University of Camerino

School of Science and Technology

Master of Science in Computer Science (LM-18)

Course of Performance Analysis and Simulation

Shop Manager Project Report

Student
Marco Scarpetta

Professor
Michele Loreti

Student ID
115926

A.Y. 2020/2021

Index

Introduction

In this report I present the work done as project for the PAS (Performance Analysis and Simulation) course at the University of Camerino.

The project requires to use the framework *Sibilla* in order to model a given system and then analyse the results obtained during the simulation of that system.

Follows a description of the scenario I modeled.

Project Description - Shop Manager

A shop owner aims to estimate the number of sales clerks that are needed to satisfy requests and expectations of his customers.

We know that in the average a customers arrives at the shop every ma minutes, while a clerk needs ms minutes to serve a customer.

Unfortunately, in the shop at most N customers can wait at the same time to be served. We also let K denote the number of clerks working in the shop.

Let $N \in \{5, 10, 25\}$, $K \in \{1, 2, 5\}$, $m_a = \{1, 2, 5, 10\}$, and $m_s = \{1, 2, 5, 10\}$ use the methodologies considered in the lecture to estimate the following values:

- clerks utilisation (fraction of time a clerk is working);
- average number of customers served by the shop per time unit;
- average number of waiting customers;
- average number of customers per time units that cannot enter in the shop.

Hence the main goal of the project is to define a population model describing the scenario mentioned above and, after running a system simulation through the framework *Sibilla*, analyse the data obtained in order to draw the required values.

Population Model

In this chapter I discuss how I implemented the population model, starting from the parameters and the species till the rules. Of course, I also describe the initial state of the system and the measure that I used in order to get the expected results.

As required by the framework *Sibilla*, the model is specified in a file having *.pm* extension.

Parameters

First of all, I started by defining all the parameters needed for the system. In particular, as described in the specification, I used:

N: parameter that represents the maximum number of customers that can wait in the shop at the same time. It has to be taken as input by the system when the initial state is set using the *"init"* command;

K: parameter that represents the number of working clerks in the shop. Also this parameter has to be set with *"init"* command;

Ma: parameter used to represent the time needed for a new customer arrival;

Ms: parameter used to represent the time needed for a clerk to serve a customer. Both *Ma* and *Ms* can be configured via *"set"* command.

To model the system I also defined two further constants:

R: constant representing the rate function used in the transitions: it is set to 1.0 by default;

T: constant used to represent the total number of customers that can be involved in the system: for the sake of simplicity, is set by default to 20 and cannot be modified when the initial state is configured.

Species

The next step was to identify those species that are involved in the model.

From the specification I thought a scenario in which a customer is the focus of the system and in particular he can be in four different states. For this reason I modeled the following species:

F: species representing a potential customer that is still far from the shop;

A: species representing the fact that a customer has just arrived at the shop;

W: species that represents a customer that is waiting in the shop to be served;

S: species representing the fact that a clerk is serving a customer.

Moreover, other two species were necessary to keep track the number of waiting customer, and consequently the available space in the shop, and the number of working clerk. Hence I defined:

P: species that represents the available space in the shop, thus the number of customers that can enter;

C: species representing the number of available clerks that can serve the waiting customers.

Rules

Once I defined the parameters and the species I modeled the real behavior of the system that mainly consists of four different rules:

far_to_arrival (f_to_a): rule that represent a customer that passes from a state in which he is far from the shop (F) to a state in which he arrives at the shop (A).

The rate of this transition is in inverse relation to the number Ma, that is the time needed for a new customer arrival (the higher is Ma, the lower is the probability of a new arrival and it is reasonable due to the more time needed):

$$F \text{ -[} \tfrac{R}{Ma} \text{]-> } A \tag{2.1}$$

arrival_to_waiting (a_to_w): rule representing a customer that enters in the shop and starts waiting for a clerk. Of course the transition is enabled if and only if there is room in the shop, which means a species P is available.

Once the rule is performed (for simplicity here the rate is set to R, i.e. 1.0) the customer passes to state W while the species P disappears, meaning that one available space has been occupied:

$$A \mid P \text{ -[} R \text{]-> } W \tag{2.2}$$

waiting_to_serving (w_to_s): rule representing a customer that starts to be served by a clerk. Of course the transition is enabled if and only if there is at least one available clerk in the shop, which means a species C is free.

Once the rule is performed (also here the rate is set to R) the customer passes to state S while the species C disappears, meaning that one clerk is now busy:

$$W \mid C \text{ -[} R \text{]-> } S \tag{2.3}$$

serving_to_far (s_to_f): rule that represent a customer that has been served and go out from the shop. The rate of this transition is in inverse relation to the number *Ms*, that is the time needed for a clerk to serve a customer (the higher is *Ms*, the lower is the probability to get out the shop and it is reasonable due to the higher serving time).

At the end the customers passes to the state F whereas both species P (one space in the shop) and C (one clerk) return available:

$$S -[\ \tfrac{R}{Ms}\ ]\rightarrow F \mid P \mid C \tag{2.4}$$

Measures

Last but not least, in order to estimate the required values, I defined a custom measure E: the latter is used to keep track of the number of customer per time units that cannot enter in the shop.

This measure is given by the maximum between zero and the number of customers who have arrived at the shop minus the number of customer that can actually enter (the latter is obtained by subtracting the number of waiting customers and working clerks from the number N, i.e. the maximum number of customer that can wait in the shop).

In the code the mentioned measure is defined using the ternary operator as follows:

```
measure E = #A - (N - #W - #S) > 0 ? #A - (N - #W - #S) : 0;
```

Code 2.1: Measure representing the customers that cannot enter in the shop

Actually *Sibilla* does not allow the use of a single parameter or constant when a new measure is defined, so in the code I replaced N with the same integer given as input (of course, it must be a future improvement of the framework).

No further measures are required because, in order to analyse the average number of waiting customer or working clerks, the data produced by the simulation already contain all the necessary information (for each species we have the mean, variance and standard deviation per time units).

PCTMC model

According to the definition, a Population Continuous Time Markov Chain (PCTMC) model is a tuple consisting of a vector of variables X, a counting domain D, a set of transitions T and an initial state *d0*. At this point we are able to define:

- The vector of variables representing the six species I defined above:

$$X = \langle\ F,\ A,\ W,\ S,\ P,\ C\ \rangle \tag{2.5}$$

- The domain of the species, indeed F and A depends on the number of possible customers (T), W and P depend on the number of customer that can wait in the shop (N), while species S and C are affected by the number of clerks K:

$$D = [\,0, T\,] \times [\,0, T\,] \times [\,0, N\,] \times [\,0, K\,] \times [\,0, N\,] \times [\,0, K\,] \tag{2.6}$$

- The set of transitions representing the four rules previously described:

$$\begin{aligned} T = \{&\langle\, f\_to\_a,\ 1_F,\ 1_A,\ R/Ma\,\rangle, \\ &\langle\, a\_to\_w,\ 1_A + 1_P,\ 1_W,\ R\,\rangle, \\ &\langle\, w\_to\_s,\ 1_W + 1_C,\ 1_S,\ R\,\rangle, \\ &\langle\, s\_to\_f,\ 1_S,\ 1_F + 1_P + 1_C,\ R/Ms\,\rangle\} \end{aligned} \tag{2.7}$$

- The initial state of the system that consists of a number T (set to 20) of species F, a number N of species P and a number K of species C (parameters N and K are taken as input):

$$d0 = \langle\, T,\ 0,\ 0,\ 0,\ N,\ K\,\rangle \tag{2.8}$$

  In the code the initial state is defined as follows:

```
system init(N, K) = F < T > | P < N > | C < K > ;
```

Code 2.2: Initial State of the system

At this point the model has been implemented so the focus goes to the simulation phase.

Simulation

In this chapter I present the result obtained by performing the simulation of the system: in particular, multiple scenarios have been taken into account indeed different combinations of the rates and the numbers of customers or clerks have been tested.

However, first of all I would like to introduce how to run the system because, in addition to the standard way, I implemented an interesting python script that could simplify all the process.

How to run

Once the system has been modeled we are able to perform the simulation.

- The first step is to download the framework Sibilla:

  ```
  git clone https://github.com/quasylab/sibilla.git
  ```

- Then run the available *gradle* script to build the tool (replace *./gradlew* with *.\gradlew.bat* in Windows):

  ```
  cd sibilla
  ./gradlew build
  ./gradlew installDist
  ```

  The compiled tool is then available in the folder *shell/build/install/sshell*.

- Download the shop_manager project that I implemented:

  ```
  git clone https://github.com/Scarp94/shop_manager.git
  ```

- Move the *shop_manager* folder into *sibilla/shell/build/install/sshell/examples*

Now we can run the simulation in two different ways.

Via Sibilla Shell

The first option is to run the simulation using the interactive *Sibilla* shell (sshell):

- In *./sibilla* directory run:

  ```
  cd shell\build\install\sshell\bin
  ```

  Or in *./shop_manager* directory run:

```
cd ..\..\bin
```

- Then start the simulation environment using:

  ```
  sh sshell
  ```

  on Linux/MacOS systems or simply:

  ```
  sshell.bat
  ```

  on Windows machines.

- Finally type one by one the available and appropriate shell commands to start the simulation.

  Otherwise we can directly run the *.sib* file that I created (code 3.1) containing all the necessary commands for the simulation:

  ```
  run "../examples/shop_manager/shop_manager.sib"
  ```

  Using this approach the resulting data will be stored in the default folder *./examples/shop_manager/results*.

```
module "population"
load "../examples/shop_manager/shop_manager.pm"
set "Ma" 1.0
set "Ms" 1.0
env
init "init" (10.0, 5.0)
add all measures
measures
deadline 480
dt 1.0
replica 100
simulate
save output "../examples/shop_manager/results"
            prefix "shop_Manager" postfix ""
```

Code 3.1: Example of the .sib file

Via ad-hoc Python script

The second way to run the simulation is to use the Python script *main.py* I implemented that maybe could simplify all the process.

Since the project requires to analyse the system under different scenarios and combination of parameters, every time you should type each command using the *sshell* or you should change the parameter in the *.sib* file and re-run the simulation. But of course this is a bit cumbersome and time consuming.

The script aims to solve this problem and indeed to perform the simulation it is only necessary to run it passing the required parameters:

- In *./sibilla* directory run the command to move in the *shop_manager folder*:

```
cd shell/build/install/sshell/examples/shop_manager
```

- Then simply run:

  ```
  python main.py -w 10 -c 5 -a 1 -s 1
  ```

  Where:

  1. **-w** (or –wait) is maximum number of waiting customers, corresponding to parameter N;
  2. **-c** (or –clerks) is the number of available clerks, corresponding to parameter K;
  3. **-a** (or –arrival) is the time needed for a new customer arrival, corresponding to parameter Ma;
  4. **-s** (or –serve) is the time needed to serve a customer, corresponding to parameter Ms;

The script performs the following task:

- Parser of the given parameters;
- Creation of the output directory;
- Definition of the commands that *Sibilla* has to run (the ones that we can also write in the *.sib* file);
- Replacement of the special characters according to the operating system of the user;
- Finally it runs the commands.

It is easy to notice that in this way we can perform several simulation simply passing different parameters to the script.

In this manner the resulting data will be stored in *./examples/shop_manager/results* in sub-folders having different names depending on the passed parameters (for instance *W10_C5_A1_S1*).

Moreover, a strength of the script is the fact that can be run using different operating systems (tested on Window 10, Ubuntu 10.04 and Mac Os 10.15).

Scenarios

In order to test the system, different scenarios have been analysed and several simulations have been performed.

In particular I decided to focus on five combinations of parameters that I considered relevant to carry out some reasoning and analysis:

	N	K	Ma	Ms
Scenario 1	5	2	1	1
Scenario 2	10	2	1	1
Scenario 3	10	5	1	1
Scenario 4	10	5	1	2
Scenario 5	10	5	2	2

Starting with a configuration in which at most 5 customers can wait in the shop (N), only 2 clerks are available (K) and 1 minute is required for a new customer arrival and to serve a customer (Ma and Ms), each simulation is performed by increasing just one of those parameters: in this way we can better analyse how the single parameter affects the system.

Of course there are also further parameters which are in common for each simulation:

deadline: represents the simulation deadline, it is set to 480 in order to simulate a working day (8 hours times 60 minutes);

dt: represents the sampling time, it is set to 1;

replica: represents the number of simulation replicas, it is set to 100 (that's enough to obtain valuable results).

Given a starting scenario, in my opinion, the expectations are the following:

- if we increase N then the number of species W should increase while the number of customers that cannot enter in the shop should decrease;
- if we increase K then the number of species W and the number of customers that cannot enter in the shop should decrease, while the number of species S should increase;
- if we increase Ma then the number of waiting and served customers (so the clerks utilization too) and the number of customers that cannot enter in the shop should decrease;
- if we increase Ms then the number of waiting customers, the number of customers that cannot enter in the shop and the number of species S (so the clerks utilization as well) should increase.

Starting from those expectations, the objective of the project was to use *Sibilla* in order perform different simulations and then analyse the resulting data.

Thus, once defined the scenarios that has to be analysed, the next step was to run the system simulation by leveraging the Python script I implemented using the different parameters.

Before analyzing the results I would like to highlight the fact that the simulation is performed according to an older version of *Sibilla* (refer to commits made on June 30, 2021). So if one would replicate the same test I did using the latest version, maybe some incompatibility issues may arise.

Results

The results of each simulation are several *.csv* file, two for each added measure (one represents the number of the species and the other represents the fraction of that species).

Those files, each of which contains data concerning the mean, the variance and the standard deviation per time unit, were used to get some charts in order to perform the required analysis and check that the results are in line with respect to the expectations.

Since the simulation produced several *.csv* file, I implemented another *Python* script that is able to manipulate and analyze them: in particular I used the *Pandas* library to do some filtering and to add some features to the dataset, while I use *Matplotlib* to get the desired charts.

As required from the project description, I made the four analysis obtaining the following results:

1. **Average number of waiting customers**

   Here I used the data in the file containing the information about the number of species W representing the customers who are waiting in the shop, obtaining the following charts:

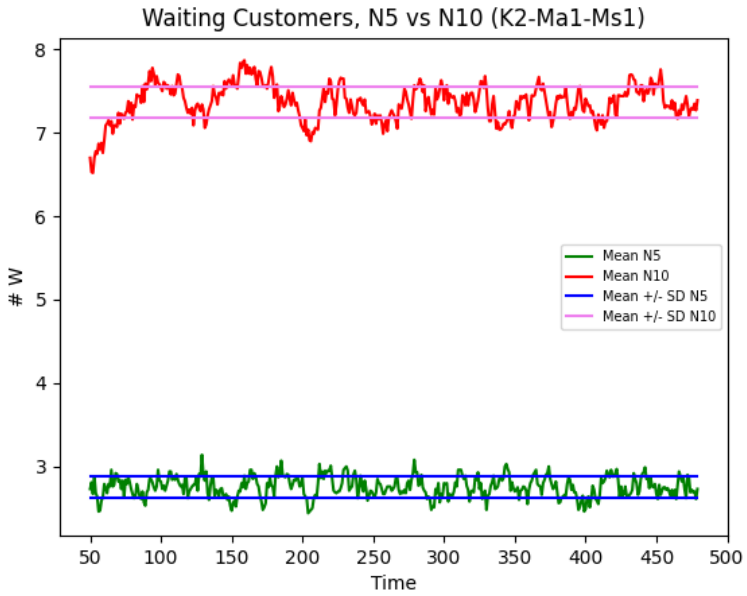

Figure 4.1: Chart of waiting customers, N=5 vs N=10

In the chart 4.1 we can see that the number of waiting customers increases when the parameter N also increases. Of course it is the expected and right behaviour because it means that more customer can wait in the shop.

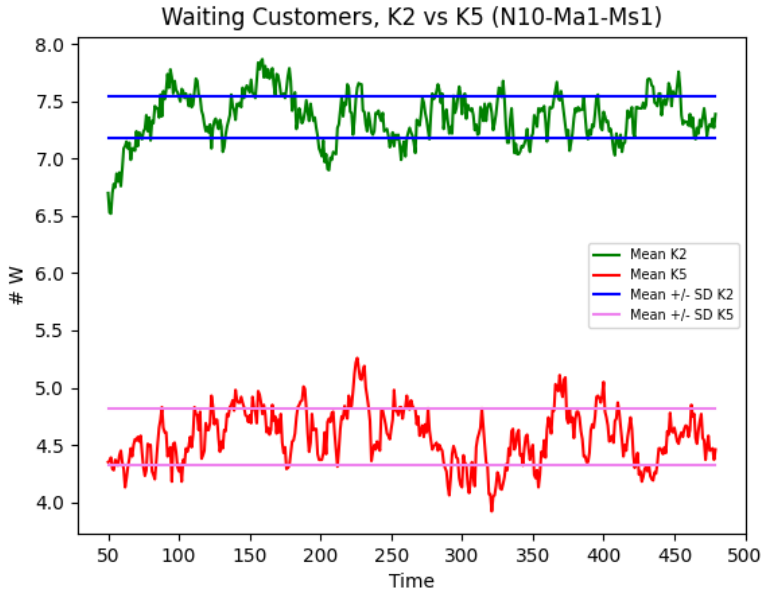

Figure 4.2: Chart of waiting customers, K=2 vs K=5

Instead in the chart 4.2 we can note that the number of waiting customers decreases when the parameter K increases and also here it is the expected and right behaviour: it means that there are more working clerks so a higher number of customers are served.

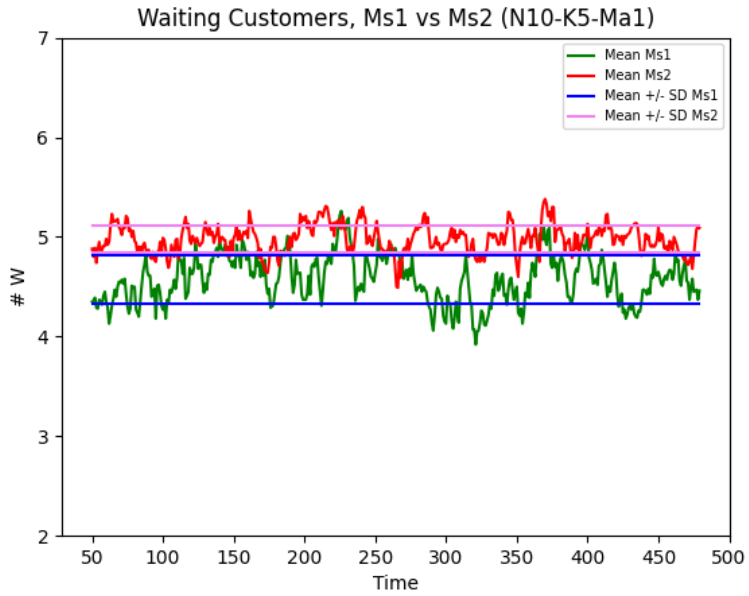

Figure 4.3: Chart of waiting customers, Ms=1 vs Ms=2

In the chart 4.3 we can see that the number of waiting customers increases when the parameter *Ms* also increases: it means that when a clerk needs more time to serve a customer it is reasonable that we could have more customer that are waiting in the shop.

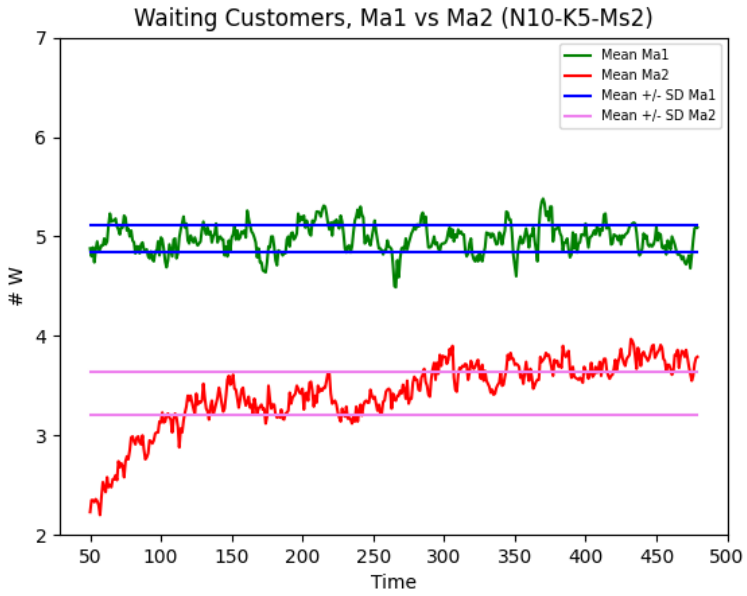

Figure 4.4: Chart of waiting customers, Ma=1 vs Ma=2

Rather, in the chart 4.4 we can observe that the number of waiting customers decreases when the parameter *Ma* increases: it means that when a customer needs more time to arrive in the shop it is reasonable that we could have more customer that are waiting in the shop.

2. **Average number of customers that cannot enter**.

   Here I used the data in the file containing the information about the custom measure *E* that I defined, obtaining the following charts:

   In the chart 4.5 we can see that the number of customers that cannot enter in the shop decreases when the parameter *N* increases: it means that if a higher number of customer can wait in the shop, of course there will be less customer outside it.

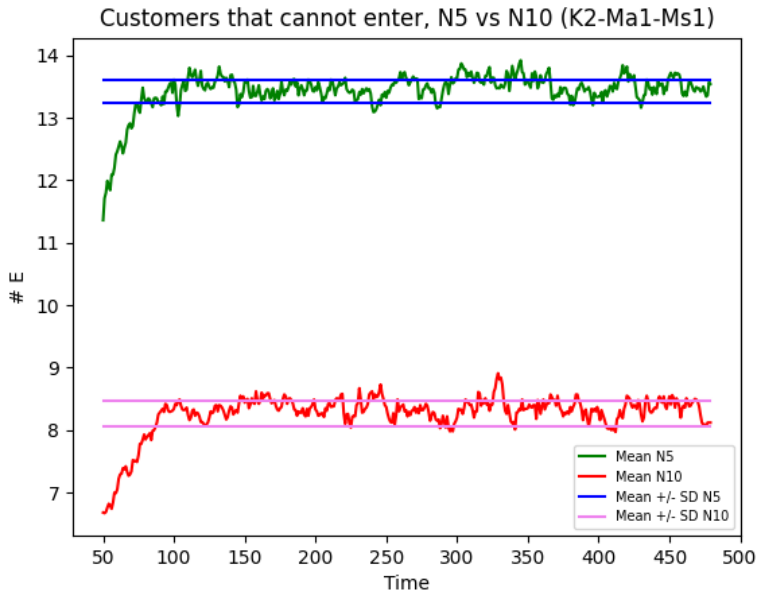

Figure 4.5: Chart of customers that cannot enter, N=5 vs N=10

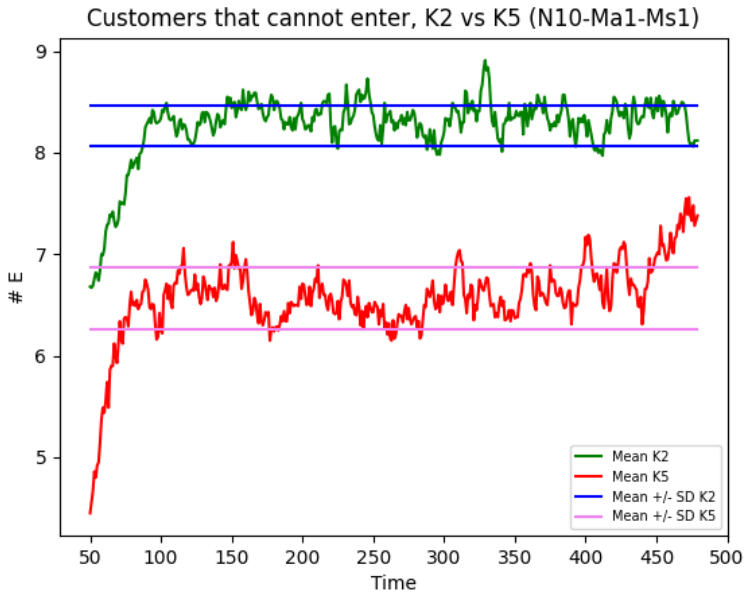

Figure 4.6: Chart of customers that cannot enter, K=2 vs K=5

Also in the chart 4.6 we can see that the number of customers that cannot enter in the shop decreases when the parameter K increases: it means that a higher number of customer are served so they move more quickly from one state to another without remaining too much outside the shop (species E).

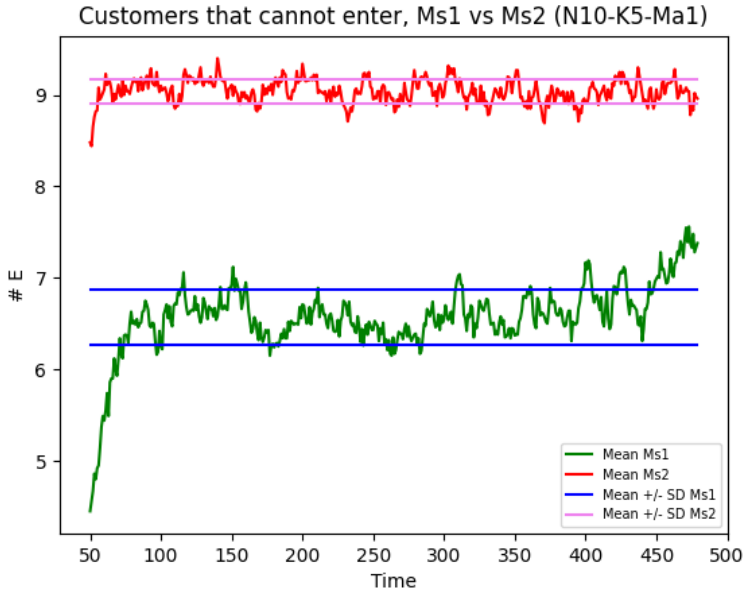

Figure 4.7: Chart of customers that cannot enter, Ms=1 vs Ms=2

On the contrary, in the chart 4.7 we can state that the number of customers that cannot enter in the shop increases when the parameter *Ms* increases: it means that, since a clerk need more time time to serve a customer, we will find more waiting clients in the shop and consequently also more clients outside of it.

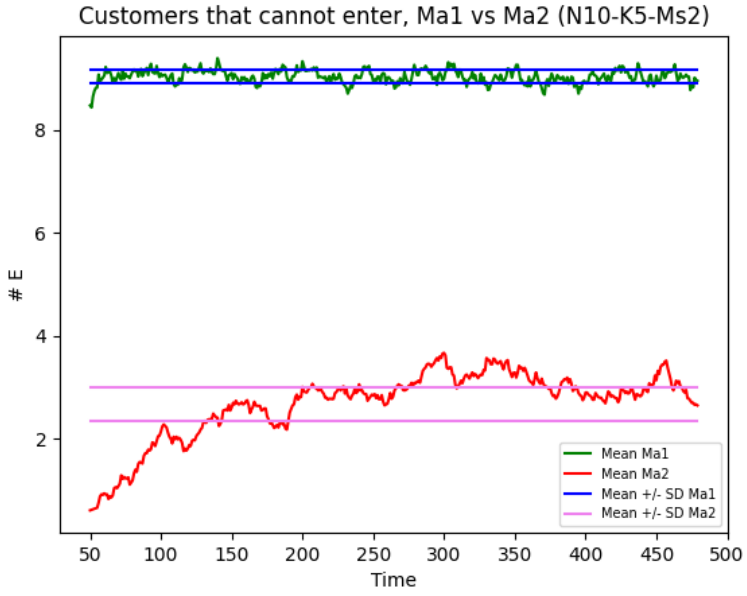

Figure 4.8: Chart of customers that cannot enter, Ma=1 vs Ma=2

Whereas in the chart 4.8 we can point out that the number of customers that cannot enter in the shop decreases when the parameter *Ma* increases and, of course, it is the expected behavior because a customer need more time to arrive

at the shop so we have fewer waiting clients.

3. **Average number of customers served by the shop**

To perform this analysis I used the data in the file containing the information about the number of species S representing the served customers, obtaining the following charts:

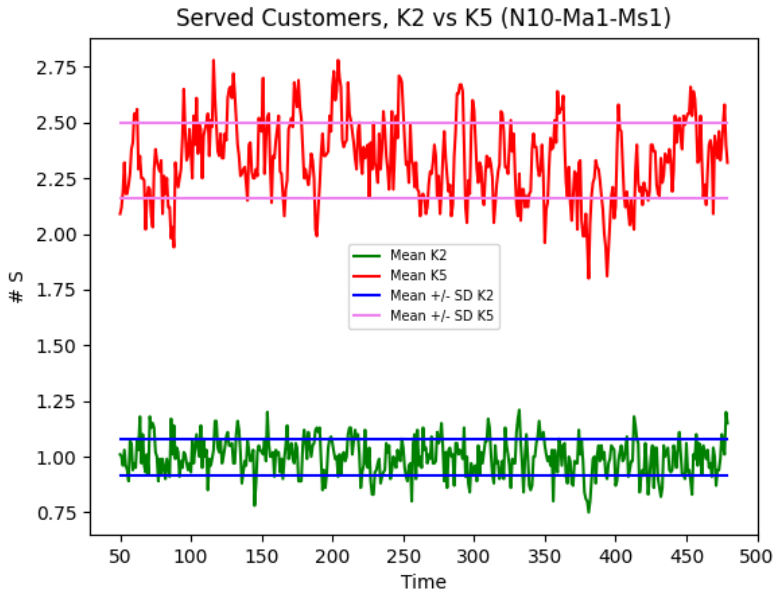

Figure 4.9: Chart of served customers, K=2 vs K=5

From the above figure (4.9) we can see that the number of served customer increases when the parameter k increases: the behavior is so obvious because the more available clerks, the more customers can be served, spending less time waiting.

Also in the chart 4.10 we can note that the number of served customer increases: indeed when the parameter Ms increases (time needed to serve a customer), it means that we can find more species of type S at the same time because a clerk needs more time to serve a client.

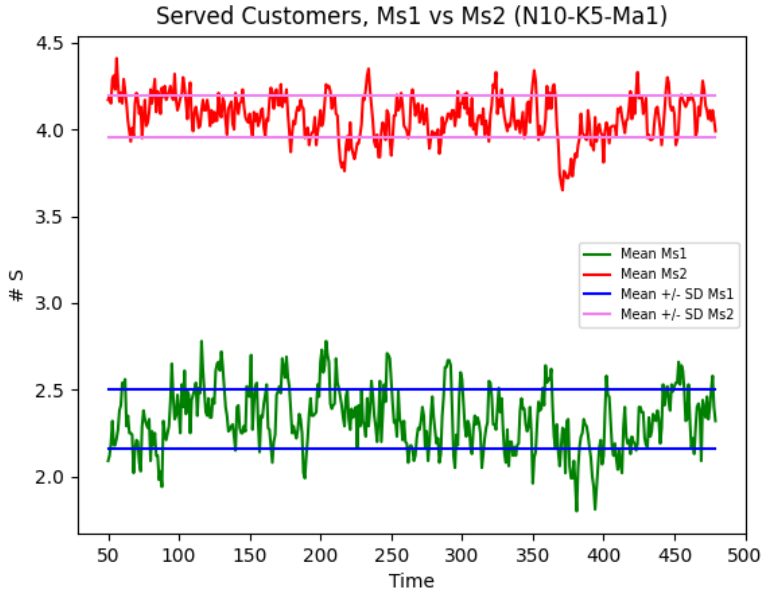

Figure 4.10: Chart of served customers, Ms=1 vs Ms=2

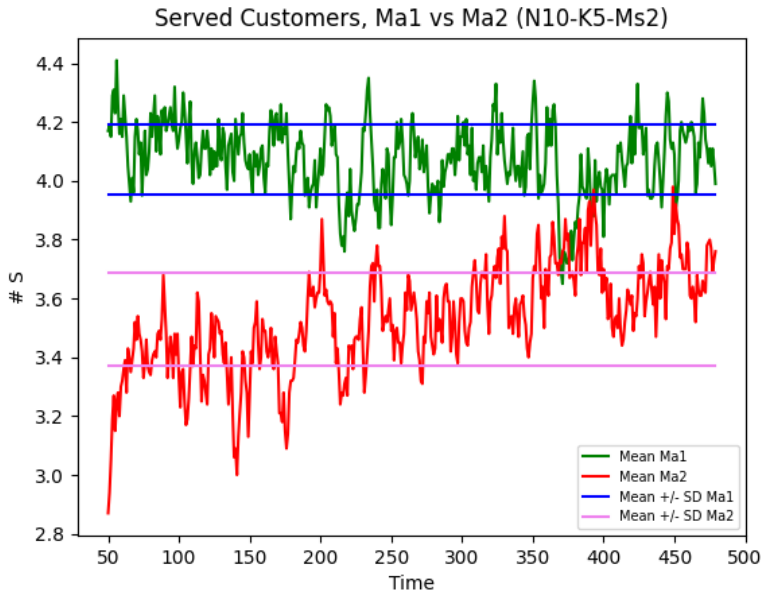

Figure 4.11: Chart of served customers, Ma=1 vs Ma=2

Whereas from the chart 4.11 we can observe that the number of served customer decreases when the parameter Ma increases (time needed for a new customer arrival): also here it is the right behaviour because it means that we can find less species of type A and W at the same time, since Ma affect the rate of the rule that allows a customer to leave the state F (far from the shop).

Considering the scenario I mentioned in the section 3.2 the number of served customer is not affected by parameter N and for the sake of simplicity I didn't

show the related chart because the lines are quite overlapped.

It is worth to notice that for these first analysis I also reported in the previous charts the information about the standard deviation giving us the information about how much the values of the distribution are close to or differ from the mean. In particular I showed the range given by the mean plus one standard deviation and the mean minus one standard deviation: in this way one can also see how many values fall in this range to better understand the distribution of each scenario (on the average 65-70% of the values, and that's a similarity with the Normal Distribution where 68% of them are within one standard deviation away from the mean). In our case, since the sample size is 100, we can assume that the sampling distribution of the sample mean is normally distributed according to the Central Limit Theorem. Moreover, the values of the standard deviation in each scenario are quite low and close to zero so I can say that the performed simulations are pretty reliable.

I would only highlights the fact that we have the highest values of standard deviation when we increase the parameter *Ma* or we decrease *Ms*. In my opinion, this is due to the less number of customer at the beginning in the case of *Ma*, because the values take more time before remaining stable. While in case of *Ms* depends on how fast a clerk serves a customer giving him the possibility to move faster from one state to another so to have maybe more different values per time unit on each simulations.

4. **Clerks utilization**

Here I used the data in the file containing the information about the number of species S representing the customers who are currently served because, at the same time, it gives us information about the number of working clerks (of course we can also use the data produced by the species C).

The utilization, expressed as a percentage, is calculated as follows:

$$Utilization(\%) = \frac{Number\ of\ working\ clerks\ (\ \#S\ )}{Total\ number\ of\ clerks\ (\ K\ )}\ \text{x } 100$$

For each scenario I obtained the following charts:

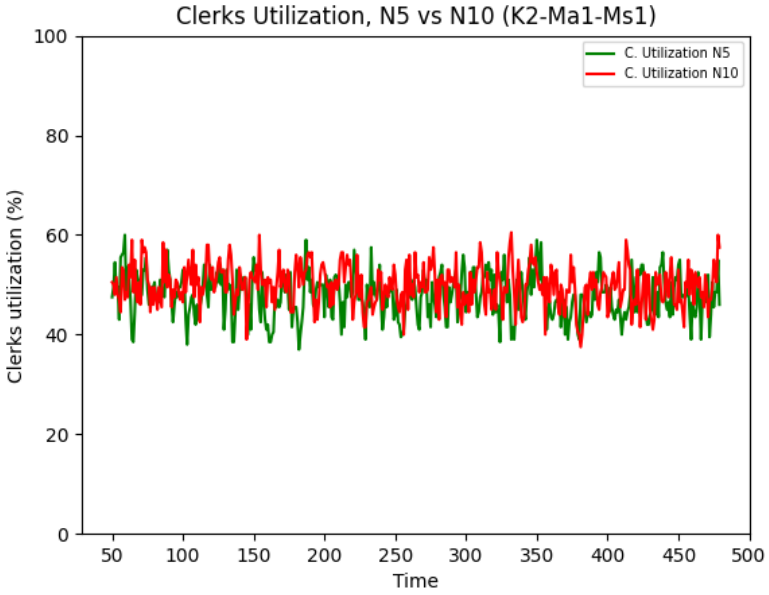

Figure 4.12: Chart of clerks utilization, N=5 vs N=10

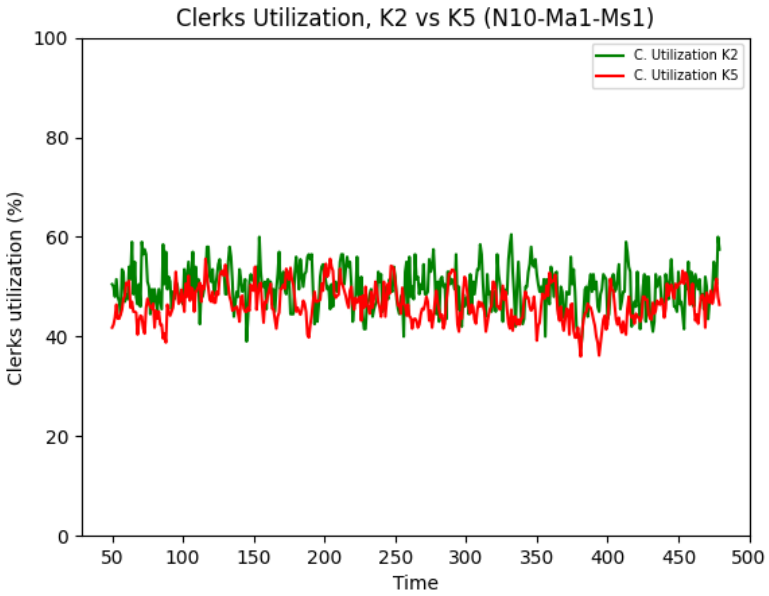

Figure 4.13: Chart of clerks utilization, K=2 vs K=5

From the charts 4.12 and 4.13 we can see that the clerks utilization (about 50%) does not change considering those parameters. This is maybe due to the fact that the clerks serve the customers quite faster ($Ms = 1$) so we find more species S available per time unit.

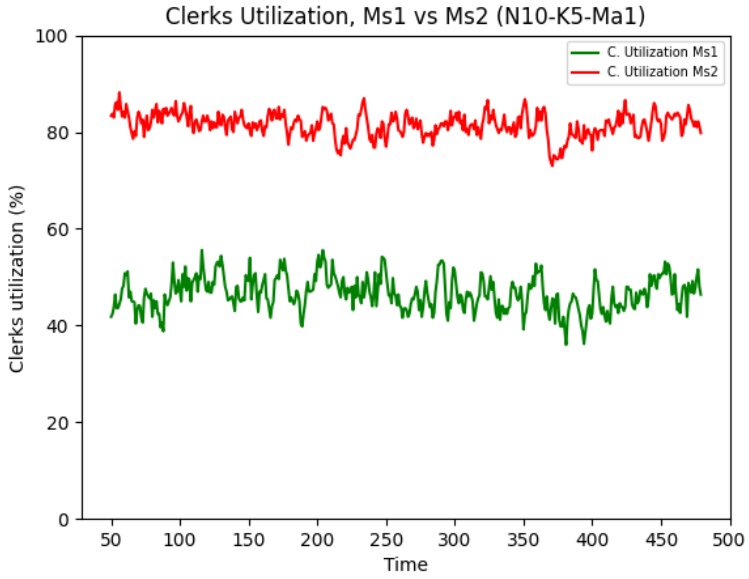

Figure 4.14: Chart of clerks utilization, Ms=1 vs Ms=2

Rather, in the chart 4.14 we can note that the clerks utilization increases when the parameter *Ms* also increases: this reflect the expectation indeed it means that a clerk needs more time to serve a customer so we can have more species of type S at the same time per time unit (more busy clerks).

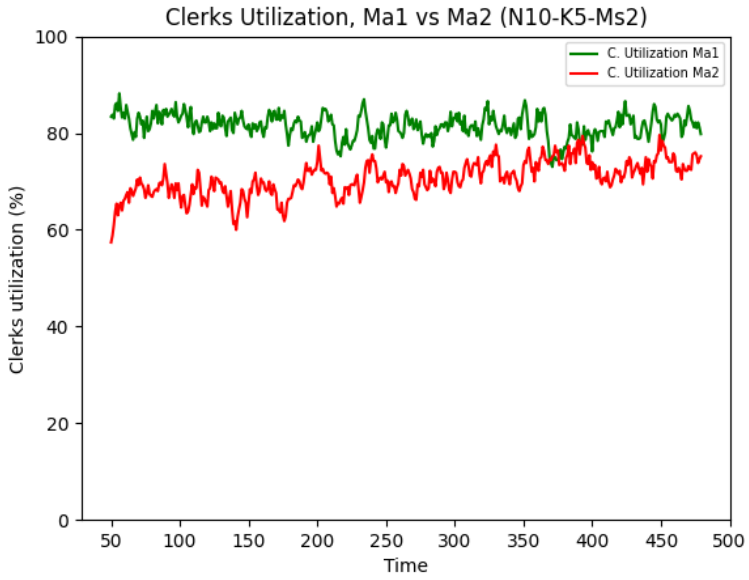

Figure 4.15: Chart of clerks utilization, Ma=1 vs Ma=2

Finally, in the last chart 4.15 we can observe that the clerks utilization decreases when the parameter *Ma* increases: this is the correct behaviour because it means that a customer need more time to arrive so we can have less species of type A and W and of course less busy clerks at the same time per time unit.

Conclusion

After having thoroughly described the work done so far, I want to sum up the results and give some final impressions.

First of all, as required in the project description, I used the framework *Sibilla* to model a system that simulates the flow of the customers in a shop considering a working day.

Once implemented the model with its parameters, species, rules and measures I performed several simulations taking into account five different scenario to see how the system behaves changing each time one single parameter.

Then the data produced by each simulations are used to get the required analysis:

- clerks utilisation;
- average of served customers;
- average of waiting customers;
- average of customers that cannot enter.

Several charts are then provided in order to show the required values, their distribution and the fact that most of the expectations are met. The obtained results can be summarized in the following table:

N	K	Ma	Ms	Mean # W (*)	Mean # E (*)	Mean # S (*)	Utilization
5	2	1	1	2.76 (71%)	13.43 (65%)	0.96 (63%)	48%
10	2	1	1	7.36 (61%)	8.26 (66%)	0.99 (67%)	50%
10	5	1	1	4.57 (69%)	6.57 (75%)	2.33 (66%)	46%
10	5	1	2	4.98 (67%)	9.04 (70%)	4.07 (69%)	81%
10	5	2	2	3.42 (51%)	2.68 (46%)	3.53 (62%)	70%

Table 5.1: Results of the analysis for each considered scenario. The *(*)* represents how many values of the distribution are within one standard deviation away from the mean)

To conclude the report I can say that this project was useful to put into practise what I have studied during the course of PAS and in particular the use of new tools is always challenging and allows you to gain expertise. In general I'm very pleased with the entire course held by professor Michele Loreti indeed, besides being interesting for the covered topics, it blends well with the other courses of the curriculum IAS that I've chosen.