Prediction of queue waiting times

Oriol Prat Font

April 23, 2023

Abstract—Queuing theory is a branch of mathematics that studies, analyses and predicts the behaviour of queues using queuing models. Even though queuing models were originated in the early 1900s, they are not widely used in modern times. This project uses the Poisson distribution together with exponential distribution and other probability techniques to implement a queuing model that can predict the behaviour of queues and estimate the expected customer waiting times and their lengths. The project also involves the generation of plots for representing and analysing the data. The system can be monitored with dashboards that provide real-time data about the queues and alerts that are set off when a queue exceeds a previously defined limit.
Python is used to implement the queuing model, and the simulations are made in the discrete-event simulation framework SimPy, that is based on Python. The plots are also generated in Python, using the library Matplotlib. On the other hand, the dashboards used to monitor the system are created in PowerBI.

Index Terms— behaviour of a queue, discrete event, exponential distribution, Matplotlib, monitorization, M/M/1, Poisson, Python, SimPy

1 Introduction

AMUSEMENT parks are popular destinations for people seeking entertainment and adventure. These parks offer a wide range of attractions, from breathtaking roller-coasters to family rides. However, there's something that frustrates everyone: long queues. Queues can lead to a negative experience of the park and can cause discomfort to the visitors.

This project focuses on a big amusement park that has this exact problem. The understanding of the behaviour of its queues is needed in order to find alternatives that can provide the fast services visitors demand. Understanding how queues work and being able to predict them can optimize processes and increase the visitor's satisfaction.

In the project the principal objective is to accurately predict waiting times. In order to do so, a queuing model is implemented using Poisson and exponential distribution along with other probability techniques. Additionally, plots and charts are designed and implemented to easily analyse the queue. Furthermore, by monitoring the system using dashboards the behaviour of the queue can be understood and, therefore, different options for improving the queuing system can be contemplated.

- *Contact e-mail: Oriol.PratF@autonoma.cat*
- *Project tutored by Antonio Espinosa Morales (Departament d'Arquitectura de Computadors i Sistemes Operatius)*
- *Course 2022/23*

2 Objectives

After defining and analyzing the problem to be solved, a series of objectives are outlined to successfully conclude this project:

- Accurately predict queue waiting times.
- Analyze the queues using plots.
- Monitor the system with dashboards.
- Propose alternatives that can reduce waiting times or make queues more enjoyable.

The proposal to achieve them consists of a plan that follows several steps.

The plan begins with the implementation of a queuing model that can predict waiting times so that it's possible to analyze and understand the behavior of the queue. The model used is a variation of the M/M/1 and it is programmed in Python [1]. This model simulates a queue with Poisson arrivals, but it has been modified to have constant service times. The reason for this modification is that in amusement parks the time of service is the ride time, which doesn't change depending on the server or the number of people in queue.

This model is designed in Python using SimPy [2]. This discrete-event simulation library is really useful for simulating real-life events in Python. It can easily model active components such as customers and servers, which are

required in this project. There are a lot of projects that use this library, some of them being related to queueing theory and queueing systems [3]. The system created in the project using SimPy can control queues, managing the flow of customers based on the behavior of each queue and their real-time status. Once the model is implemented, we are able to identify areas of improvement in the queue, this allows us to reduce wait times and, therefore, optimize processes.

The next step is to create plots and charts using the output data from the model. This is also implemented in Python using the library Matplotlib, which provides a wide range of visualization options and customizations while being easy to use and supporting other libraries [4]. Most of these plots use queuing and service times, but they are also useful to visualize the number of people in queue at any time. All these visualizations are fundamental in order to better comprehend and easily analyze the queues.

Additionally, the system can be monitored through dashboards created with PowerBI [5]. This dashboard consists of a first page where the length of each attraction at that moment is displayed. This value is shown using a color gradient that goes from green to red depending on the length of each queue. It is also possible to obtain more information from any attraction by clicking on it in the previous page. This leads to another page in the dashboard where KPIs such as queue lengths and wait-times are displayed in order to better understand the status of each queue. Furthermore, notifications and alerts are also sent to staff when queue lengths exceed a limit previously defined.

3 State of the art

At present, companies from all kinds of sectors have already designed queuing models and queue management systems. Their objective is to optimize their businesses, and queuing theory can contribute to it because it can provide shorter waiting times, that lead to a higher customer satisfaction.

Queuing theory has been studied throughout the last few years, which has contributed to the creation of diverse technologies used to optimize queues. These first technologies date back to the 1970s, where physical and electronic methods such as barriers and tickets were first developed. Later on, big industries like healthcare and retail began using these technologies, which lead to an improvement in queuing management systems by implementing real-time data and customer feedback tools.

Today, queue management systems are used in mobile apps, virtual queues, and even contactless payments. They are also starting to implement machine learning algorithms to predict customer behaviour and future long waiting times.

In recent years, there has been significant research on queueing models in amusement parks, with the aim of improving queue management and enhancing visitor experience. This specially includes models such as M/M/1 and M/M/k, to predict queue performance based on different factors, like ride capacity, arrival rates, and service times. Research has also focused on the use of simulation and analytical models to evaluate the effectiveness of different strategies for improving queue performance, such as single-rider lines, virtual queues, and express passes [6]. Machine learning and artificial intelligence are also emerging as a promising approach in the entertainment industry.

However, there are still challenges in applying queuing models to amusement parks, such as the dynamic nature of the park environment, the complexity of its queue behaviour, and the variability of demand. Therefore, more research in this field is required to further develop and optimise queueing models in this sector, with the ultimate goal of giving visitors a better experience and increasing the park's income.

As new technologies emerge, queuing theory will become more and more valuable. One technology that can benefit from queuing models and simulations is self-driving vehicles. With autonomous cars becoming more and more common, queuing models can be used to optimize traffic flow and minimize waits during congestions. Additionally, going further, in a future world where all vehicles are autonomous, queuing models could completely get rid of congestions.

4 Methodology and planning

The methodology used to complete this project is based in the Scrum framework. As only one person is actively involved in the project, this framework has been adapted to suit the needs [7].

In the project, the Scrum responsibilities of product owner, scrum master and development team are slightly modified so that they can all be assigned to a single person.

The agile methodology used is based in sprints. In the longer tasks there have been more than one sprint and they have been divided on smaller parts of the task. The sprints have lasted between one week and two, depending on the complexity of each task. In addition, there have been weekly meetings with the project tutor. The tasks planning is shown in table 1.

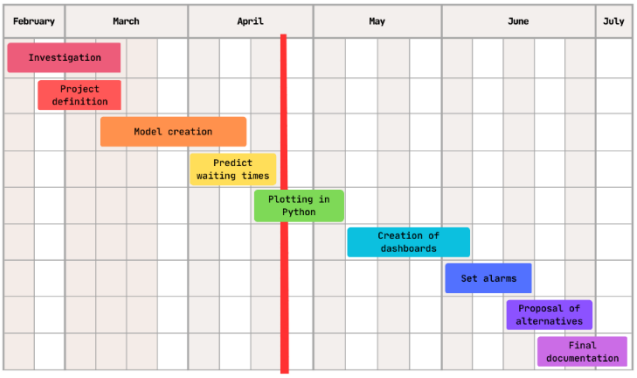

TABLE 1: Planning of the project

As there's only one actively involved person in this project, no planning, tracking or control tools have been used. On the other hand, Microsoft Teams [8] and Outlook have been used as communication tools with the project tutor, although there have also been in-person meetings.

5 Development

As mentioned previously, the proposal to achieve the objectives is based in a series of steps that must be carried out in a specific way, following a systematic order. Some of these tasks have already been completed and will be presented in the following sections while the next steps are described later on in this document.

5.1 Creation of the model

The first thing needed to start implementing the model was the data, so a dataset containing the data of 50 attractions from the amusement park was created. This data refers to the number of people who ride each of the attractions every hour, being the columns the opening hours in the park's schedule.

The model has been created in Python using the simulation framework SimPy. Python is an easy-to-use programming language that offers a wide range of libraries, such as Pandas [9] for data manipulation and NumPy [10] for numerical computing. These have been used in the model together with the previously mentioned library Matplotlib.

This model consists of a Queue Simulation class that receives three variables: the number of users, the arrival rate and the service rate. This class has a function called 'run()' that loops over each of the users and processes them in the simulated queue.

The simulated queue follows the same process as a real queue. First of all, the arrival time of the user is generated using Poisson distribution and the arrival rate defined in the class. The user then waits for the simulation to get to this 'Arrival time'. Once the user is in line it will wait until the server is available, with the server being unavailable referring to it attending another user in that moment. When the server is available and the user is the next in queue, it will be attended. This instance is defined as the 'Queue end time', which will also be the 'Service start time'. The service time is computed based on the service rate initially defined in the class. This service time is designed to have almost identical values through all the iterations because the time of service in a park is the ride time, which is always the same. The user will stay there until the service time is completed, which will be the 'Departure time'. At that moment the user will depart and the next user in line will start the service. A flowchart of these simulated queues is shown in image 1.

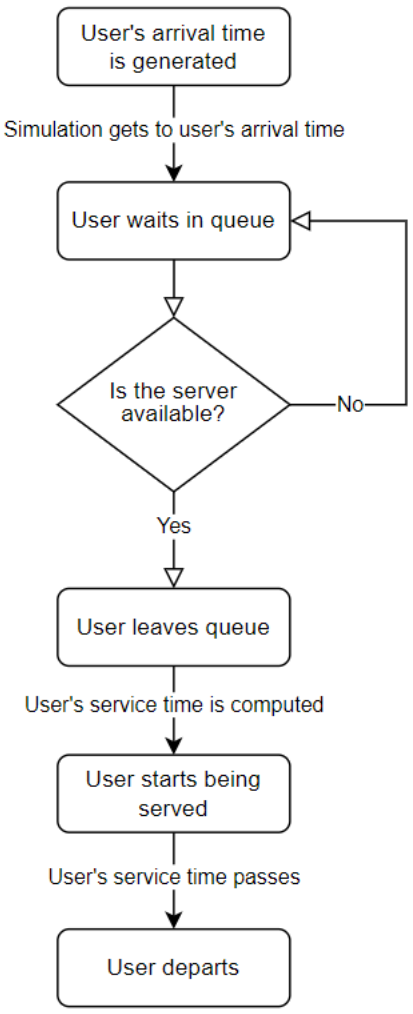

IMAGE 1: Simulated queue's flowchart

Once a user departs, its simulation times are saved in a Pandas dataframe where the rows refer to each of the users in the simulation. Later on, the 'Time in queue' and 'Service time' variables are computed using the following formulas:

$$Time\ in\ queue = Queue\ end\ time - Arrival\ time$$

$$Service\ time = Departure\ time - Service\ start\ time$$

These two variables are also saved in the dataframe with the rest of the simulation times. When all the users in the queue have been served the simulation ends and the Queue Simulation class returns this dataframe.

It's important to acknowledge that, in order to get closer to a real queuing system, the rides are considered to carry 8 users at the same time. Therefore, every time the server is available, the first 8 users in queue will start being attended at the same moment and will also depart when the service is completed at the same time.

5.2 Prediction of wait times

Once the model was implemented, the data was extracted from the dataset and loaded in a dataframe using Pandas. Each of the values in the dataframe refer to the number of people who get in a specific attraction in a specific hour from the park's opening schedule. These values are used in the Queue Simulation class as the number of users. The parameters of arrival rate and service rate are defined as the same value because if the arrival rate is higher than the service rate the number of people in queue would be constantly growing and vice versa.

When the simulation of all the users from one attraction in one hour is completed and the 'run()' function returns the dataframe of simulation times, the average of all the 'Time in queue' times is computed. This value will be the average waiting time for a specific attraction in a specific hour and it is stored in a new dataframe. The average waiting times for all attractions and times are computed and stored in this dataframe as soon as each simulation is completed.

As a result, when all the simulations are done the script exports the original dataset but with the values of the average waiting times instead of the number of people, which were the values in the initial data. Additionally, the source code prints the average wait time of all the values in the exported dataframe in order to get a better understanding of how the queue has performed in the simulations.

5.3 Plotting

When all the simulations are completed and the wait times have been exported, the plotting part of the code begins. In this section a Gantt chart and a histogram of wait times are generated using Python's Matplotlib library [11].

A Gantt chart for one of the simulations is shown in image 2.

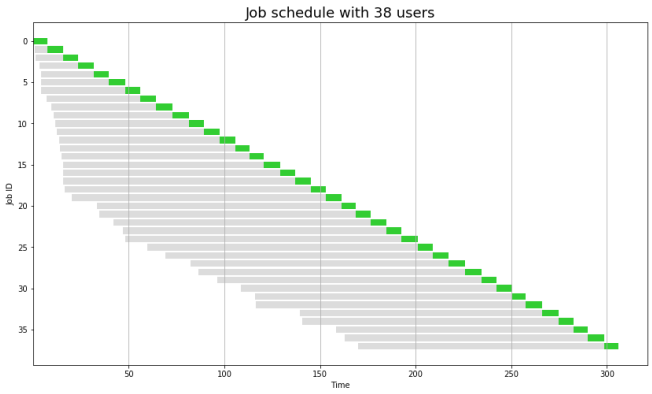

IMAGE 2: Gantt chart of a simulation

This chart represents the moments when all the users in one of the simulations join and wait in queue, start being served and depart in a simulated attraction. The part where a user is waiting in queue is shown with a grey bar, while the part where it is served is painted in green. It can be seen that the arrival times, which are the beginning of the grey bars, show a Poisson distribution. On the other hand, the service times, which are the lengths of the green bars, are almost the same in all the instances because they represent the ride times, which are constant in parks.

The image 3 compares the difference on waiting times from an underloaded, a standard and an overloaded attraction respectively. The waiting times are plotted using a histogram in order to bring into focus how the waiting times vary depending on the flow of the queue and the number of users. There's also a red vertical line that indicates the average waiting time of all the simulations to provide additional information.

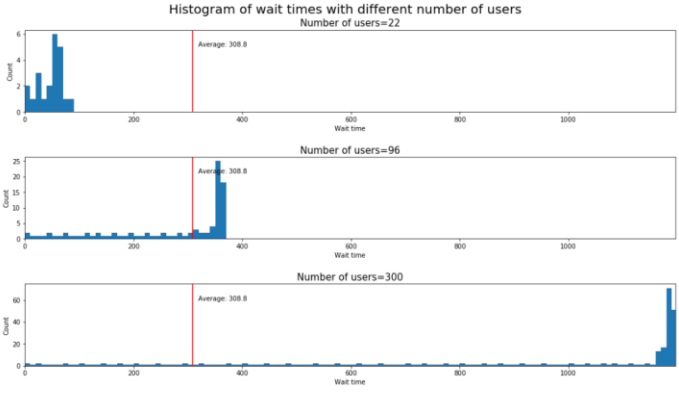

IMAGE 3: Histogram of waiting times

6 Next Steps

The tasks presented before are just the beginning of this project. As seen in table 1, there are more steps before concluding this thesis. First of all, the plotting section in Python is still not completed, as I am looking forward to adding more interesting and useful diagrams and charts.

Later on, the design and creation of dashboards in Power BI will begin. This will be one of the longest tasks in this project, together with the creation of the model I have already made. These dashboards will allow the user to have easier access to key information about the queuing system and will be able to implement real-time data in the future.

Once this dashboards are completed, the implementation of alarms will begin. This part of the project will be able to give real-time feedback to the park's managers and, therefore, give them the opportunity to make crucial decisions before long queues emerge in the busiest attractions.

References

[1] M/M/1 queuing system (2021). EventHelix. [consulted: March 6, 2023]. Available on the Internet: https://www.eventhelix.com/congestion-control/m-m-1/

[2] SimPy overview (2016). Team SimPy. [consulted: March 5, 2023]. Available on the Internet: https://simpy.readthedocs.io/en/latest/

[3] Introduction to simulation with SimPy (2022). Darío Weitz. [consulted: March 12, 2023]. Available on the Internet: https://towardsdatascience.com/introduction-to-simulation-with-simpy-322606d4ba0c

[4] Matplotlib (2012). The Matplotlib development team. [consulted: April 6, 2023]. Available on the Internet: https://matplotlib.org/

[5] Power BI (2015). Microsoft. [consulted: March 12, 2023]. Available on the Internet: https://powerbi.microsoft.com/en-us/

[6] Virtual queuing in amusement parks (2021). Attractions.io. [consulted: March 8, 2023]. Available on the Internet: https://attractions.io/learn/virtual-queuing-the-future-of-theme-park-experiences

[7] Adapting Agile Scrum methodology for individuals (2017). Lucidchart. [consulted: March 8, 2023]. Available on the Internet: https://www.lucidchart.com/blog/scrum-for-one

[8] Microsoft Teams (2020). Microsoft. [consulted: March 12, 2023]. Available on the Internet: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software

[9] Pandas documentation (2023). Pandas team. [consulted: April 7, 2023]. Available on the Internet: https://pandas.pydata.org/docs/

[10] NumPy documentation (2020). NumPy developers. [consulted: April 7, 2023]. Available on the Internet: https://numpy.org/doc/stable/

[11] Gantt charts with Python's Matplotlib (2021). Thiago Carvalho. [consulted: April 2, 2023]. Available on the Internet: https://towardsdatascience.com/gantt-charts-with-pythons-matplotlib-395b7af72d72