Università degli Studi di Torino

School of Science of Nature
Department of Physics
Master's degree in Physics of Complex Systems

Agent-based simulation: hospital reception and M/M/c queue

Eleonora Misino

Academic Year 2018/2019

Abstract

In this report we describe an agent-based simulation of a hospital reception realized with NetLogo. The hospital reception is described as a $M/M/c$ queue and we use the related formulas to compute the theoretical values for the number of patients in the reception (L), the number of waiting patients (L_q) and the average time spent in queue (W_q). The system configuration is determined by three parameters: the number of front offices (c), the arrival rate (λ) and the service rate (μ). We simulate different configurations of the system and we compare the experimental results with the corresponding teorethical values of L, L_q and W_q. Because of the time discretization eamployed in the simulation, the theoretical values and the experimental ones are not equal. We study the trend of L, L_q and W_q at the change of c, λ and μ. As expected, L, L_q and W_q decrease with the increasing of c and μ, while they increase with λ.

Contents

1 Introduction to Queueing Theory

In a day routine there are lots of situations where we could find ourselves stuck in a queue. Let's think about the flow of commuters or about shopping: both situations represent potential sources of queues. There is a mathematical structure focused on the study of queues named Queueing Theory. This theory is applied in many fields from telecommunication to traffic engineering, from industrial engineering to optimization of public and private services. The Queueing theory was developed in the 20th century and is considered a branch of operational research. A queueing model is constructed in order to predict queue lengths and waiting time and its results are often used to solve problems in business, industry and society.
The key elements of a queueing system are the customer and the server. The former refers to anything that arrives at a facility and requires service, such as people, machines, trucks; the latter refers to any resource that provides the requested service, such as front office, retrieval machines, and runways. A queue is schematized with a queueing node, where customers arrive, wait a little in the waiting line and depart after being processed by a server (Figure 1 [1]). A queueing node can have one or multiple servers.

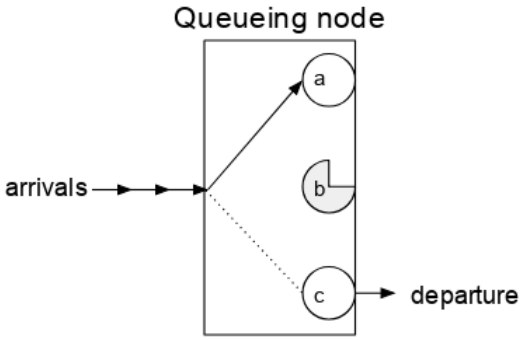

Figure 1: scheme of a queueing node with multiple servers. The customers arrive with particular services to require and depart when their services are provided.

The customers come from the calling population that can be either finite or infinite. The system has a specific capacity, finite or infinite, that sets a limit on the number of customers that may be in the waiting line.
Both arrivals and services have a time distribution that distinguishes a particular queueing model. Another fundamental aspect of a queueing system is the queueing discipline that determines the processing order.
There is a standard notation for queueing system introduced by D. G. Kendall [1]. Kendall's notation describes and classifies queueing models using six parameters written $A/S/c/K/N/D$ where:

- A refers to the arrivals distribution in time;
- S refers to the services time distribution;

[1] This image is *Dt-rush-8* own work with copyright CC BY-SA 4.0.

- c is the number of servers at the node;
- K is the system capacity;
- N is the size of the calling population;
- D refers to the queueing discipline.

When the last three parameters are not specified, it is assumed that $K = \infty$, $N = \infty$ and $D = FIFO$. $FIFO$ is the acronym of First-In First-Out and it means that the first customers entering in the node is the first leaving it.

1.1 M/M/c queue

One of the simplest queue systems is the $M/M/c$ queue, where the arrivals are governed by a Poisson process and the service times have an exponential distribution. The parameter M stands for Markovian or memoryless, this meaning that the system is a Markov chain where the future state is related only to the present state. The number of servers at the queue node is a finite number c, so if there are less than c customers, some of the servers will be idle. On the contrary, if there are more than c customers, the customers queue in the infinite buffer.
The state space of this stochastic process is the set $\{0, 1, 2, 3, \dots\}$ where each value corresponds to the number of customers present in the system, including any currently in service [2].
There are two fundamental parameters describing this system:

- the occurring rate of arrivals λ;
- the service rate μ characterising the exponential distribution of service times.

Birth-death process

It is important to understand the concept of equilibrium for a queue. For this purpose, we introduce the birth-death process, where there are only two state transitions named **birth** and **death**. When a birth occurs, the process moves from state i to $i + 1$, otherwise, when a death occurs, the process moves from state i to $i - 1$.
A birth-death process is characterised by birth rates $\{\lambda_i\}_{i=0\dots\infty}$ and death rates $\{\mu_i\}_{i=0\dots\infty}$, where i refers to the matching state [2]. The $M/M/c$ queue is a birth-death process with:

$$\lambda_i = \lambda \quad , \quad \forall \quad i \tag{1}$$

$$\mu_i = i\mu \quad , \quad \forall \quad i \leq c \tag{2}$$

$$\mu_i = c\mu \quad , \quad \forall \quad i \geq c \tag{3}$$

Figure 2 [2] shows the state space of a $M/M/c$ queue:

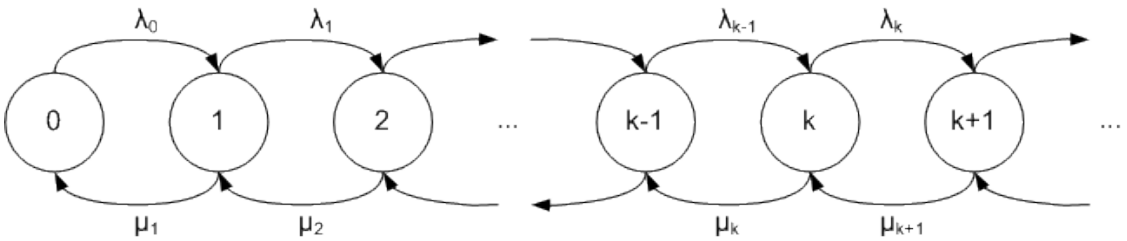

Figure 2: state space diagram of $M/M/c$ queue. Arrivals are governed by a Poisson process and move the process from state i to $i+1$ with rate λ_i; the service times are exponentially distributed and move the state i to $i-1$ with rate μ_i.

The $M/M/c$ queue is in a steady state when the arrival rate is equal to the departure rate. It is understandable that the equilibrium condition is $\lambda < c\mu$ [2]. Let us define $\rho = \frac{\lambda}{c\mu}$, so the equilibrium condition is now $\rho < 1$.

An M/M/c system: the hospital reception

An example of a system that can be simulated with a $M/M/c$ queue is the hospital reception. In this queueing system the customers are the patients, while the servers are the front offices. Let us imagine an hospital with a finite number of front offices and a potentially infinite number of patients to be accepted. The patients require different services, each of them with a different time to be computed. Let us assume that each front office is appointed to provide any required service. It is reasonable to assume that the first patient entering the queue is the first to be served, so we have a $FIFO$ queueing discipline.
This hospital reception can be studied through an agent-based simulation, reproducing the interaction between the patients and the reception with pre-set parameters for the arrivals distribution and the service time distribution.

[2] This image is *NeoUrfahraner* own work with copyright CC BY-SA 3.0.

2 Methods

2.1 Model overview

We use *NetLogo* 6.0.4 to create the agent-based simulation of the hospital reception. The model is based on the interaction between two different breeds of agents: *patients* and *front offices*. The number of front offices is fixed and configurable by the user. Two others fundamental configurable parameters are the arrival rate λ and the service rate μ.
The patients arrivals are governed by a Poisson distribution with mean λ. The queue discipline is *First In - First Out* ($FIFO$): the first patient entering the queue is the first to be served by any available front office. Each busy front office has a specific completion time for its current task. This time is determined by an exponential distribution with mean $\frac{1}{\mu}$.
Figure 3 shows the hospital reception with the waiting patients and the front offices. The idle front offices are green, while the busy ones are red. At the bottom there are three sliders to set the number of front offices, the arrivals rate λ and the service rate μ. The buttons *setup* and *go* enable to setup the model and to run the simulation.

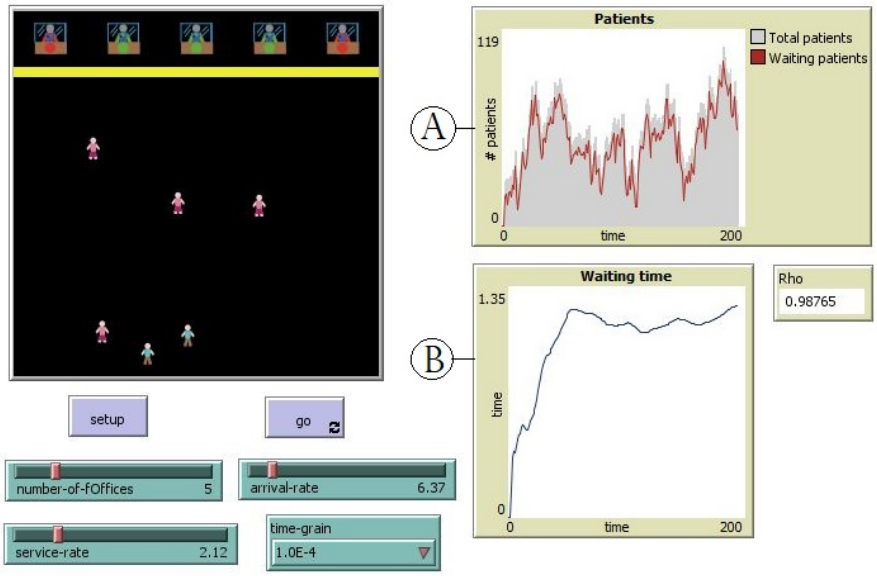

Figure 3: an overview of the model.

The number of patients in the reception and the number of waiting patients are plotted in graphic A, while in graphic B is reported the average time spent in queue. The value of ρ is displayed on the right. At the bottom there is also a slider, named *time-grain*, which allows a finer time discretization.

2.2 The discretization problem

In the first draft of the model there was a problem with the equilibrium condition. The number of patients in the system (L), the queue length (L_q) and the average time spent in queue (W_q) diverged, albeit $\rho < 1$.
The $M/M/c$ queue is a continuous Markovian process and L, L_q and W_q can be computed using the following formulas [2].

$$L = c\rho + \frac{\rho}{1-\rho}\pi_{c+} \tag{4}$$

$$L_q = \frac{\rho}{1-\rho}\pi_{c+} \tag{5}$$

$$L_q = \lambda\frac{\rho}{1-\rho}\pi_{c+} \tag{6}$$

In (4),(5) and (6) π_{c+} refers to the probability of having all the front offices (servers) busy[3].

In NetLogo the time is not continuous and its discretization introduces a bias in the queue simulation. Let us consider a process with service rate μ, so that the service time for each task is extracted from an exponential distribution with mean $\frac{1}{\mu}$. As we can see in Figure 4, with $\mu = 1$ is highly probable to extract a number smaller than 1.

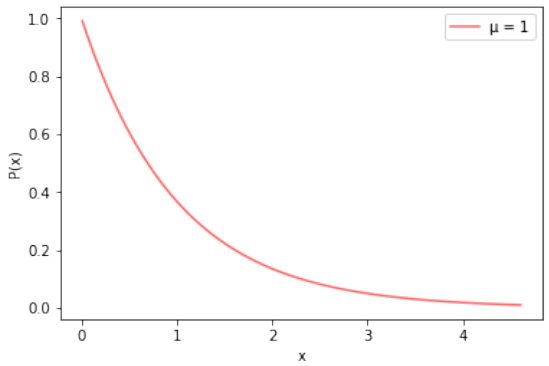

Figure 4: the probability distribution function for the exponential distribution with rate $\mu = 1$.

Using NetLogo *tick* as the time unit for the system, it is clear that we do a rough approximation: each service time smaller than 1 has to be approximated with 0 or 1 *tick*, introducing an error that causes the system divergence.

[3]See the Appendix 5.1 for the formulas inherent to π_{c+}.

This problem can be solved by creating a finer time grain. For this purpose, we can use the function *tick-advance* [4], that advances the tick counter by a floating point number. This function allows us to create a finer discretization, while the time unit is still a *tick* (Figure 5).

```
to go

  if (ticks) > 0 [

  set total-patients lput count patients total-patients
  set waiting-patients lput length queue waiting-patients
  set ave-time-in-queue lput mean time-in-queue ave-time-in-queue
  set busy-fOffices lput ((count fOffices with [color = red] / number-of-fOffices) * 100 ) busy-fOffices

  ]

  if ticks = 0  [

  set total-patients lput 0 total-patients
  set waiting-patients lput 0 waiting-patients
  set ave-time-in-queue lput 0 ave-time-in-queue
  set time-in-queue lput 0 time-in-queue
  set busy-fOffices lput 0 busy-fOffices
  ]

  if (precision ticks prec =  ceiling (precision ticks prec)) [

    patients-arrive

    set mean-patients mean total-patients
    set mean-waiting-patients mean waiting-patients
    set mean-time-in-queue (mean ave-time-in-queue)
    set mean-busy-fOffices mean busy-fOffices

    update-plots

    set total-patients []
    set waiting-patients []
    set ave-time-in-queue []
    set busy-fOffices []
  ]

  end-service
  begin-service

  tick-advance time-grain ;Advances the tick counter by time-grain.

```

Figure 5: code section in which the time discretization is implemented.

Even though the system now respects the equilibrium condition, this solution does not completely fix the problem: the experimental values of L, L_q and W_q do not match the theoretical ones. L, L_q and W_q are obtained from the simulation by computing the average number of patients in the system (L), the average number of waiting patients (L_q) and the average time spent in queue (W_q). These quantities are computed for each integer *tick*.

The threads represent an alternative solution. The idea is to create two threads, one for the observer and one for the model. Both the model and the observer have their own time, respectively t and T. The model executes its tasks (patients' arrival, services providing, etc.) independently of the observer, until t is equal to a multiple of T. At that point, the quantities L, L_q and W_q are computed. Therefore, the time grain is set by the observer and it is possible to get a finer discretization[5].

[4] This function has a precision problem, see the Appendix 5.2 for more details.

[5] Two examples of threads in NetLogo, developed by Pietro Terna: `https://terna.to.it/econophysics19/NetLogo_examples/2threads.nlogo` and `https://terna.to.it/econophysics19/NetLogo_examples/2threadsWithGraphics.nlogo`.

3 Results

We studied the system response to different values of its parameters c (number of front offices), λ (arrival rate) and μ (service rate). Figure 6, 7, 8 show the system evolution in the time interval $[0, 200]$ *ticks* for three different configurations, corresponding to three different values of ρ. In the first graphic on the right we can see the number of patients in the reception and the number of waiting patients. The second graphic shows the average time spent in queue. Both graphics are updated every *tick*. The time discretization is the same for all the configurations ($time - grain = 1 \times 10^{-4}$)

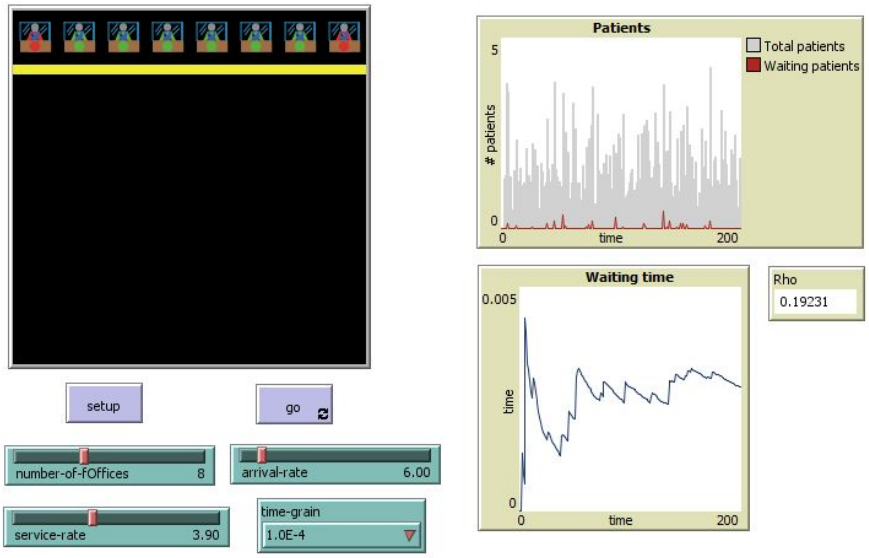

Figure 6: system time evolution with $c = 8, \lambda = 6.00$ and $\mu = 3.90$.

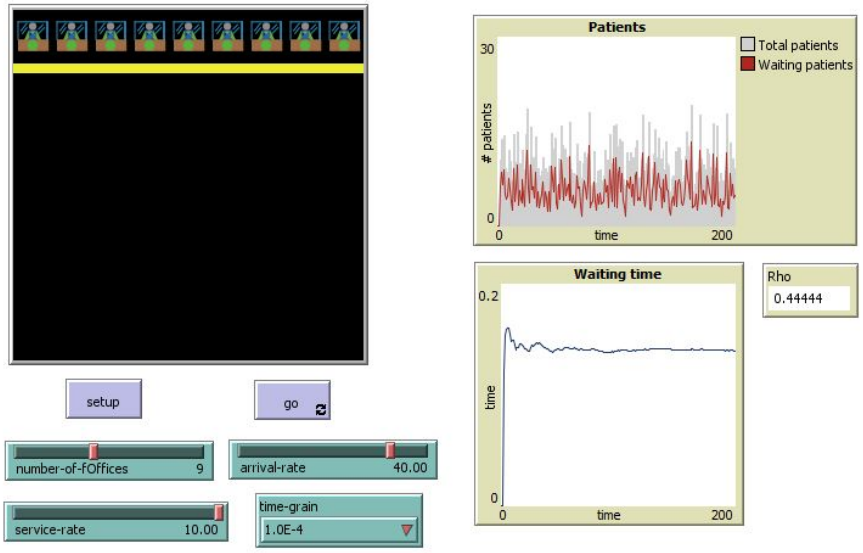

Figure 7: system time evolution with $c = 9, \lambda = 40.00$ and $\mu = 10.00$.

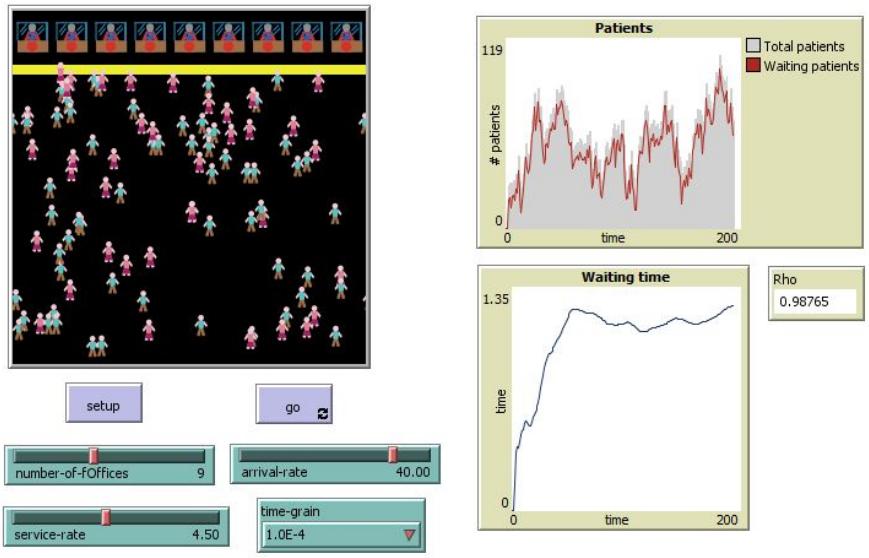

Figure 8: system time evolution with $c = 9, \lambda = 40.00$ and $\mu = 4.50$.

The experimental values of L, L_q and W_q shown in Figure 6 and Figure 7 are bigger that the theoretical ones (Table 1). On the contrary, comparing the third row of Table 1 with the results shown in Figure 8, we observe that the model results underrate the theoretical values.

c - λ - μ	$L(patients)$	$L_q(patients)$	$W_q(s)$
8 - 6 - 3.9	1.54	4.93×10^{-5}	8.21×10^{-6}
9 - 40 - 10	4.02	1.90×10^{-2}	4.75×10^{-4}
9 - 40 - 4.5	85.51	76.62	1.92

Table 1: Theoretical values for L, L_q and W_q, computed with the formulas (4), (5) and (6). The first column displays the values of c, λ and μ.

We run the model with $c = 8$, $\lambda = 7.01$ and $\mu = 0.88$ for about 1000 *ticks* to better understand its behaviour when ρ is close to 1. In this configuration $\rho = 0.9957$ and the theoretical values are $L = 238.41$, $L_q = 230.45$ and $W_q = 32.87$. The experimental results are shown in Figure 9 and they clearly underrate the theoretical values.

These differences highlight that the time discretization employed in the simulation is not fine enough to approximate the continuous nature of the $M/M/c$ queue, even though it is able to stabilize the system, as shown below.

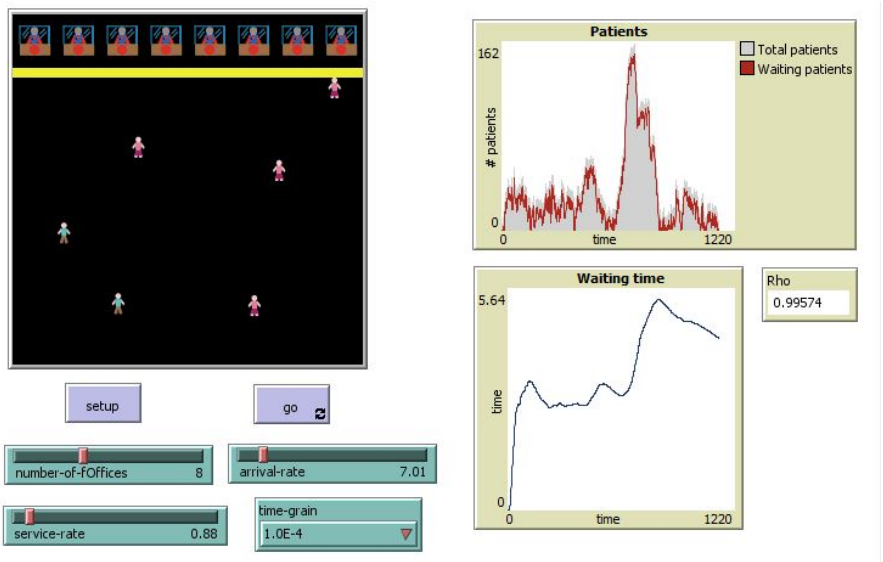

Figure 9: system time evolution with $c = 8, \lambda = 7.01$ and $\mu = 0.88$.

We also studied the system response to different values of *time-grain*. As shown in Figure 10, the system does not respect the equilibrium condition if the *time-grain* parameter is set to 1, while by using a finer time discretization the system responds correctly (Figure 11).

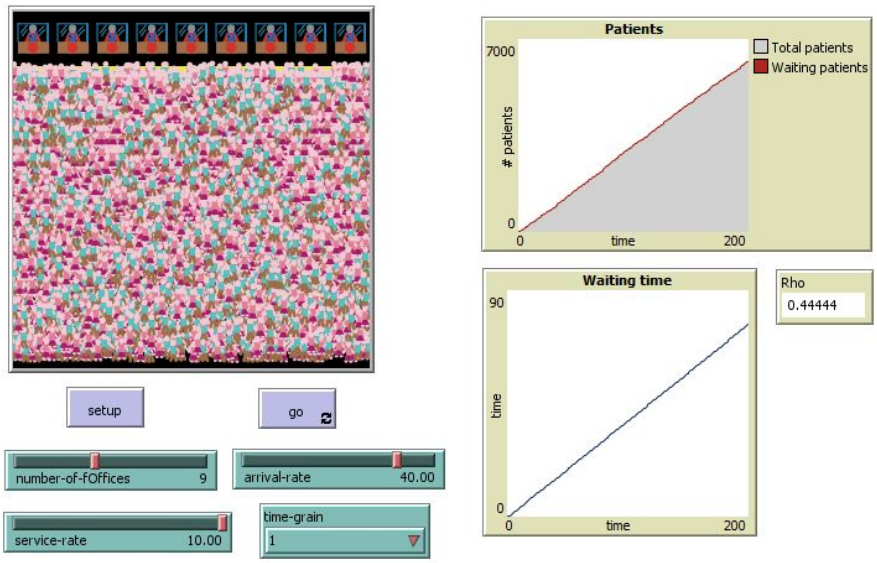

Figure 10: the system diverges with $time - grain = 1$, even though $\rho < 1$.

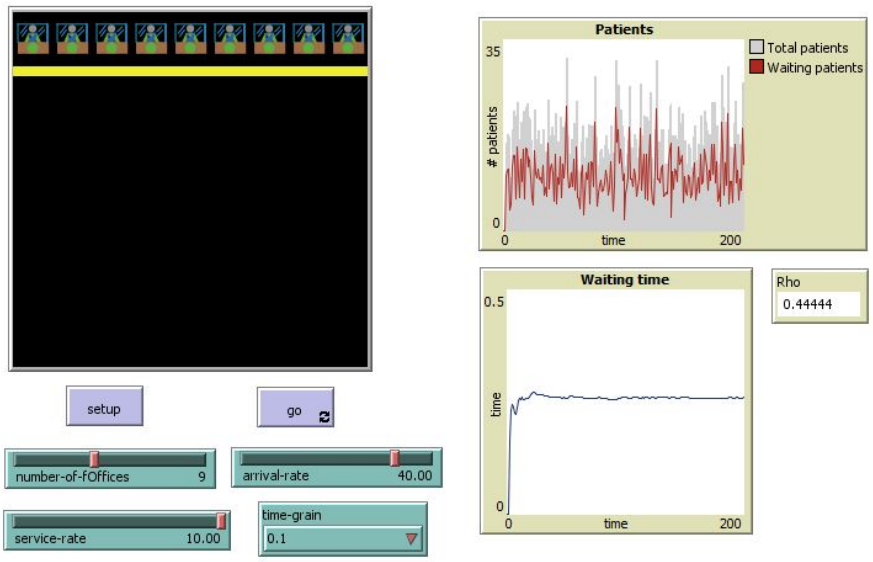

Figure 11: the system respects the equilibrium condition with $time - grain = 0.1$ *ticks*.

4 Discussion

This simulation can be used to study the relation between the number of front offices, the service rate or the patients' arrival rate and the meaningful quantities of the system. The results can be used as a starting point for an optimization process of a hospital reception.
The following graphics are obtained by setting different values of c, λ and μ and by computing L, L_q and W_q from the simulation outputs.
In Figure 12 the fixed parameters are $\lambda = 40.00$ and $\mu = 10.00$. We run the model seven times for 200 *ticks* with a different number of front offices. As expected, L, L_q and W_q decrease with the increasing c.

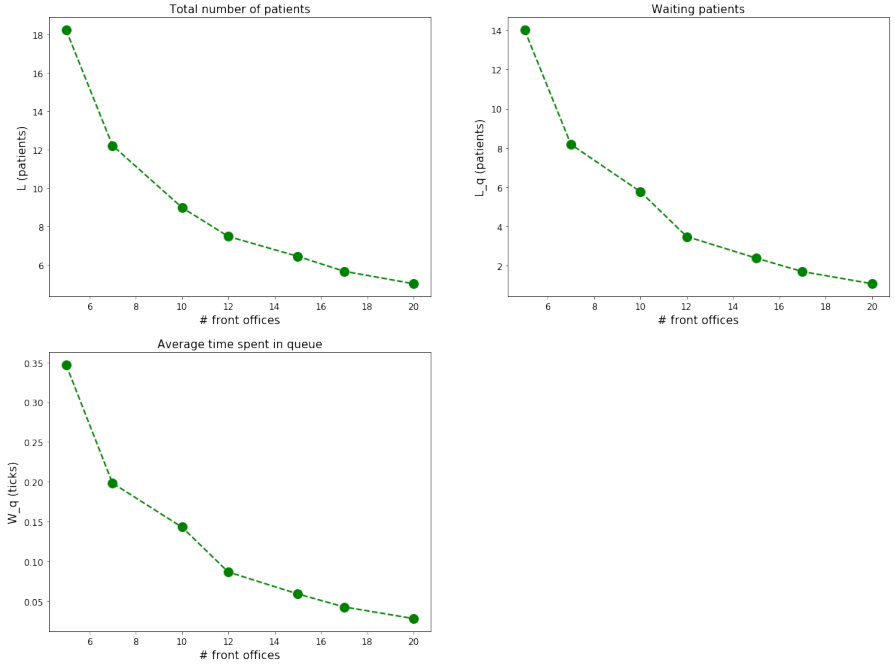

Figure 12: trend of L, L_q and W_q for different values of c, with $\lambda = 40.00$ and $\mu = 10.00$.

The same process has been repeated for different service rates, with $c = 10$ and $\lambda = 10.00$. As expected, L, L_q and W_q decrease with the increasing of the service rate (Figure 13).

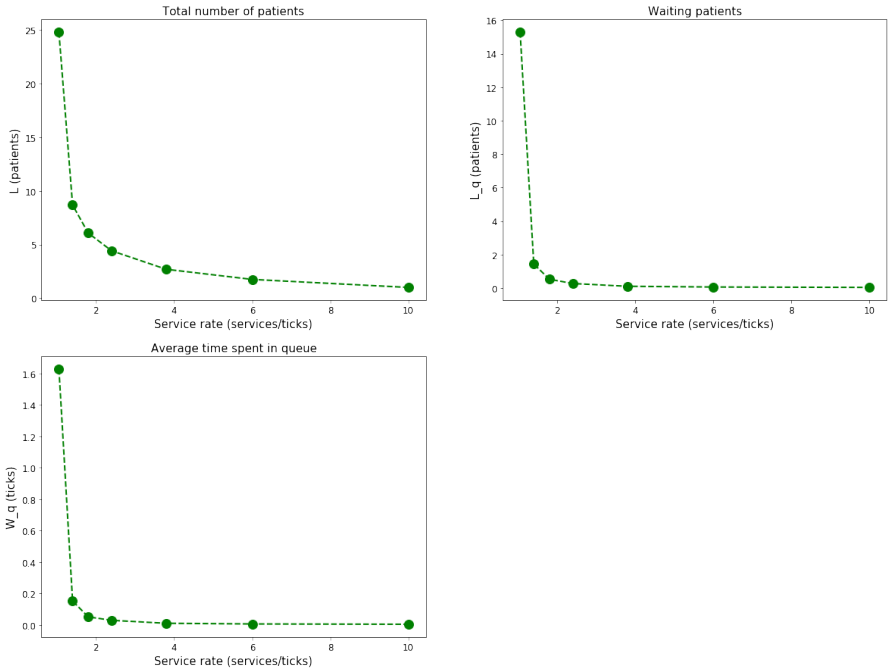

Figure 13: trend of L, L_q and W_q for different values of μ, with $c = 10$ and $\lambda = 10.00$.

We set $c = 10$ and $\mu = 10.00$ and we run the model five times for 200 *ticks* with an increasing rate of arrivals. The result, shown in Figure 14, is an increasing trend of L, L_q and W_q.

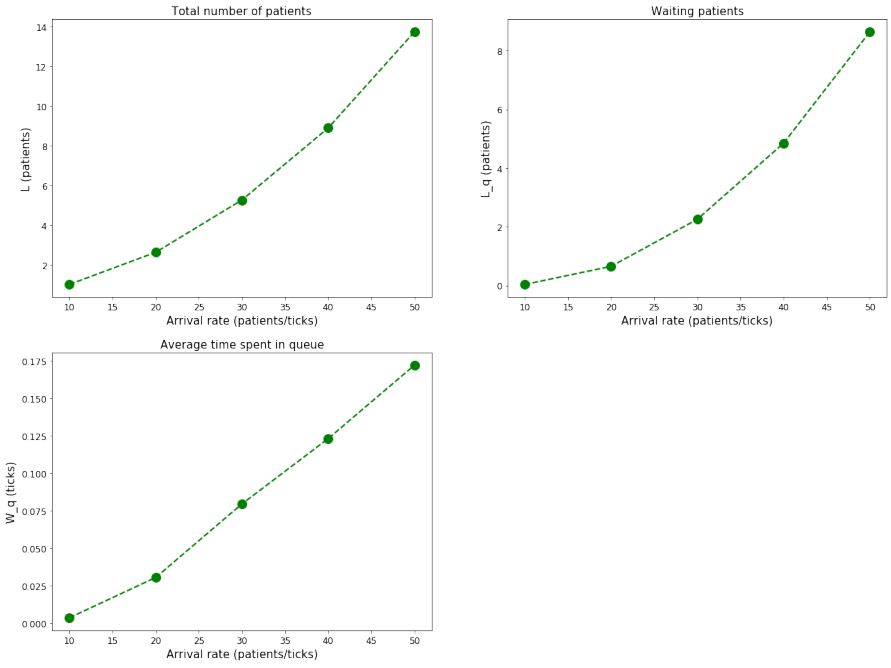

Figure 14: trend of L, L_q and W_q for different values of λ, with $c = 10$ and $\mu = 10.00$.

These results show that there are more than one strategies to reduce the queue length and the waiting time:

- increasing the number of front offices;
- increasing the service rate;
- decreasing the arrival rate.

The first strategy is easy to achieve, but it could be too much expensive. On the other hand, the last two strategies are not easy to realize. Starting from the simulation results, it could be useful to set a series of experiments designed to find the best solution. An example is a finer study of the queue discipline. Another possibility could be to specialize the front offices: different services will be provided by different front offices. This study should be combined with an analysis of the service types distribution, in order to optimize the number of front offices for each type of service required.

5 Appendix

5.1 M/M/c theoretical formulas

In this section we show the formulas for the probability mass function of the system stationary distribution [2].

$$\pi_{c+} = \frac{(c\rho)^c}{c!(1-\rho)}\pi_0 \tag{7}$$

$$\pi_0 = \left[\left(\sum_{k=0}^{c-1}\frac{(c\rho)^k}{k!}\right) + \frac{(c\rho)^c}{c!(1-\rho)}\right]^{-1} \tag{8}$$

π_k refers to the probability that the system contains k customers.

5.2 *tick-advance* function

The *tick-advance* function advances the tick counter by a number close to its argument. For example, the command *tick-advance 0.1* produces the tick counter advancement shown in Figure 15.

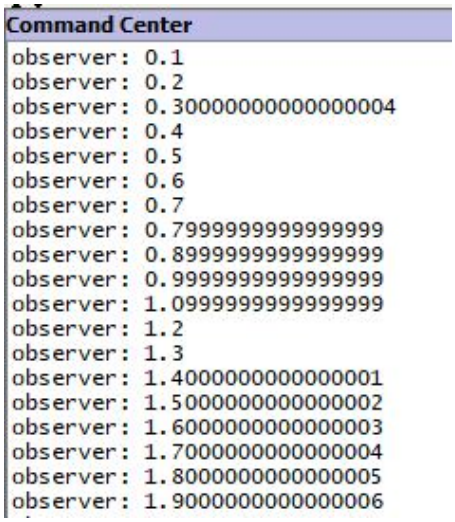

Figure 15: tick counter advancement product by *tick-advance* 0.1.

This imperfection causes several problems when we try to identify an integer tick, which correspond to the time unit for the system. To solve this problem, we use the function *precision*, in order to erase the pathological digits. The number of significant digits is set according to the current time discretization, i. e. is the argument of *tick-advance*.

Bibliography

[1] (2013) Kendall's Notation. In: Gass S.I., Fu M.C. (eds) Encyclopedia of Operations Research and Management Science. Springer, Boston, MA

[2] (2016) J. Sztrik, *Basic Queueing Theory* , University of Debrecen, Faculty of Informatics