A Bike Sharing System Simulator

Alberto Fernández[1], Sandra Timón[1], Carlos Ruiz[1], Tao Cumplido[2],

Holger Billhardt[1], Jürgen Dunkel[2]

[1]CETINIA, University Rey Juan Carlos, Madrid, Spain
{alberto.fernandez, sandra.timon,
carlos.ruiz,holger.billhardt}@urjc.es
[2] Computer Science department, Hochschule Hannover, Germany
taocumplido@gmail.com,juergen.dunkel@hs-hannover.de

Abstract. Bike-sharing systems are becoming very popular in big cities. They provide a cheap and green mean of transportation used for commuting and leisure. Being a shared limited resource, it is common to reach imbalanced situations where some stations have either no bikes or only empty slots, thus decreasing the performance of the system. To solve such situations, trucks are typically used to move bikes among stations in order to reach a more homogeneous distribution. Recently, research works are focusing on a complementary action to reduce imbalances consisting in incentivizing users to take (or return) bikes from stations with many bikes rather than those with few bikes, e.g. by fare discounts. In this paper, we present simulator for analyzing bike-sharing systems. Several user generation distributions can be configured. The simulator is specifically designed with the aim of evaluating incentive-based rebalancing strategies. The paper describes in detail the characteristics and potential of the simulator, including several experiments.

Keywords: bike sharing, simulator, smart transportation

1 Introduction

Nowadays, due to the great pollution produced by motor vehicles and since not everyone can afford to acquire and maintain one, bike-sharing systems have emerged in big cities to try to palliate these problems. These systems allow citizens to move between different geographical points in a simple and economical way, as they can rent a bike at any station and return it at a different one. Also, some of these bike-sharing systems offer the possibility of making bike or slot reservations, which is even more convenient for the users. However, resources in bike-sharing systems, i. e. stations and, consequently, bikes and slots, are limited. Therefore, it is important to manage these resources as efficiently as possible so as to provide a good quality of service and user experience. To this aim, it is necessary that each station has the appropriate number of bikes and slots at each moment, so that a user who arrives at a station to rent a bike (or

wants to reserve a bike) can find available bikes, or the one who arrives at a station to return a bike (or wants to reserve a slot) finds an available slot.

Traditionally, bike balancing has been done by trucks that transport bikes from some stations to others. Some research has focused on optimizing the static balancing problem [1–3], where the routes of trucks at night or off-peak periods are optimized. More recently, the dynamic problem has been considered, which involves predicting the demand on each station in the next period and optimizing the distribution of bikes in stations so as to maximize the number of trips (i.e. reduce the number of "no-service" situations) [4–6]. While those approaches only consider trucks as the means to rebalance the bike-sharing system, there are other works that try to incentivize bike users to collaborate in the system rebalancing [7–10], typically by dynamically modifying rental prices. Using prices as incentives is also common in static strategies, for example BiciMAD[1] introduces discounts on final rental price if the user hires a bike at a station with more available bikes or if he returns it to a station with many empty slots.

While most current systems are station-based, the development of location technologies (e.g. GPS) have provoked the appearance of several floating systems [1, 12]. In such systems, bikes can be left "floating" anywhere in the city so the problem of finding an empty slot is avoided. In those systems, the problem is to keep the different areas of the city covered with available bikes.

In this paper we present a simulator that allows analyzing the behavior of station-based bike-sharing systems. The ultimate goal of the simulator is to help to evaluate dynamic rebalancing algorithms based on user incentives in different settings. In this sense, different user types (behaviors) can be defined and different balancing strategies can be implemented. Although some simulation models exist [11], to the best of our knowledge, there are no available simulators of this kind. Chemla et al. [7] presented an open-source simulator for evaluating their proposed algorithms for dynamic balancing but, unfortunately, the simulator is no longer available.

The rest of the paper is organized as follows. In section 2 we introduce the kind of systems this simulator is aimed at. Section 3 details the architecture and its components. Some experiments are presented in section 4. We conclude the paper in section 5.

2 Context and Problem Description

In this paper we envision a system that dynamically provides incentives to persuade users to help in the balancing of the distributions of bikes in the city. We concentrate on station-based systems, e.g., bikes can only be taken or left at fixed stations in the city and each station has a fixed number of slots to hold available bikes. Furthermore, we consider that users can make reservations on stations, either of a bike or of a slot to leave a bike. The global architecture of the envisioned system is presented in Fig. 1.

There is a *physical infrastructure* consisting on bikes and stations, which a fixed number of slots. The information about the current situation of the stations and bikes is received by the fleet operator, which can use that information to provide more or less

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

dynamic management of the stations. For example, different incentives could be given to users to try and keep the system balanced. Such incentives might be fixed (e.g. in Madrid a discount is given if a bike is taken from a station with more than 70% of occupancy) or could be modified dynamically. Through smartphones or web applications users may access recommendation services, such as station information, booking bikes/slots, routes searches between stations, etc. Those recommendation services, created by the system operator or third party providers, make use of the dynamic management of stations (e.g. state or prices), so they could recommend/incentivize booking a bike in a station with some discount.

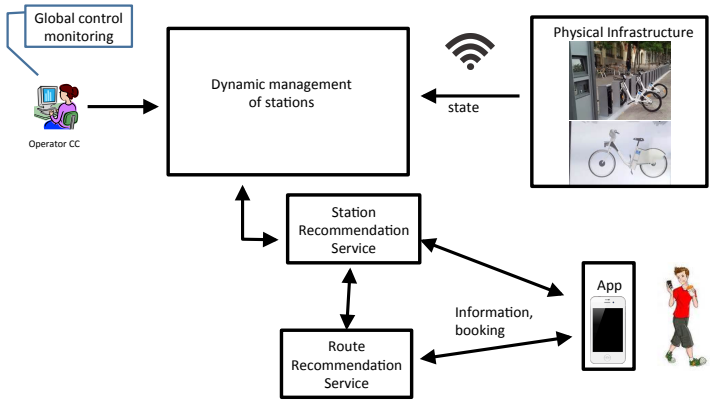

Fig. 1. Bike Sharing System.

The dynamic management of stations along with recommendation services are fundamental to try and keep the system as efficient as possible, i.e. reduce or mitigate imbalances. In order to design and compare different dynamic balancing strategies, a bike sharing simulator is required.

The aim of this paper is to present a simulator we have designed as general as possible in order to evaluate different balancing strategies and also allowing the definition of different types of users with different behaviors.

In the simulator, a user can appear in the system at any point in the simulated area (we will see more details in section 3.3), which includes stations. The first thing a user has to decide is which station he wants to take a bike from and whether he wants to reserve it before moving there. That decision is on the user behavior model and might be taken using support applications (information, recommendation, etc.). Once the user arrives at the chosen station, it may happen that there are no available bikes (even if there were some when the user consulted the information). In that case, the user must decide whether to choose a new station (and possible reservation) or to leave the system without rental. The same situation may also happen even for users having bookings since the expiration time could occur before arriving at the station.

In case that the user successfully hired a bike, the destination is decided, which can be a particular station, or the user just uses the bike to "go around". In the latter case the user eventually decides to return the bike at any point in the city (this allows for

simulating users that rent bikes for fun without knowing the destination at rental time). In case the user knows the target station, he must decide whether to make a reservation of a slot or not. Now, the situation is very similar to rental: there might be no empty slots when the user arrives at a station. In this case, the user does not have the option of leaving the system. Instead has to search for a new station to leave the bike.

3 Architecture

In this section we describe the architecture of our simulator, which is depicted in Fig. 2. The core of the simulator comprises (i) a bike sharing system infrastructure, (ii) user management and (iii) an event-based simulator engine.

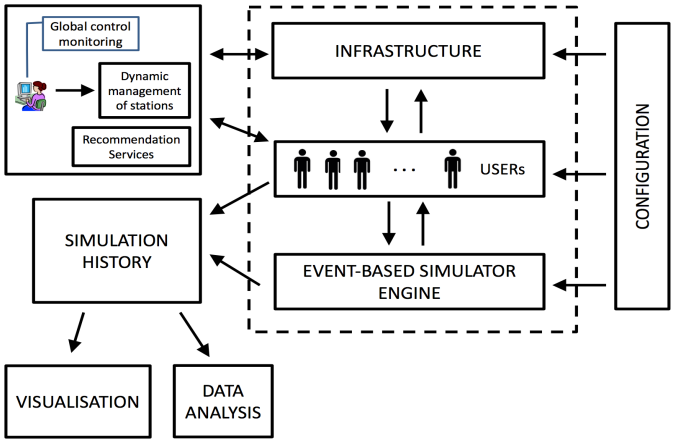

Fig. 2. Simulator architecture.

The bike sharing *infrastructure* represents the stations and bikes in the simulator. It contains information about the location of stations, number of empty bikes and slots, etc. and manages the rentals and bookings of bikes and slots. The *user management* includes different types of users and manages the interaction of them with the infrastructure as well as the information and recommendation services provided by the bike fleet operator. The *event-based simulator engine* is the main component of the core. It is in charge of processing the different events that occur in the system (new users, rentals, returns, bookings, etc.), triggering new events when necessary.

A *configuration* module is in charge of setting the parameters of a specific simulation, such as the infrastructure used, types and probability distribution of users, simulation time, geographical area to simulate, etc. During the simulation, users, depending on their behavior, can use *operator's services* to make decisions such as which station to take a bike from. In that module is where balancing algorithms can be implemented.

The simulation track is stored in a *simulation history*, which can be used later for graphical visualization and/or data analysis. This architecture allows us to decouple the simulation from its visualization and analysis. When the simulation is run the results

are stored. The *Visualization* component shows the evolution of the simulation on a map. It can load a different simulation from a file, so this is a functionality that can be used without executing any simulation. The same is applied to *data analysis:* the statistics of any pre-recorded simulation can be analyzed.

In the following we describe in more detail the main building blocks of the simulator.

3.1 Simulator Engine

There are two main approaches to implement simulators. In continuous simulation, a fixed time-step is defined and the system is updated every step. In many steps there might not be any changes with respect to the previous one. On the other hand, in discrete or event-based simulation, the system is only updated when a new event occurs.

Our simulator follows an event-based approach, since the state of the system only changes when new events occur (see Table 1). However, the visualization component uses a continue simulation approach (see section 3.5).

Table 1 shows a brief explanation of each event type. Although not detailed there, each event type has different parameters, among others the time in which the event takes place.

Table 1. Event types.

Event type	Description
User Appears	It represents the user appearance and, consequently, the moment when he determines the station where he wants to rent a bike from and whether he wants to make a bike reservation.
User Arrives at Station to Rent Bike Without Reservation	It is not sure that the user will be able to take a bike: it depends on if the station has bikes available
User Arrives at Station to Rent Bike With Reservation	It is sure the user will be able to take a bike from the station.
Bike Reservation Timeout	A bike reservation has expired before the user arrived at the destination station while the reservation was active (valid)
User Wants to Return Bike	The user has cycled somewhere in the city and now has decided he wants to go to a station to return the bike
User Arrives at Station to Return Bike Without Reservation	As the station may not have available slots, it is not sure the user will be able to return his bike there
User Arrives at Station to Return Bike With Reservation	A user is returning a bike because he had a reservation of an empty slot
Slot Reservation Timeout	It means the slot reservation has expired before the user arrived at the destination station

The simulation engine operation is based on a priority queue of events (ordered by the time instant they happen). Initially, all the "*User Appears"* events for the whole simulation are introduced (they are generated at the beginning of the simulation or loaded from a file). Then, the events are processed in order. The execution of events may require some decisions to be made by the user model associated to the event, and it usually creates and inserts new events into the queue.

For example, event *User Appears(user1,100)* states that *user1* appears at instant t=100. When that event is processed, *user1* must decide (her user model is invoked) the station she wants to rent a bike from and whether she wants to make a reservation. Then, the route and arrival time are calculated[2], and the corresponding arrival event is inserted into the queue (*User Arrives at Station to Rent Bike With/Without Reservation*). In addition, if the user decided to book a bike, the *Bike Reservation Timeout* is also added with the established maximum reservation time (it is a system configuration parameter).

3.2 User Management

Users are responsible of making decisions on different moments of the simulation in which they are invoked by the simulator engine when processing the events. Our simulator easily allows using different types of users in the same simulation.

Each kind of user has his own implementation for the decision methods, which gives them their particular behavior, usually defined by a set of configuration parameters. There are different types of decisions to be made, for example, leaving the system (after a bike reservation timeout, after arriving at a station without available bikes...), making bike or slot reservations, deciding the destination station after renting it or to cycle somewhere in the city before deciding where to return it, etc.

An interesting issue is how users choose stations to hire or return bikes. They can use a recommendation system that provides information about fleet state as well as incentives to keep the fleet as balanced as possible. Users process that information to make decisions based on their behavior model. For example, one user might not want to go to a station located further than 500m whatever discount he gets, while other might have a more complex decision function (e.g. combining distance and incentives).

Currently, our simulator provides the following types of users:

- *Uninformed*. This user tries to take/return bikes from the nearest station. He does not have information about availability of bikes/slots, so it may happen that there is no availability when the user arrives at the station.
- *Informed*. It is an informed user that knows the state of the fleet and only goes to stations with available bikes/slots. He chooses target stations by the shortest distance.
- *Obedient*. It contacts the recommender system and always follows its suggestions. Therefore, he is the perfect user from the dynamic balancing system's point of view.
- *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.
- *Random*. This user randomly makes most of his decisions.
- *Commuter*. He always goes directly from the origin to the destination station to arrive as soon as possible at work, study, etc.
- *Tourist*. His main characteristic is that, after renting a bike, he always cycles somewhere in the city before deciding where to return it.

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

- *Distance/Resources ratio*. He chooses lowest ratio of the distance to the station and the number of available bikes or slots it has.
- *Most available resources*. This user always goes to the station with more available bikes (for rentals) or empty slots (for returns).

3.3 Experiment Configuration

Simulation experiments are configured by providing three types of configuration parameters: infrastructure, user generation and global configuration. These configuration parameters are stored in *json* files. We are currently developing a user interface to facilitate its creation.

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

User generation. It specifies where and when users appear in the system, as well as their type. User generation is defined by *entry points*. An entry point is a user generator defined by a location in the map (longitude and latitude coordinates), radius, probability distribution (e.g. Poisson(λ=5)), type of users, number of users (0 for no limit) and starting time. Users are generated following the indicated distribution in a random position within the circle specified by the parameter *radius* around the center *location*. Each generated user is randomly assigned a walking and cycle speed in a predefined range. Entry points provide a flexible way of configuring different types of scenarios. For example, if we know demands at stations, we can simulate it by setting entry points at each station's location (and radius = 0) and demand distribution. Different types of users can be generated at the same point by defining several entry points at the same location.

Generated users (with *type*, *position* and *time*) are stored in a file that is fed into the simulator engine. This way, we can load the same users in several experiments, for example, to evaluate different station configurations, balancing strategies, etc.

Global configuration. Global parameters of the experiments are specified in this part. They include the maximum *reservation time*, total *simulation time*, random *seed* (optional, it can be used to generate the same sequence of random values), *bounding box* of the map (top-left and bottom-right corners), *map* (we use OSM to calculate routes and times) and *output path* for the history file. We consider time units in seconds.

3.4 Simulation History

During the execution of the simulation, a history file is generated. It includes all the information about the events that occur in the simulation, such as new users, arrivals at stations, taking/returning bikes, routes taken by the users, etc. The information is represented incrementally, i.e. at each event, the change in the system is recorded.

Simulation history is used for visualization (section 3.5), which allows reproducing the result of the simulation graphically as many times as desired, without needing to execute the simulation again. This is very convenient since it separates simulation and visualization, so they do not need to be synchronized. Actually, they usually run at a different pace. For example, a simulation of one day may take a few seconds, but its visualization might last for several minutes. Furthermore, the user might want to visualize different cases already stored without executing the simulation again.

History is also used for data analysis (section 3.6), where different quality measures are taken to assess the efficiency of the bike sharing system. Again, the separation of simulator engine and analysis brings several advantages. It allows executing complex simulations that may take a long time, and once finished (offline), their data can be analyzed quickly. Since executions are stored in files, they can be easily processed and their results, compared.

Furthermore, simulation histories could be used for developing demand forecast methods, e.g. histories of execution could be used as training data to machine learning algorithms.

3.5 Visualization

The aim of the visualization component is to display the produced history data of the simulation in a useful and appealing way (see Fig. 3). This includes rendering geographic data on a map and making the status of the current situation visible. Additionally, the visualization is designed and implemented in an entity-agnostic way, resulting in a pluggable architecture that allows to easily add concrete entity models. This works well with the data format used in the simulation history (section 3.4).

The visualization essentially acts as a reproducer of a set of recorded data where certain entities are displayed on a map with additional status information. The reproduction can occur in two ways:

1. A step-by-step reproduction of the data. This simply takes the next or previous recorded timestamp and applies the changes.
2. A playback mode simulating real-time. This mode waits in simulated real-time for the next or previous event to occur but updates the position of moving entities in between. The speed in which real-time is simulated can be controlled. A negative factor will rewind the playback.

To render the map, data from OpenStreetMap is used. The actual entities are displayed by an interactive marker, where the actual image and interactivity can be configured for each entity individually. It is also possible to not display an entity at all. In the bike-rental simulation these are currently the reservations and bikes. In the simulation a bicycle is either parked at a station or is rented by a user to cycle around. Both cases do not require actually displaying the bike, rather the user marker should reflect whether a user is currently a pedestrian or a cyclist. The plug-in interface allows to easily adapt to new scenarios though. For example, it would be no problem to add a display strategy for bikes that are not parked at stations but at any point in the city, as long as this information is present in the history data.

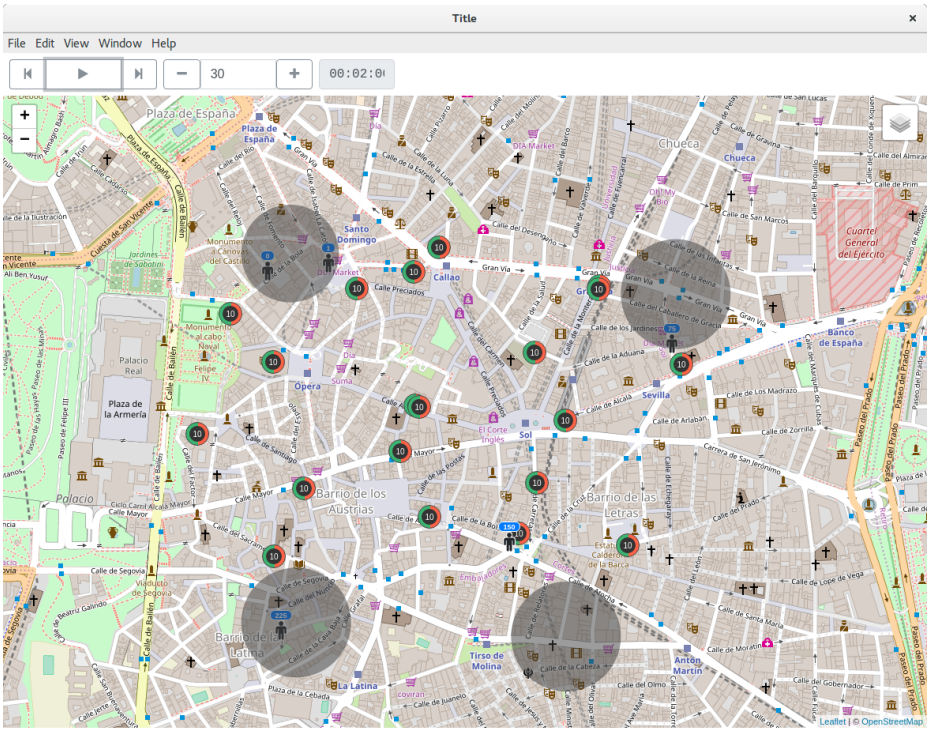

Fig. 3. Snapshot of the visualization interface. Entry points locations and radius are indicated in grey circles Small circles represent stations, with the number of available bikes shown in them. The ratio of available bikes and free slots is shown in red and green, respectively. Bike and person symbols represent users riding or walking, respectively.

3.6 Data Analysis

Since the main purpose of this simulator is to assess algorithms that try to balance the number of available resources (bikes/slots) at a bike-sharing system, it provides quality measures that allow to evaluate the efficiency of those algorithms. The basic measures that provides our simulator are the following, which are stored per station and globally: *Successful hires* (*SH*, total number of bike rentals), *Failed hires* (*FH*, total number of attempts to hire a bike that failed due to unavailability), *Successful returns* (*SR*), *Failed returns* (*FR*), *Successful bike booking* (*SB*), *Failed bike bookings* (*FB*), *Successful slot booking* (*SS*), and *Failed slot bookings* (*FS*).

With the global values, we define the following quality measures (*N* is the total number of users, i.e. total bike demand):

- *Demand satisfaction* (*DS*): measures the ratio of users who were able to hire a bike (either at first trial or not), including those who booked a bike in advance.
$$DS = SH / N$$
- *Return satisfaction*(*RS*): measures the ratio of users who were able to return their bikes (either at first trial or not), including those who booked a dock in advance.

Note that return demand is SH (those who actually hired a bike). The result should be 1, otherwise there are users who did not find an empty slot.

$$RS = SR / SH$$

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

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

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

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

All these measures are written in a *csv* format file so it can be imported into a more powerful statistical software for further analysis.

4 Evaluation

We have carried out several experiments to evaluate the simulator. We chose a 3km square map of central Madrid and set 20 stations in real locations of the current bike-sharing system of Madrid (BiciMAD). They all have a capacity of 20 bikes. Initially, each station was set to 10 available bikes (thus, 10 empty slots too). We set four entry points. The location of those elements is shown in Fig. 3. We set a maximum of two failed attempts to rent or book a bike before leaving the system without using it.

In order to test the effect of a balancing strategy, we created a simple recommendation system that returns the stations within a range of 600m to the user's location, sorted by the ratio of available bikes or slots, depending on whether the user wants to hire or return a bike, respectively. The goal of this paper is not proposing a good strategy, we only created one to show the potential of the simulator.

We carried out several experiments with the following types of users (presented in section 3.2): *Uninformed*, *Informed*, *Obedient*, *Informed-R* and *Obedient-R*.

In order to compare the influence of the balancing strategy, we executed configurations with users of only one type in each simulation, as well as a combination of *Informed* and *Obedient* (50% of each). In order to have more precise comparison, users were generated at the same location and time for all configurations, i.e. users were generated once and stored in a file, which was loaded in every experiment.

Fig. 4 shows the results of demand satisfaction (DS) for different number of users. As expected, fleet performance with *Uninformed* users is the worst of all cases. Tests with *Obedient* users outperform *Informed* since the use of a balancing strategy reduces the number of times that stations get empty. When users make reservations they get bikes at stations more frequently up to the fleet becomes highly saturated (3000 users), where *Obedient-R* starts declining. The reason is that stations assigned by the recommender are farther than usual so they keep bikes occupied (used+reserved) longer times thus other users cannot take them. However, *Informed-R* does not suffer that effect since they always go to the closer station. Hire efficiency (HE) is shown in Fig. 5.. The results are in line with DS. As expected, users that make reservations always hire bikes.

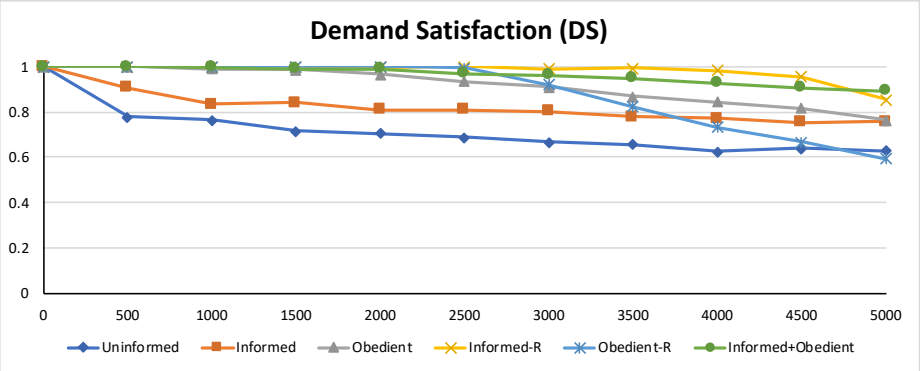

Fig. 4. Demand Satisfaction execution results.

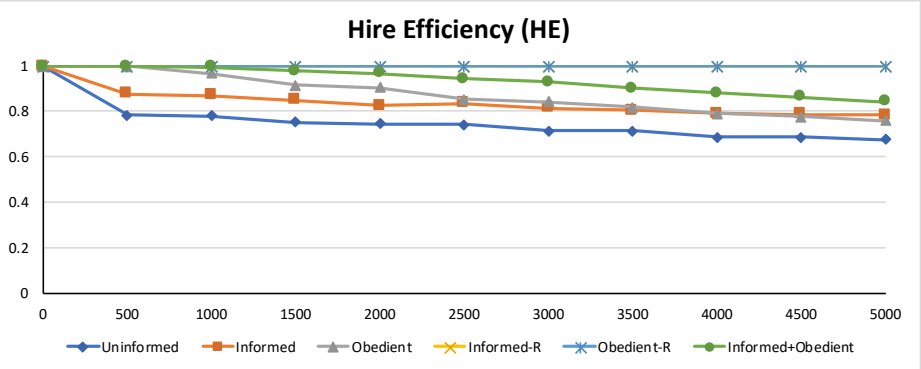

Fig. 5. Hire Efficiency execution results. *Informed-R* coincides with *Obedient-R*

5 Conclusion

In this paper we have presented a bike sharing system simulator. We described its architecture in detail and carried out some experiments so as to show the type of results that can be obtained. Stations of different size and locations can be configured, users with different behaviors can be randomly generated everywhere in a map following different probability distributions, simulation results can be stored, retrieved, graphically visualized and quality measures are taken for its analysis.

The architecture design allowed us to separate the configuration from the simulation execution, and the latter from the visualization and analysis. Thus, the simulator can generate users or load them from a file. Likewise, the visualization interface or data analysis tool can load previously stored simulation histories.

The simulator can be used with different objectives. On one hand, it can be used to assess a specific bike-sharing system infrastructure (station locations, size, etc.) before deploying it by testing how the proposed infrastructure behaves to a given expected

demand. On the other hand, different incentive based strategies can be implemented and evaluated.

Currently we are working on creating graphical interfaces to assist users to create configurations. Other future lines related to the simulator include adding further quality measures (e.g. time a station is empty, extra distance walking by users, etc.). In addition, we plan to work on designing incentive based algorithms and test them with the simulator.

Acknowledgments. Work partially supported by the Autonomous Region of Madrid (grant "MOSI-AGIL-CM" (S2013/ICE-3019) co-funded by EU Structural Funds FSE and FEDER), project "SURF" (TIN2015-65515-C4-4-R (MINECO /FEDER)) funded by the Spanish Ministry of Economy and Competitiveness, and through the Excellence Research Group GES2ME (Ref. 30VCPIGI05) co-funded by URJC-Santander Bank.

References

1. Pal, A., Zhang, Y.: Free-floating bike sharing: Solving real-life large-scale static rebalancing problems. Transportation Research Part C: Emerging Technologies 80, 92–116 (2017).
2. Erdoğan, G., Battarra, M., Calvo, R.W.:An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. European Journal of Operational Research 245(3), 667–679(2015).
3. Forma I. A., Raviv, T., Tzur, M.: A 3-step math heuristic for the static repositioning problem in bike-sharing systems. Transportation Research Part B: Methodological71, 230–47 (2015).
4. Contardo, C., Morency, C., Rousseau, L. M.: Balancing a dynamic public bike-sharing system.Technical Reportvol. 4. CIRRELT.(2012).
5. O'Mahony, E.,Shmoys, D. B.: 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 (2015).
6. Schuijbroek, J., Hampshire, R. C., Van Hoeve, W. J.: Inventory rebalancing and vehicle routing in bike sharing systems. European Journal of Operational Research 257(3), 992–1004 (2017).
7. Chemla, D., Meunier, F., Pradeau, T., Calvo, R. W., Yahiaoui, H.: Self-service bike sharing systems: Simulation, repositioning, pricing. https://hal.archives-ouvertes.fr/hal-00824078. (2013).
8. Fricker, C., Gast, N.: Incentives and regulations in bike-sharing systems with stations of finite capacity. arXiv preprint arXiv:12011178 (2012).
9. Pfrommer, J., Warrington, J., Schildbach, G., Morari, M.: Dynamic vehicle redistribution and online price incentives in shared mobility systems. IEEE Transactions on Intelligent Transportation Systems 15(4), 1567–1578 (2014).
10. Waserhole, A., Jost, V.: Pricing in Vehicle Sharing Systems: Optimization in queuing networks with product forms OSP. 2012. <hal-00751744v5> (2014).
11. Romero, J. P., Moura, J. L., Ibeas, A., Alonso, B.: A simulation tool for bicycle sharing systems in multimodal networks. Transportation Planning and Technology 38(6), 646-663 (2015).
12. Reiss, S., Bogenberger, K.: Optimal bike fleet management by smart relocation methods: Combining an operator-based with an user-based relocation strategy. IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 2613–2618 (2016).