g., the availability of bikes or slots), as well as on the users individual objectives, each user that enters the bike sharing system follows a path from an initial state to an end state within the diagram presented in Fig. 1. The proposed user life cycle is general enough to encompass many different user models to be running in the same simulation. User models are characterised

by their behaviour in the three user decision nodes (hexagons in Fig. 1): “rental”, “after taking bike” and “return”.

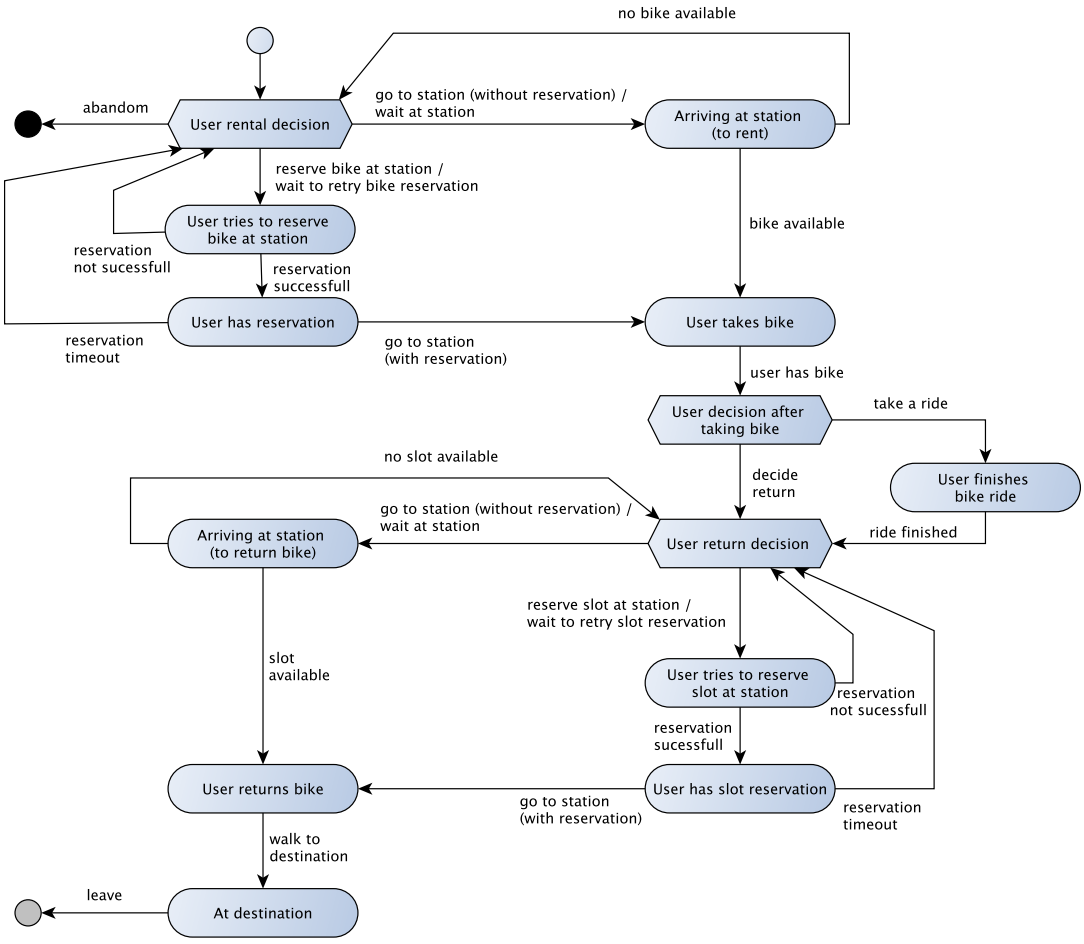

Fig. 1. User life cycle state diagram. Hexagons represent user decisions.

A user appears in the system at some time and position with the intention to rent a bike. In general, users appear at any locations in the city, not necessarily at stations. After appearing, the user handles a “rental decision” with the following possible outcomes: i) reserve a bike at some station, ii) go directly to some station in order to rent a bike, or iii) abandon the system. For reserving and/or finding stations, the user may use some available information or recommendation system. If the user decided to reserve a

bike, she tries to do so (possibly using some application). If the reservation has been successful, the user goes to the station where she reserved a bike. If the reservation was not successful, the user again has to "decide" what to do. An active reservation may finish after some timeout period, without the user having reached the station. We assume that the user is informed about such a timeout (e.g., via an app) and thus, has to retake again her rental decision. In case a user arrives at a station (without a reservation), it may happen that there are no bikes available when the user arrives. In such a case, the user has to "decide" again what is her next step. After a failed reservation intent or after arriving at a station with no available bikes, a user may "wait and retry" after some time. That is, she may try to reserve again or may wait for an available bike at the same station. Furthermore, as it can be seen in the figure, the user has always the choice to abandon the system without taking a bike. Usually, a user would abandon if she does not find a bike after trying in one or more stations, if there is no bike available at a station closed to her current position, after waiting for availability for some time, etc.

Once the user is able to take a bike (either with or without a reservation), she will "decide" whether to take some ride in the city or to go directly towards a destination point. In the second case, the user will directly take the "return decision" whereas in the first case she will decide on returning the bike after the ride has finished. In the "user return decision", she will select a station to leave the bike (usually near her destination point) and decide to either reserve a slot at that station or to go there without a reservation.

The reservation of slots is treated similarly as the reservation of bikes. The user re-decides if a reservation intent has not been successful and also if the reservation timeout has occurred. If the user goes directly to a station (without a reservation), it could be possible that, after arriving, there is no empty slot available. Also, in this case

the user has to reconsider its return decision. Again, if reserving a slot or returning a bike fails, the user may “wait and retry” again at the same station. Eventually, the user reaches a station with an available slot (with or without reservation) and can return the bike, walk to her final destination and leaves the system. It should be noted that a user cannot abandon the system when she still holds a bike, as it must be returned first. That means, we do not consider possible malicious behaviours of users.

It should be noted that the “user rental decision” and the “user return decision” may be repeated during the live cycle of the user in the system. For example, suppose that a user appears and decides (“user rental decision”) to go to a station. If upon arrival the station is empty, then a “user rental decision” has to be made to decide the next action (e.g. try in a different station, wait some time, etc.). Obviously, a user knows about past events (e.g., failed reservations, failed bike rentals, etc.) and, thus, may decide differently in these decision processes in different situations (e.g. do not go to a station already visited to return a bike).

4 Architecture

In this section we describe the architecture of the Bike3S simulator, whose main building blocks are depicted in Fig. 2.

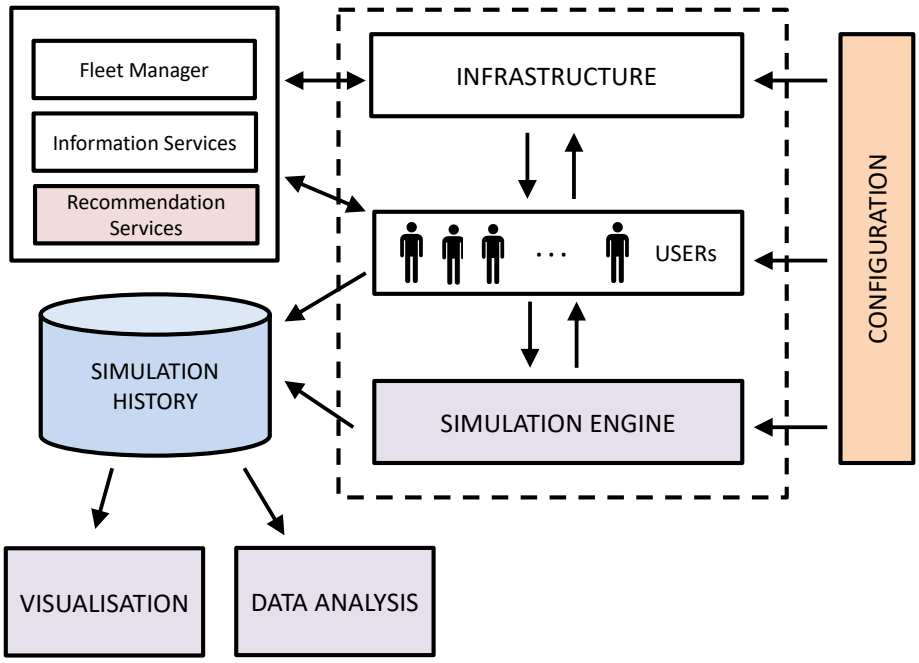

Fig. 2. Simulator architecture.

The core of the simulator consists of the simulation engine that manages the users as well as the bike sharing system infrastructure.

The bike sharing *infrastructure* represents the physical entities of the bike sharing system, i.e. stations and bikes, in the simulator. It contains information about the location of stations, number of available bikes and slots, etc. and of the bikes and slots (their current states). In the current version of the simulator, we do not distinguish among specific bikes and/or slots. However, it would be easy to extend the simulator in the future to account for characteristic like bike usage statistics, maintenance, load level (if electric), etc. The infrastructure is in charge of managing bike rentals and returns as well as bike and slot reservations.

A simulation usually involves several *users*, possibly of different types. During the simulation, users interact with the infrastructure to take/return bikes or make reservations of bikes or slots. In order to take their decisions, users may access external information or recommendation systems. In the figure we represent three kinds of external

services: (i) *fleet manager* refers to the operator of the system who is able to accomplish actions that would modify something in the infrastructure, moving bikes with trucks or changing fares, for example; (ii) *information services* provide (objective) information to users about the current state of the fleet, such as the number of bikes in a station, closest station with available bikes, distance to a station, etc.; and (iii) *recommendation services* provide more processed information, which may include suggestions or requests to go to specific stations because it is better to keep the bike fleet balanced.

The external services can be integrated in the simulator. In particular, it is possible to define and integrate different recommendation services and/or fleet managers with different behaviours.

The initial simulation setup is accomplished through a *Configuration* module that is in charge of specifying global simulation parameters, the initial situation of stations, and the generation of users. During a simulation, the simulation engine stores the necessary information for all the events that have occurred in external files (*Simulation History*). These files can be used later to visualise the simulation in a graphic interface (*Visualization*) or to analyse the performance of simulated system with the *Data Analysis* module.

The general architecture decouples the simulation itself from the *Configuration*, *Visualization* and *Data Analysis* modules. The connection between these modules is accomplished with files in the Json[4] format. This allows for an easy creation of other modules that could be integrated into the architecture.

[4] JavaScript Object Notation (JSON) is a lightweight data-interchange format. https://json.org/

The simulation core (backend) has been implemented in Java. For the user interfaces (frontend) we used Electron and Typescript, while Angular was used for the visualization component.

4.1 Simulation Engine

There are two main approaches to implement agent-based simulators: i) continuous simulation, and ii) event-based simulation. In continuous simulation, a fixed time-step is defined, and the system executes the agents' actions (if any) and updates the state of the environment accordingly at each time step. In many steps there might not be no action nor any changes with respect to the previous state. On the other hand, in event-based simulation, the state of the environment is only updated at the precise time some event occurs. The different simulated entities may fire new events, which are processed by the simulator in sequential order.

In our case, we follow an event-based approach, where the events correspond to the different states of the live cycle of users in the system (as shown in Fig. 1).

Table 1 shows a brief explanation of each event type. Although not detailed there, each event type has different parameters: at least the time instant, and the user involved. Some events have additional parameters (involved station, reservation, …), depending on the event type.

Table 1.Event types.

Event type	Description
User Appears	Represents the user appearance in the system.
User Decides Rental	The user takes the "user rental decision", deciding to abandon, go directly to a station or to try a reservation.
User Arrives at Station (to rent)	The user arrives at a station to rent a bike. It is not sure that the user will be able to take a bike: it depends on if the station has bikes available
User Tries to Reserve Bike	A user tries to make a reservation of a bike at a station. The reservation intent may be successful or unsuccessful. If unsuccessful, the user again takes the "user rental decision". If the reservation was successful, the user has a reservation.
User Has Bike Reservation	If a user has a bike reservation, she walks towards the corresponding station. The reservation may finish with a timeout before the user arrives at the destination station.
Bike Reservation Timeout	A bike reservation has expired before the user arrived at the station. The user will have to take again the "user rental decision".
User Takes Bike	The user takes a bike from the station and performs the "user decision after taking bike". There are two options: either to take a ride or to decide to go to some station to return the bike.
User Finishes Ride	The user has cycled somewhere and now has decided that she wants to go to a station to return the bike.
User Decides Return	This event corresponds to the "user return decision". There are two options: try to reserve a slot at the desired return station or to go directly to the desired return station.
User Arrives at Station (to return bike)	The user arrives at a station to return the bike. As the station may have no available slots, it is not sure the user will be able to return her bike there. In this case, she would have to take again the "user return decision".
User Tries to Reserve Slot	A user tries to reserve a slot at a station. The reservation intent may be successful or unsuccessful. If unsuccessful, the user again takes the "user return decision". If the reservation was successful, the user has a slot reservation.
User Has Slot Reservation	If a user has a slot reservation, she cycles towards the corresponding station. The reservation may finish with a timeout before the user arrives at the destination station.
Slot Reservation Timeout	The slot reservation has finished before the user arrives at the station.
User Returns Bike	The user returns the bike at the station and walks to her final destination.
User Arrives at Destination	The user has reached her destination and leaves the system.
User Leaves System	The user leaves the system.
Fleet manager - Take Bike from Station	A bike is taken from a station (may be issued by the fleet manager).

Fleet manager - Add Bike to Station	A bike is added to a station (may be issued by the fleet manager).
Fleet manager - Check Situation	The fleet manager checks the situation of the infrastructure and may decide to introduce modifications or may issue managing events.

The simulation engine operation is based on a priority queue of events (ordered by the time instant they happen). Initially, for each user to be simulated an event of type "*User Appears"* is created with the time and location indicated in the users' configuration file. Then, the events are processed in order. The execution of events may require some decisions to be made by the user, and usually creates and inserts new events into the queue.

The last three events can be issued by the *fleet operator*. "*Take Bike From Station*" and "*Add Bike To Station*" allows a fleet operator to take bikes from the system (e.g., for repairing), adding new bikes to the system, or moving bikes from one station to another, e.g. to implement rebalancing strategies that are carried out with trucks (as it is done in many real world bike-sharing systems). *"Check Situation"* is an event that passes the control to the implemented fleet manager and allows it to analyse the current state of the system, to introduce modifications and to issue new managing events. When checking the situation, a fleet manager can issue a new *"Check Situation"* event at some point in the future. This allows for a periodic monitorization of the system.

Movements of users in the system are simulated using an external routes server[5]. However, the simulator does not update continuously the position of the users. Instead, a new event would be created with the time at which the user would arrive at its destination. Fig. 3 shows an example, where there are several events that must occur at times 100, 120, 340, 710, 800 and so on. We simplified the information shown in the queue.

[5] We use OpenStreetMap (OSM): https://www.openstreetmap.org

The next event to be processed is *EV(UserAppears, user1,t=100, pos1)* states that *user1* appears at instant t=100 in *pos1*. When this event is processed, *user1* takes the "user rental decision". Let´s suppose she decides to go directly to station 55. After the decision, the route and arrival time are calculated, and the corresponding arrival event is inserted into the queue (*User Arrives at Station (to rent)*). In the example, it takes the user ten minutes (600 sec.) to reach station 55.

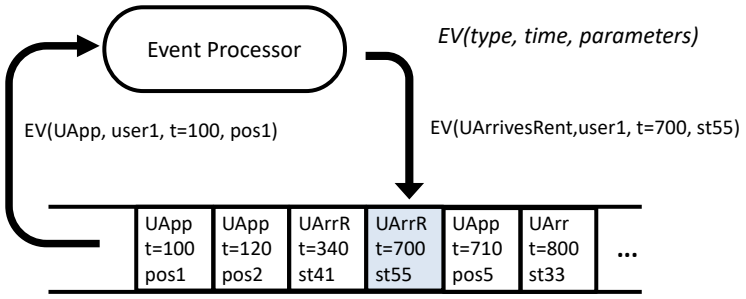

Fig. 3. Event processing example.

In case a user decides to reserve a bike, the event processor calculates if the established maximum reservation time (a system configuration parameter) would expire before the user arrives at the corresponding station. If that was the case, the *Bike Reservation Timeout* event would be inserted into the queue instead of a *Users Takes Bike* event.

4.2 User Management

Users are the principle simulated entities. In fact, a simulation consists of simulating a set of users' behaviours in the system, since the moment they appear until they leave. As mentioned before, the simulation engine controls the flow of user events that may occur by processing the event queue. Most events require some decisions that have to be taken by the corresponding user (e.g., select a station, decide to reserve, …). In real life

different users have different behaviour models and react differently under similar circumstances. For example, some people may not mind walking 500 meters to get a bike while others may only be willing to walk less than 300m. We allow this by using the notion of user types, which represent implementations of different user models.

Each *user type* must provide an implementation of the following user decision methods that the simulation engine may invoke, and which correspond to the different decisions presented in Fig. 1 but contextualised to the situation that has led to the need for a user decision (e.g., a failed bike rental, etc.):

- *decideAfterAppearning*
- *decideAfterFailedRental*
- *decideAfterFailedBikeReservation*
- *decideAfterBikeReservationTimeout*
- *decideAfterGettingBike*
- *decideAfterFailedReturn*
- *decideAfterFinishingRide*
- *decideAfterFailedSlotReservation*
- *decideAfterSlotReservationTimeout*

In practice, we define an abstract type *User* which specifies the abstract decision methods and also provides a basic set of user parameters and default values:

- *position*: indicates the geographical position where a user appears
- *destinationPlace*: the final destination the user wants to go to
- *intermediatePosition*: specifies a geographical location in the region of interest. If this parameter is not null, the user decides to “go for a ride” after getting a

bike. In this case, the user would go with the bike to the specified position. After arriving there, she would decide to return the bike close to the destination place.

- *timeInstant*: the instant (second from simulation start) where the user appears
- *walkingVelocity*: the walking velocity of the user (default 1.4 m/s (Mohler, Thompson, Creem-Regehr, Pick & Warren, 2007; Levine & Norenzayan, 1999))
- *cyclingVelocity*: the cycling velocity of the user (default 4.0 m/s)

Besides these parameters, different user types can specify additional parameters, like the maximum number of rentals (or reservation) a user would try until she abandons the system, or the maximum distance she is willing to walk to find a bike.

An interesting issue is how users choose stations to rent or to return a bike. In particular, a user type may use a specific information or recommendation system for this task. This allows us to test and evaluate different recommendation strategies, e.g., for obtaining a better balancing of the bikes in the global system.

Currently, we have implemented the following user types:

- *Uninformed*. This user tries to take/return bikes just from the nearest station. She does not have information about availability of bikes/slots, so it may happen that there are no bikes/slots available when the user arrives at the station.
- *Informed*. It is an informed user that knows the state of the fleet and always selects the closest station with bikes or the closest station to its destination point that has available slots, when returning a bike. It may happen that the user can not rent a bike when arriving at a station, because other users have taken all available bikes before.

- *Obedient*. It contacts a particular recommender system and always follows its suggestions. This user is adequate for testing different recommendation strategies.
- *Informed-R*. It is like *Informed* but making reservations before going to stations.
- *Obedient-R*. It is like *Obedient* but making reservations before going to stations.

All these user types have additional parameters:

- *minRentalAttempts*: specifies the number of rental attempts the user would try before abandon the system without renting a bike,
- *maxDistanceToRentBike*: specifies the maximum distance a user is willing to walk to rent a bike. For instance, an *Informed* user would abandon if there is no station with an available bike within this distance.

The implementation of new user types in the simulator is fairly easy. Any new user simply has to extend the abstract *User* class. Furthermore, we use reflection[6] to specify the user types of a particular simulation. Thus, no additional changes in the simulator are necessary.

[6] Java reflection allows to explore object characteristics (e.g. class name, methods, instances of a class, annotations …) and to perform operations (invoke methods, create instances, etc.) at runtime. In our case, using reflection, it is possible to implement different user, recommendation system and fleet manager types and to load them into the simulator without modifying any other source code.

4.3 Experiment Configuration

Simulation experiments are configured by providing three types of configuration parameters: Stations, Users and Global, as shown in Fig. 4. These configuration parameters are stored in *json* files.

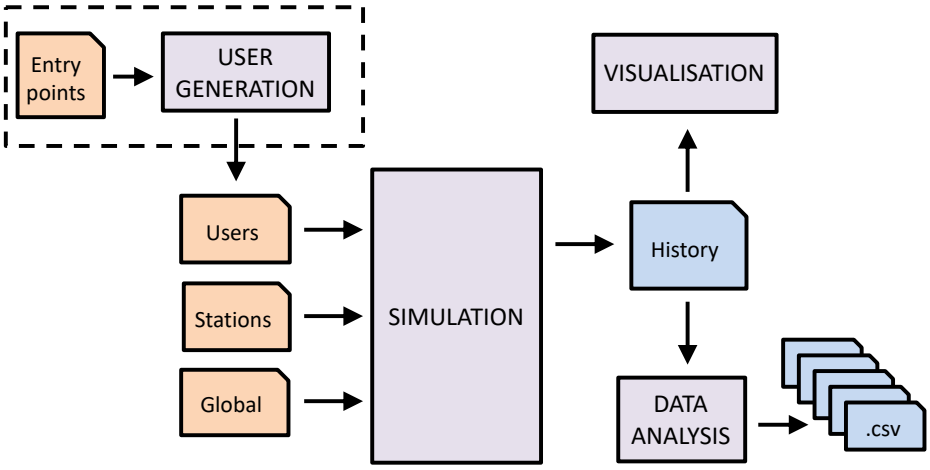

Fig. 4. Experiment configuration and generated files

Stations.

It contains information about the stations in the system, including physical location, capacity and initial number of bikes.

Global.

Specifies the global parameters of a simulation experiment. They include the following fields:

- *reservation time*: the time before a reservation expires
- *total simulation time*: the total duration of the simulation (in seconds)

- *random seed*: (optional) it can be used to generate the same sequence of random values.
- *bounding box*: of the underlying area of interest (geographical coordinates of top-left and bottom-right corners)
- *debug mode*: true or false, specifies whether or not debug information is generated
- *start date time*: (optional) a reference date and time for which the simulation is taking place. This information is useful if the balancing strategy uses expected demand.
- *graph manager type:* specifies the file graph manager used for calculating routes. Currently we only implement the use of OSM to calculate routes and times. It contains a *typeName* and a set of *parameters*.
- *recommendation system type*: (optional) is the type of recommendation system used by users that make use of such a system. It contains a *typeName* and a set of *parameters*.
- *fleet manager type*: (optional) is the repositioning system used in the simulation. It contains a *typeName* and a set of *parameters*.
- *demand manager type*: (optional) is the type of expected demand management system available for the recommendation and fleet manager systems. It contains a *typeName* and a set of *parameters*.

With regard to the *recommendation system type* and *fleet manager type*, similar to user types, different recommendation systems and fleet managers can be implemented. Again, we use reflection to select the particular implementation of a recommendation system and fleet manager in the simulator, based on the value of the field

typeName in the configuration file. Parameters of a particular implementation are specified in the *parameters* set.

Each recommendation system has to extend the abstract class *RecommendationSystem* that implements some auxiliary methods and defines the abstract methods: *recommendStationToRentBike* and *recommendStationToReturnBike*. Each implementation of a recommendation system has to implement these two methods that should return an ordered list of stations which are recommended to a specific user (at a given time and position) for renting/returning a bike. Each fleet manager implementation has to extend the abstract class *FleetManager* and has to implement the methods *initialActions* and *checkSituation*. The first method is called right at the beginning of any simulation and allows a fleet manager to setup initial managing events in the event queue, for example, to monitor the system at certain time intervals. The second method is called from any "*Fleet manager - Check Situation"* event and represents the specific implementation of a monitoring activity.

Users.

Specifies the list of all users that are generated during the simulation. Each user is specified through the parameters described in section 4.2 (*position, destinationPlace*, *intermediatePosition, timeInstant, walkingVelocity* and *cyclingVelocity*), as well as its *userType* (which corresponds to an implementation of a particular user model) and any other parameters that are specific to the particular *userType* implementation.

All three configuration files can be generated manually or via a user interface (see Fig. 5). With respect to the Users configuration, we also implemented a *User Generator* tool for generating users randomly by specifying a set of *Entry Points*. Each en-

try point either represents a single user or a “user generator”, allowing to generate random user trips between an origin and destination area. In the latter case, users are generated with random values for the fields: *position, destinationPlace*, *timeInstant*, *walkingVelocity* and *cyclingVelocity*. The *timeInstant* is generated from a starting time with a Poisson distribution with a parameter λ (where λ represents appearance rate in users/h). The Poisson distribution is a discrete probability distribution that represents events occurring at time intervals and independent of the time of the last event. It has been used by others to model user arrivals at a bike sharing system, e.g. (Chemla et al. 2013). With regard to the *position* and *destinationPlace* fields, each of those values is generated randomly within a circle with a specified radius from a defined origin location and destination location, respectively (both radius may be different). In this way and using different entry points, it is possible to generate (randomized) user trips that correspond to a typical origin/destination matrix, where origin and destination are areas or fixed locations. The user generator creates a *User* configuration file, which can then be used as input to the simulator.

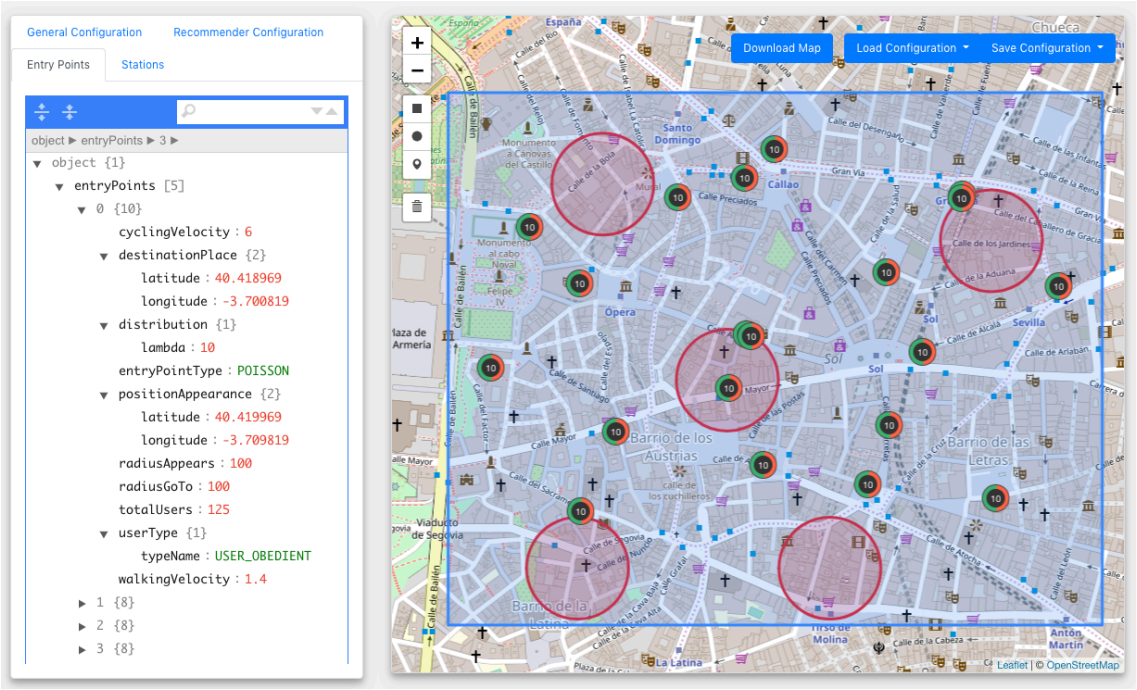

Fig. 5. Snapshot of the configuration interface. Big red circles represent entry points, with their origin location. Small circles are stations, with the number of available bikes in them. The ratio of available bikes and free slots is shown in red and green, respectively. The left-hand side shows the details of one entry point.

4.4 Visualisation

The aim of the visualisation module is to display the produced history data of the simulation in a graphical and appealing way (see a snapshot in Fig. 6). This includes rendering geographic data on a map and presenting the current state of the system at each moment.

The visualization essentially acts as a reproducer of a set of recorded data where the entities are displayed on a map with additional status information. In particular, the situations of all stations and all users that are currently in the system is presented. The reproduction is done in a playback mode simulating real-time. Internally, the visualiza-

tion module processes the data of the events that have taken place (as stored in the history files) and updates the situation of the entities in simulated real-time. In contrast to the simulation engine, here, the movements of the users (either when walking or when cycling) are simulated, that is the position of the users is continuously updated. The standard visualisation speed is “real-time” but the speed can be increased or decreased (+/- icons). It is also possible to rewind the playback (e.g., with a negative speed factor).

To render the map and to simulate movements, data from OpenStreetMap is used. The actual entities (users and stations) are displayed by interactive markers that provide additional information of the current state of each entity. As we mentioned before, our simulation tool is event based and, thus, does not track the actual movements of users (just the events of starting or ending a movement). However, the visualisation module does show the movements of users by translating the route data (available in the simulation history) to positions at each particular instant.

Fig. 6. Snapshot of the visualization interface. Bike and person symbols represent users riding or walking, respectively. Numbers on top of the symbols are identifiers.

4.5 Data analysis

In this section we describe the quality metrics our simulator generates to evaluate the performance of a BSS. More extended explanations can be found in (Fernández et al., 2018).

We use the following notations to define the metrics: *N* is the total number of users, (i.e. total bike demand), *Successful hires* (*SH)* is the total number of bike rentals, *Failed hires* (*FH)* represents the total number of attempts to hire a bike that failed due to unavailability, *Successful returns* (*SR*) and *Failed returns* (*FR*) represent the same numbers for bike returns. Based on these measures, we calculate the following metrics:

- *Demand satisfaction* (*DS*): ratio of users who were able to hire a bike (either at first trial or not), including those who booked a bike in advance. This is one of the most common metrics used to evaluates BSS (e.g. (Chemla et al., 2013))

$$DS = SH / N$$

With regard to bike returns, we do not define a similar measure since we assume that all users that have taken a bike must return it to some station.

- *Hire efficiency* (*HE*): ratio between the number of rentals and the total rental attempts of those users who hired a bike (FH_h).

$$HE = SH / (SH+FH_h)$$

- *Return efficiency* (*RE*): ratio between the number of returns and the total return attempts:

$$RE = SR / (SH+FR)$$

- *Average Empty Time (AET)*: is the average time a station is empty, which means it is potentially denying a service. This metric is useful when users have access to information about bike availability before making their decision (in real life, e.g. via an app). In those cases, they may go to the nearest station with available bikes.
- *Average Deviation (AD) with regards to a balanced situation.* Here a station is considered balanced if the number of available bikes is half its capacity (thus keeping half empty slots). For each station the average deviation is calculated as the average absolute error. This metric is the average of all stations. *AD* might be a useful metric for assessing balancing strategies that try to keep each station at half occupancy. This assumes, by default, that the optimal inventory level is half the station's capacity. Clearly, this may be suboptimal, as the optimal inventory level depends on the demand context (i.e. unbalance demand level) and on the repositioning strategies used. For example, a nearly full station may be "balanced" if much more requests than returns are expected in the next operating period.
- *Users' time in the system*. It is the time users spend in the system. The total time (*TT*) is the sum of walking time to origin station (T_{os}), cycling travel time to return station (T_{rs}) and walking time to final user destination (T_{fd}):

$$TT = T_{os} + T_{rs} + T_{fd}$$

Note that T_{os} and T_{rs} include walking or cycling (respectively) from station to station if no available bike/slot is found.

All these measures are generated by the data analyser and are written to files in *csv* format so they can be imported into more powerful data analysis tools for further analysis.

5 Evaluation

In the sequel we present, first, a validation of the Bike3S simulation tool where we analysed a real case scenario. Afterwards, we present three examples to show the possibilities of our simulation tool for evaluating management decisions or strategies, especially balancing strategies: a simulation of "designed scenarios" in the city of Madrid and simulations with real data in the cities of Madrid and London. However, in this paper, our research is not focused on adequate balancing strategies. We just implemented a set of rather simple strategies as examples.

5.1 Real-case validation

In order to evaluate the correct operation of the Bike2S simulator, we tried to replicate the operation of BiciMAD, the public bike sharing system in Madrid (Spain), for several different days with real data. This system covers an area of about 5x5 km square of central Madrid and counts currently on 174 active stations and about 1800 bikes (at September 2019). The capacity of the stations is between 12 and 30, but most stations have 24 slots. There are publicly available data of the usage of the BiciMAD system, in particular:

- Data of the trips including, for each trip, time of taking a bike, origin station, destination station, travel time and the approximate route. However, in order to anonymise the data, only the day and hour of the pick-up time of each trip are

given (without minutes). Each trip includes a user type, with possible values representing regular or occasional users, BiciMad staff or unidentifiable users.

- Situation of the stations: including the number of available bikes and slots and whether or not a station was active. We recompiled this information during July 2018 from the official BiciMAD API at an interval of 5 minutes.

In order to replicate a real-world scenario, we extracted the user data for a 24-hour period (in particular, from 7:00 on the 20th of July 2018 to 7:00 on the 21st of July 2018). There have been 12296 real user trips (corresponding to regular and occasional users) that started in this interval and 849 movements of bikes with trucks (trips between different stations carried out by BiciMAD staff). We used the user trip data to generate the simulated users for our simulator. Given the data of a real user trip, we neither know from the data at which position a real user decided to go to a station nor where her final destination is. Therefore, and in order to reflect the real situation as good as possible, we used the coordinates of the station on which a user unplugged /plugged a bike as the appearance and destination locations of the simulated user. The walking velocity is set to a default value of 1.4 m/s and the cycling velocity is obtained from the distance between the origin /destination stations and the user travel time. Finally, the instant of appearance of the simulated user is generated randomly within the hour of the appearance given in the real data (with a uniform distribution), since we do not know the exact minute or even second of appearance. We used the *Uninformed* user type that would just go to the closest station to get a bike (in this case the station where they appeared) and would leave the system if there were no bikes available at this station (they do not retry at the same or another station). Regarding bike returning, the users would go directly to the station closest to their destination and if there was no slot available, they would go to the next closest station to try there.

In order to replicate the movement of bikes with trucks, we implemented a simple fleet manager that issues the corresponding *"Take Bike from Station*" and *"Add Bike to Station"* events for the 849 real movements. Also here, we do not know the exact time of a movement (only the hour in which it takes place). In this case we specified a time 30 minutes after the initial corresponding hour.

For specifying the initial station configuration in the simulated scenario, we used the official (real) data at the initial time (20th of July 2018 at 7:00). There have been 169 active stations with 1796 bikes.

We analysed the performance of the simulated system in two ways: a) without bike movements with trucks, and b) with the 849 bike movements with trucks. Table 2 summarizes the results.

Table 2. Experiments results for Madrid.

UserType	# abandoned	DS	RE	TT (min)	real TT (min)
Without truck movements	496	0.959	0.886	15.63	14.75
With truck movements	162	0.987	0.993	14.78	14.75

As it can be seen in the table, the simulation results do not perfectly match the real-world scenario. In the case where the bike movements with trucks are also simulated (the case closer to reality), there are 162 users (about 1.3%) that did not find a bike at the station where they appeared (and the users abandoned the system). This gives a demand satisfaction (DS) of 0.987. Regarding the returns, there have been 80 failed return attempts (0.7%; return efficiency (RE) 0.993); e.g., in 80 cases a user did not find an empty slot and had to go to another station to try to return the bike. Finally, the average time (TT) employed in the simulated user trips is 14.78 minutes where it has been 14.75 minutes in the "real world" (real TT). We measure this time only for the users

that did not abandon. Since in this case we respect the real-world cycling velocities in the simulation, this small difference is only due to the users that could not return their bikes at the expected stations and had to spend extra time to cycle to another station.

The difference between the simulated and the real scenario is clearly due to the fact that with the available data, we do not know the exact appearance time of users (just the hour) and thus, establish the appearance time randomly within the corresponding hour. In an oversized system (e.g., with always enough available bikes and slots at each station) the approach we took to randomize the appearance time of users should not have any influence in the results. However, in the case of the BiciMAD system, it is quite common that a station becomes empty or totally occupied. Analysing the real occupation situation of the stations for the simulated period and at an interval of 5 minutes, it turns out that 4,8 % of the 48503 different measures (every 5 minutes during 24 hours for 169 stations) a station is empty and at 1% it is fully occupied. In such a situation, even very small differences in the appearance times of users will lead to bike rental or return failures. In this sense, we consider that the abandon rate of 1.3% and the return fail rate of 0.7% are due to these differences and rather confirm the correct functioning of the simulator.

Also, the differences in the results when including/not including the bike movements with trucks provide evidence for evaluation. A clear increase in rental failure and much higher in return failure can be observed if we omit the movements of bikes from one station to another through trucks. This clearly shows that such operational actions in general do have a positive effect on balancing the availability of bikes (and slots) in bike-sharing systems.

5.2 Designed scenarios

With this set of experiments, we show the possibility to create and evaluate artificial scenarios. We chose a 3km square map of central Madrid and set 20 stations in real locations of the current bike-sharing system BiciMAD. We gave them a capacity of 20 bikes and, initially, each station was set up with 10 available bikes (thus, also with 10 empty slots). We set five entry points, whose locations are shown in Fig. 5. At each entry point the appearance radius is set to 200m, i.e. users generated by an entry point appear at a random location within a distance of 200m from that entry point. Users' destinations are randomly chosen in the whole area. We set a maximum of two failed attempts to rent or reserve a bike before leaving the system without using it and a maximum distance of 600 meters a user would be willing to walk in order to find a bike. Walking and cycling velocity was set to 1.4 and 6 meters per second, respectively.

With this set of parameters, we analyse the performance of the system with different user types for increasing demand data (increasing number of users). We carried out experiments where the generation ratio of users at each entry point was 10, 20, 40, 60, 80, 120 and 150 users per hour. We evaluated the following types of users (presented in section 4.2): *Uninformed, Informed, Informed-R* and three types of *Obedient* users: i) *Obedient-AvR*, where users use a recommendation system that recommends the station with the most available resources (bikes or slots, respectively), ii) *Obedient-AvR/Dist* where the station with the best ratio between the available resources at that station and the distance of the user to the station is recommended, and iii) *Obedient-AvR-R,* like *Obedient-AvR* but users always reserve bikes or slots at the recommended stations. In all cases, the recommendation of a station is limited to stations within 600 meters of the user position when asking for a station to rent a bike and within 600 meters from the destination location when users ask for a station to leave the bike.

Fig. 7 to Fig. 12 show the evolution of the performance of the different user types with increasing demand and for the different metrics defined in section 4.5. As expected, the performance with *Uninformed* users is the worst of all cases. *Obedient* users outperform *Informed* users in the efficiency measures since the use of a balancing strategy reduces the number of times that stations get empty. However, *Obedient-AvR* decreases significantly in demand satisfaction and hire efficiency with higher demands. In fact, it behaves even worse than the simple *Informed* strategy. This is because many users that appear at almost the same moment and in similar regions will be sent to the same best station and will find that there is no bike left when they arrive (only the firsts will get them). Possible improvements to that strategy could be recommending different stations to different users (e.g. with a probability proportional to the number of available resources) or taking into account that the users who received recommendations will likely follow them, so the "virtual" number of available resources is different. Return efficiency is very high in almost all cases. This is due to the fact that users' destinations are not concentrated in specific locations as with appearances as well as that some users abandoned the system without hiring. Furthermore, *Obedient-AvR* present quite bad total time values which is due to the fact that users are send to stations that are rather far away. Both problems are mitigated with the *Obedient-AvR/Dist* type. With regard to reservations, it can be observed that users that make reservations have generally a higher chance to get a bike.

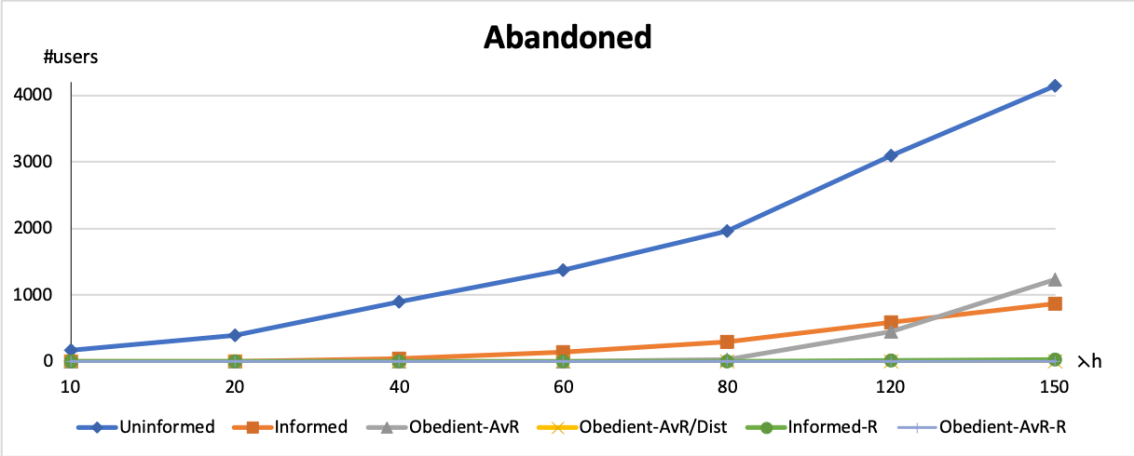

Fig. 7. Abandoned execution results. λh is the appearance rate per entry point in users per hour. This metric is equivalent to Demand Satisfaction but is easier to see. Informed-R coincides with Obedient-AvR-R.

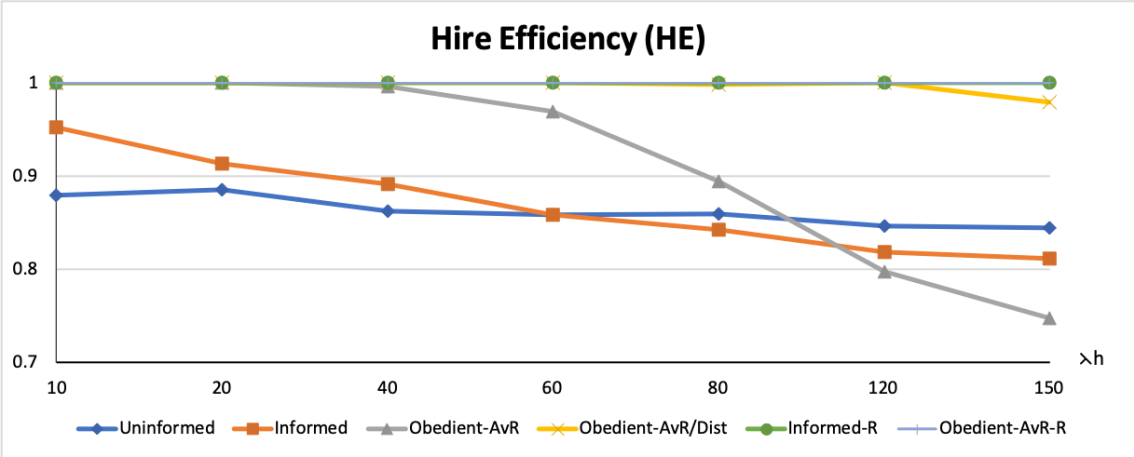

Fig. 8. Hire Efficiency execution results. Informed-R and Obedient-AvR-R coincide.

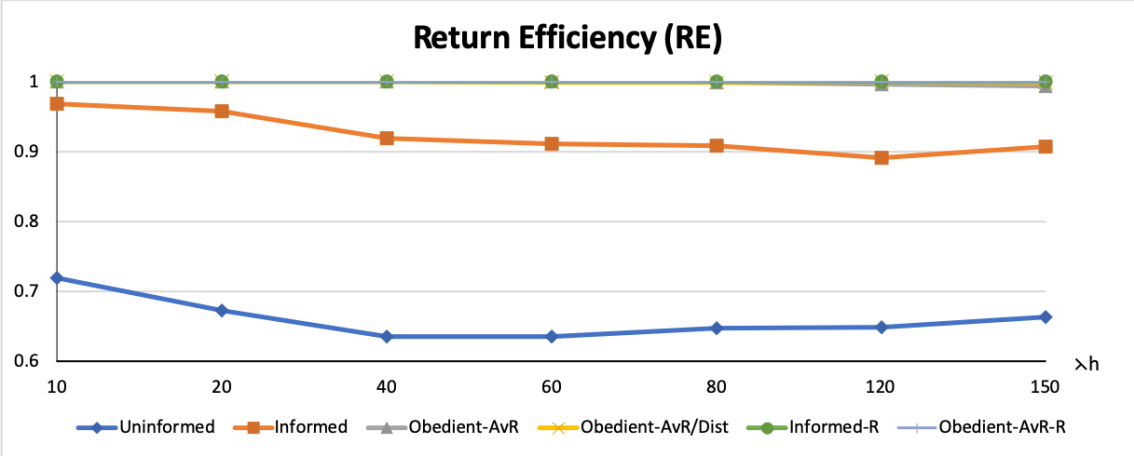

Fig. 9. Return Efficiency execution results. Informed-R and Obedient-AvR-R coincide.

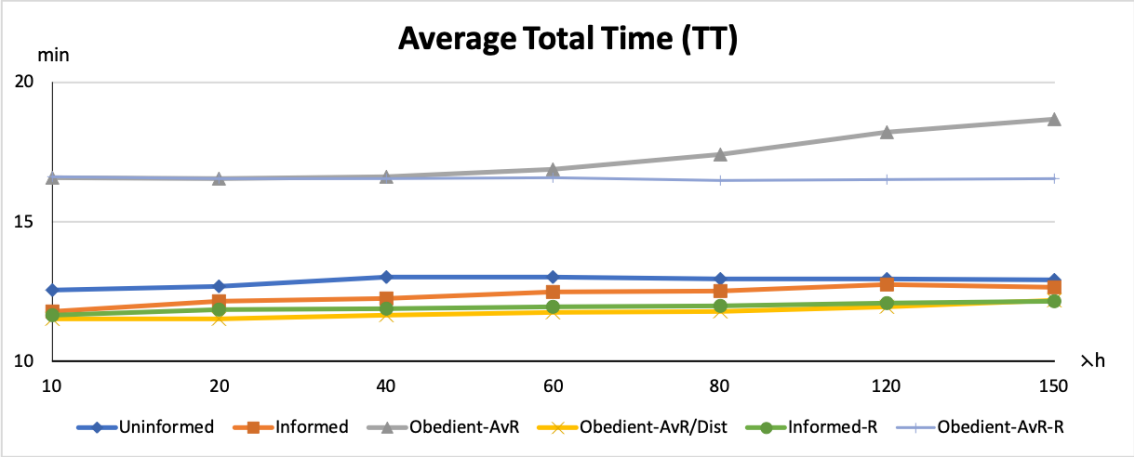

Fig. 10. Total Time execution results.

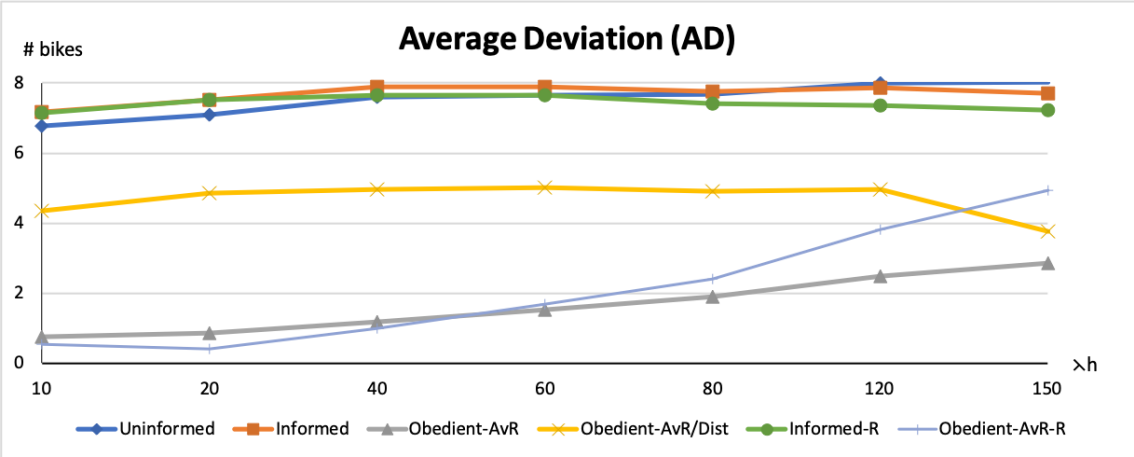

Fig. 11. Average Deviation execution results.

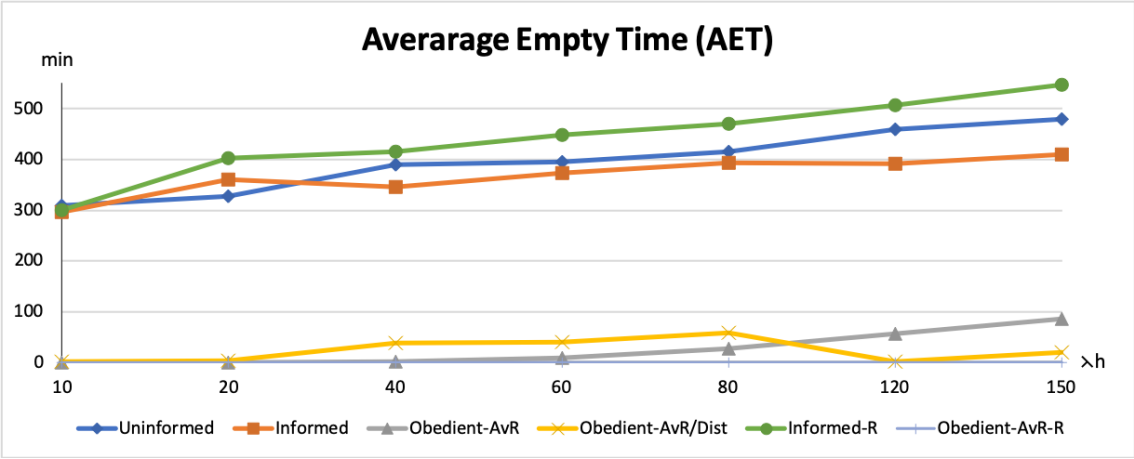

Fig. 12. Average Empty Time execution results.

5.3 Real scenario simulation for evaluating balancing strategies. Madrid and London.

In this set of experiments, we used real data from Madrid and London. The goal is to present scenarios with different real infrastructures (number, location and size of stations) and use patterns.

In the first experiments, we used again data from BiciMAD. In particular, in this case, we used the data of the 25th of June 2018, a day with a fairly high usage of the system with 13104 user trips. In order to simulate the trips in a more realistic way we

randomly generated appearance time in minutes within the specified hour (with a uniform distribution). In addition, we also randomly generated the appearance and destination location of users with a uniform distribution within a radius of 200 meters from the real origin and destination station, respectively. All the stations (173 active stations at this day) were initialised with a ratio bikes/empty slots of 0.5. We set a maximum of three failed attempts to rent or reserve a bike before a user would abandon the system without using it. Furthermore, we specified a maximum radius of 600 meters as the distance a user is willing to walk in order to find a bike. If she does not find a bike at this distance, she will also abandon the system. The walking velocity of all users has been set to 1.4 meters per second and the cycling velocity has been obtained from the information of the real route information.

We analysed the performance of the following user types (as explained in section 4.2): *Uninformed*, *Informed*, *Obedient/AvR*. and *Obedient-AvR/Dist*.

Table 2 shows the results of the experiments. The efficiency data (*DS*, *HE* and *RE*) are very high in general, especially for users that decide based on recommendations. This is normal because the historical data used in the simulation only contain successful hires. Despite this fact, those values are not always 1 since we had to randomly distribute users within each hour and close location as explained before, so the data are not exactly the same as the historical records. Uninformed users get the lowest performance in demand satisfaction (*DS*) and efficiency (*HE* and *RE*). That is due to the fact that users go to their nearest station without even checking whether there are bikes available. The other user behaviours present a better performance. If users follow the recommendation strategy to go always to the station with the most resources, the results present the best values in average empty time and deviation. The reason is that these

metrics analyse station status and this strategy tends to keep individual stations as balanced as possible. However, the performance from the users' perspective, i.e. the time users spend in the system, is much worse. If the recommendation is done by taking into account both, the distance a user would have to walk to a station as well as the available resources, then a reasonable balance of bikes at station is obtained, and users have a fairly low total time in the system.

Table 3. Experiments results for Madrid. Bold numbers indicate the best obtained result for each metric.

UserType	# abandoned	DS	HE	RE	TT (min)	AD	AET (min)
Uninformed	200	0.98	0.93	0.80	14.6	5.3	83.2
Informed	83	0.99	0.98	0.94	14.0	5.4	91.5
Obedient-AvR	**0**	**1.00**	**1.00**	**1.00**	21.7	**1.9**	**0.4**
Obedient-AvR/Dist	18	1.00	1.00	0.99	**13.8**	4.3	28.6

The experiments using real data from London cover an area of about 20x10 km square. The system includes 783 stations and 20178 bikes. The capacity of each station varies from 10 to 62 slots. We used the data of the 30th of May 2018, which included 31736 trips. The rest of parameters and decisions are the same as in the previous experiment. The results obtained (Table 4) are in line with the ones from Madrid.

Table 4. Experiments results for London. Bold numbers indicate the best obtained result for each metric.

UserType	# abandoned	DS	HE	RE	TT (min)	AD	AET (min)
Uninformed	488	0.98	0.98	0.83	22.4	6.3	12.0
Informed	60	1.00	0.98	0.93	**22.2**	6.6	16.3
Obedient-AvR	**0**	**1.00**	**1.00**	**1.00**	30.4	**3.6**	**0.0**
Obedient-AvR/Dist	**0**	**1.00**	1.00	0.99	22.4	5.7	1.2

6 Conclusion

In this paper we have described Bike3S, a station-based bike sharing system simulator. The objective of Bike3S is to analyse the behaviour of a BSS given an infrastructure (stations and bikes) and expected demand in different points of the area of interest. In addition, the simulator can be used to evaluate balancing strategies. Bike3S is highly customisable to account for different user behaviours, infrastructure and experiment configurations. We presented a validation of the tool with real data from the BSS BiciMad in Madrid as well as several use cases to show the type of experiments that can be carried out and the results that can be obtained.

The modular design of Bike3S allowed us to separate the configuration and user generation from the simulation execution, and the latter from the visualisation and analysis. Thus, the simulator can generate users or load them from a file. Likewise, the visualisation interface or data analysis tool can load previously stored simulation histories. One of the main characteristics of Bike3S is its capability to be easily extended with new user types and balancing strategies.

We are currently working on designing incentive-based balancing strategies, which are evaluated using Bike3S. Some preliminary works can be found in (Fernández et al., 2018).

In the future, we plan to extend the simulator in several lines. One is including fares to hire bikes into the infrastructure, so that discount-based incentives or dynamic pricing balancing strategies can be implemented. Another extension includes considering electric bicycles, which implies managing battery load levels, discharge/load times, decisions on which bikes to hire, etc.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness, and the Spanish Ministry of Science, Innovation and Universities, co-funded by EU FEDER Funds, through grants TIN2015-65515-C4-4-R (*SURF*) and RTI2018-095390-B-C33 (InEDGEMobility).

References

Caggiani, L., & Ottomanelli, M. (2012). A modular soft computing based method for vehicles repositioning in bike-sharing systems, *Procedia - Social and Behavioral Sciences, 54*, 675-684.

Chemla, D., Meunier, F., Pradeau, T., Calvo, R. W., Yahiaoui, H. (2013). Self-service bike sharing systems: Simulation, repositioning, pricing. https://hal.archives-ouvertes.fr/hal-00824078.

Contardo, C., Morency, C. & Rousseau, L. M. (2012). Balancing a dynamic public bike-sharing system. *Technical Report CIRRELT*, vol. 4.

Dubernet, T. & Axhausen, K.W. (2014). A Multiagent Simulation Framework for Evaluating Bike Redistribution Systems in Bike Sharing Schemes. *Arbeitsberichte Verkehrs- und Raumplanung* 1010. ETH Zurich.

Erdoğan, G., Battarra, M., Calvo, R.W. (2015). An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research* 245(3), 667–679.

Fernández, A., Billhardt, H., Timón, S., Ruiz, C., Sánchez, O. & Bernabé, I. (2018). Balancing Strategies for Bike Sharing Systems. In: Lujak M. (eds) *Agreement Technologies (AT 2018)*. LNCS, vol 11327. pp. 208–222. Springer, Cham

Fellendorf, M. (1994). Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority. In: *64th Institute of Transportation Engineers Annual Meeting* (pp. 1– 9). Springer.

Forma I. A., Raviv, T. & Tzur, M. (2015). A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transportation Research Part B: Methodological* 71, 230–47.

Fricker, C. & Gast, N. (2012). Incentives and regulations in bike-sharing systems with stations of finite capacity. arXiv preprint arXiv:12011178.

Gámez-Pérez, K., Arroyo, P. & Puente-Rivera, E. (2019). System Dynamics Simulation to Explore the Impact of a Bike Sharing System. Evidence from Mexico. Ecocience. International Journal 1(1), 60-72.

Haider, Z., Nikolaev, A., Kang, J. E. & Kwon, C. (2018). Inventory rebalancing through pricing in public bike sharing systems, *European Journal of Operational Research* 270(1), 103-117.

Horni, A., Nagel, K. & Axhausen, K. (Ed.). (2016). *Multi-Agent Transport Simulation MATSim*. Ubiquity Press, London.

Ji, S., Cherry, C. R., Han, L. D. & Jordan, D. A. (2014). Electric bike sharing: simulation of user demand and system availability. *Journal of Cleaner Production* 85, 250-257

Jian, N., Freund, D., Wiberg, H. M. & Henderson, S. G. (2016) Simulation optimization for a large-scale bike-sharing system. *Winter Simulation Conference (WSC)* pp. 602-613.

Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4), 128–138.

Levine, R. V. & Norenzayan, A. (1999). The Pace of Life in 31 Countries. *Journal of Cross-Cultural Psychology*. 30 (2), 178–205.

Lin Y-K. & Liang, F. (2017). Simulation for Balancing Bike-Sharing Systems. *International Journal of Modeling and Optimization* 7(1), 24–27.

Mohler, B. J., Thompson, W. B., Creem-Regehr, S. H., Pick, H. L. Jr. & Warren W. H. Jr. (2007). Visual flow influences gait transition speed and preferred walking speed. *Experimental Brain Research*, 181 (2), 221–228.

O'Mahony, E. & Shmoys, D. B. (2015). Data analysis and optimization for (citi)bike sharing. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)* (pp. 687–694). AAAI Press.

Pal, A., Zhang, Y. (2017). Free-floating bike sharing: Solving real-life large-scale static rebalancing problems. *Transportation Research Part C: Emerging Technologies* 80, 92–116.

Pfrommer, J., Warrington, J., Schildbach & G., Morari, M. (2014). Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems* 15(4), 1567–1578.

Romero, J. P., Moura, J. L., Ibeas, A. & Alonso, B. (2015). A simulation tool for bicycle sharing systems in multimodal networks. *Transportation Planning and Technology*, 38(6), 646-663.

Saltzman, R. M. & Bradford, R. M. (2016). Simulating a More Efficient Bike Sharing System. *Journal of Supply Chain and Operations Management* 14(2), 36–47.

Schuijbroek, J., Hampshire, R. C. & Van Hoeve, W. J. (2017). Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research* 257(3), 992–1004.

Soriguera, F. & Jiménez-Meroño, E. (2020). A continuous approximation model for the optimal design of public bike-sharing systems. *Sustainable Cities and Society* 52 101826

Soriguera, F., Casado, V. & Jiménez, E. (2018). A simulation model for public bike-sharing systems. In *CIT2018. Proceedings of the XIII Conference on Transport Engineering*. Transportation Research Procedia 33. 139–146.

Waserhole, A., & Jost, V. (2016). Pricing in vehicle sharing systems: optimization in queuing networks with product forms. *EURO J. Transportation and Logistics, 5*, 293-320.