Analysis of an Italian Emergency Department

Process Mining project A.A. 22/23

Filippo Fantinato

March 27, 2023

Contents

List of Figures

Listings

1 Question 1 - Process Model

The hospital wants to gain further insight into how the process is executed. Suppose you are a process analyst. You want to provide a Petri net that represents the actual process executions. Draw a Petri net and validate it by checking its conformance using alignments against the event log. If the alignment shows that the model is not satisfactory, iteratively improve the model until the quality of satisfactory. The model is deemed satisfactory if the fitness value is at least 0.85 and the precision value is at least 0.68.

1.1 Answer

In Figure 1 you can see the Petri Net I came up with after having analyzed the description of the process.

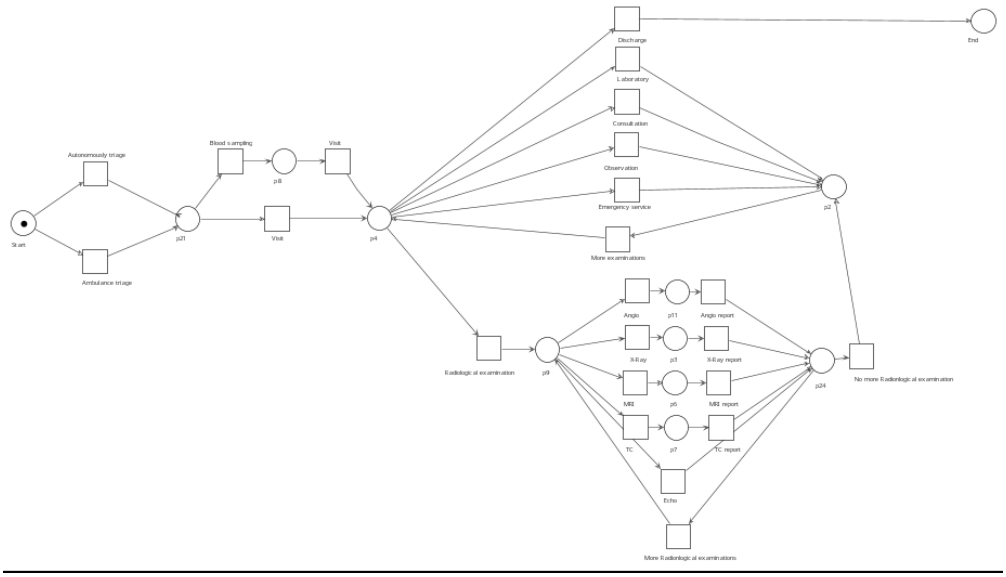

Figure 1: Drawn Petri Net

The process starts with a xor split which describes if the patient goes in through either Autonomously triage or Ambulance triage. Then, a patient can be either immediately visited or a blood sample could be taken just before being visited.

Now, the physician can decide to discharge the patient immediately or decide to subject it to Observation, Laboratory test, Emergency Service, Consultation activities or Radiological examinations, in any order and any number of steps. Such portion of the process has been modeled as a loop where each activity can be repeated 0 or more times and via xor-split, since a patient cannot be subjected to multiple examinations at the same time. The exit point from the loop is the Discharge transition, which leads to the end of the process.

The Radiological examinations consists in executing Angiography, Echocardiogram test, Magnetic Resonance Imagining, Computed Tomography (TC) or Radiographic testing (X-Ray), in any order and any number of steps, ending up having a report for each executed activity expecting for Echo test. Such portion of the

process has been modeled as a loop where each activity can be repeated 1 or more times and via xor-split, since a patient cannot be subjected to multiple Radiological examinations at the same time. To model the loops I introduced some transitions which aren't in the original description of the process, so they will be set to invisible during the computation of the alignment.

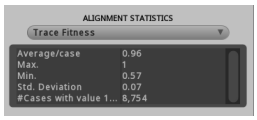

Figure 2: Drawn Petri Net fitness

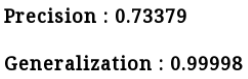

Figure 3: Drawn Petri Net precision

In order to obtain the fitness and precision values I have exploited, respectively, the ProMLite plugins *Replay a log on Petri Net for Conformance Analysis* and *Measure Precision/Generalization*. The first plugin's been used with the Petri Net and the log, while the second one's been used with the Petri Net, the log and the replay plugin result. The resulting values are 0.96 for fitness and 0.73 for precision, also available in Figure 2 and Figure 3

The parameters to assume the model is satisfactory are at least 0.85, regarding fitness, and 0.68, regarding precision, therefore I can state my model is acceptable.

2 Question 2 - Process Discovery

(a) Discover process models for different triage colors, and compare whether or not significant differences are observed in the models of different triage colors.

(b) Verify whether the hospital treats patients in accordance with the indicators of Table 1. If some patients are not treated according to Table 1, find what their percentage is.

Triage Color	**KPI Value**
Red	10min
Yellow	20min
Green	60min
Blue	90min
White	120min

Table 1: Key Performance Indicator values for each color

2.1 Answer (a)

Before starting the process discovery, I analyzed the log exploiting the library pm4py and it turns out that there a few traces which don't start with a triage and many which don't end by discharging a patient.

```
>>> pm4py.get_start_activities(log)
'AUTONOMOUSLY TRIAGE': 8079,
'AMBULANCE TRIAGE': 3378,
'VISIT': 3,
'EMERGENCY SERVICE': 8
```

Listing 1: Initial activities counting

```
>>> pm4py.get_end_activities(log)
'DISCHARGE': 10019,
'CONSULTATION': 427,
'LABORATORY': 619,
'X-RAY REPORT': 244,
'VISIT': 11,
'TC REPORT': 49,
'EMERGENCY SERVICE': 86,
'ANGIO REPORT': 8,
'ECHO': 5
```

Listing 2: Final activities counting

In order to have a clearer process discovery, I decided to throw away only the traces that don't start with a triage since there are few, while keeping the ones that end in the wrong way since there are many. Such filtering process was executed with the *Filter Log using Simple Heuristics* plugin available in ProMLite software.

I discovered the process models for different triage colors by: filtering the log for each color via the ProMLite plugin *Filter Log on Event Attribute Value*, obtaining the PetriNet model trough *Inductive visual miner* and applying *Multi-perspective Process Explorer* with the MPE Mode set on *Show Performance Mode*, in order to have also the percentage of how many times an activity is performed. Let's analyze every log and see whether there are significant differences or not.

Red log From the Red log it turns out the majority of patients in serious condition enter through Ambulance Triage, while very few people go for Autonomously Triage. The activities executed more often are: Laboratory, Consultation, Emergency service and some Radiological examinations, i.e. X-Ray, Angiography and TC. The discovered model (Figure 4) is very messy and with many activities are performed. For sure that's connected to the fact that red triage color patients are in dangerous conditions and they need many examinations before being discharged.

Yellow log Regarding the Yellow log we can see the entrances are almost equally spread over the two triage. The most performed activities are Visit, Emergency service, Laboratory, Consultation and some Radiological examinations, i.e. X-Ray, Tc, Angio and Echo. Moreover, Blood sampling is a very frequent

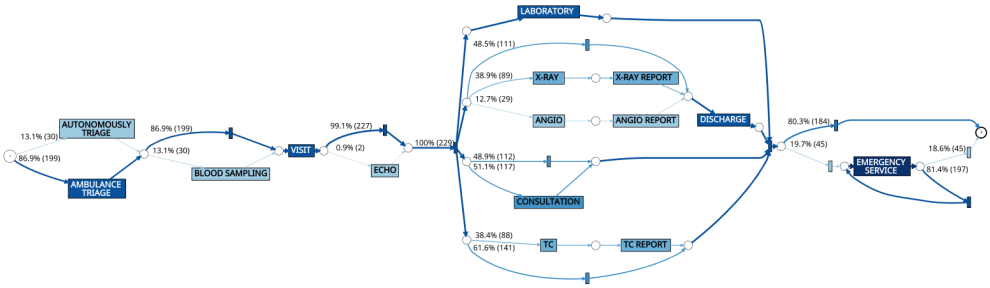

Figure 4: Process discovery for red triage color

activity in yellow triage color. The discovered model (Figure 5) is similar to the one of red triage color, indeed it's very messy with many activities performed.

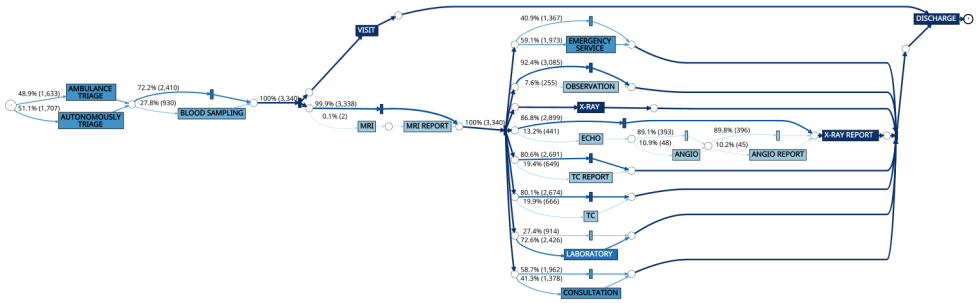

Figure 5: Process discovery for yellow triage color

Green log From the Green log we can see the majority of patients enter through Autonomously Triage, but also that a considerable number of them go for Ambulance Triage. The activities executed more often Emergency service, Laboratory, Consultation and some Radiological examinations, i.e. TC, Angio and X-Ray. Moreover, patients could be kept in observation before starting the other activities. The discovered model (Figure 6) is still messy, but not as messy as the previous ones. An important aspect to pay attention is the higher number of patients that enters via Autonomously Triage. That could be because green triage color is assigned to patients with with unimpaired vital functions, so they didn't need to call an ambulance to be helped.

Blue log Looking at the Blue log it turns out most people enter via Autonomously Triage, while few people go for Ambulance Triage. The activities executed more often are Visit, Emergency Service, Consultation and some Radiological examinations, i.e. Echo and X-Ray. Very occasionally, a blood sampling can be requested and a patient is kept under observation. The discovered model (Figure 7) is simpler and with less performed activities than previous ones.

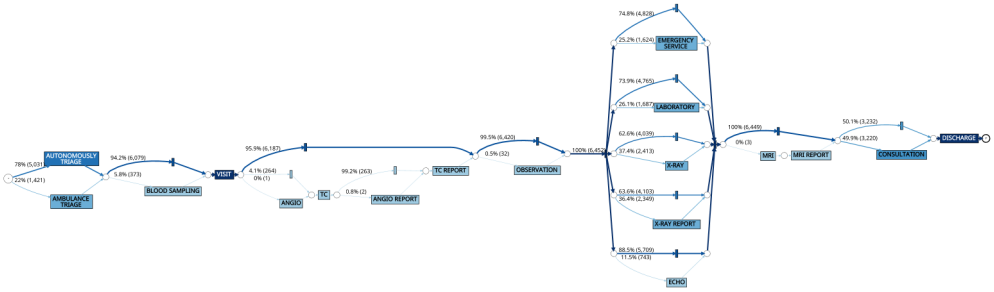

Figure 6: Process discovery for green triage color

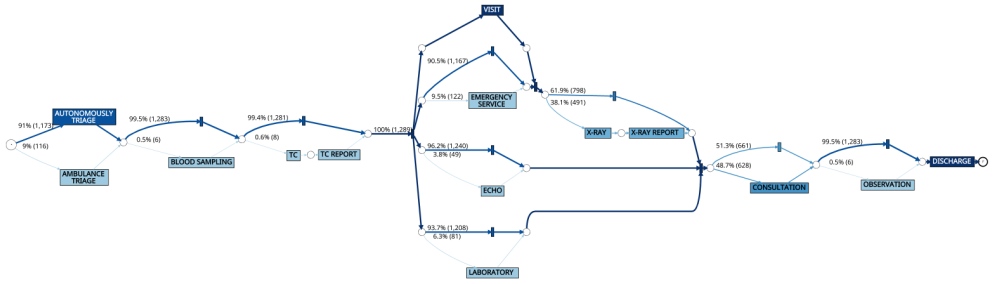

Figure 7: Process discovery for blue triage color

White log Regarding the White log we can see that almost all people enter via Autonomously Triage and very few people go for Ambulance Triage. The most executed activity is Visit, with some rare Emergency service. If it's needed, patients go through some Radiological examinations, mostly X-Ray, and Consultation. The discovered model (Figure 8) is much more simpler and the number of performed activities is lower than previous ones. That's for sure because white triage color is assigned to patients which no emergency, so after being visited and some examinations, they can be discharged.

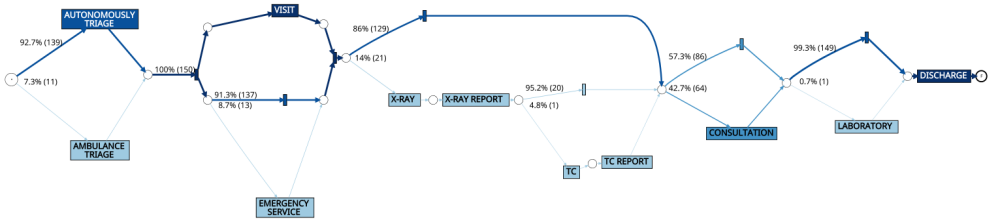

Figure 8: Process discovery for white triage color

Summing up, there are relevant differences among the furthest triage colors, such as red and white, but no for the closest ones, such as blue and white or red and yellow. Moreover, you can also notice that the discovered model is more messed up if the color corresponds to a higher urgency level. Indeed, the triage colors red and yellow are very messy with many activities, compared to the white or blue colors.

A further comparison is to analyze the distribution of events over the hours of the day for each color (Figure 9) in order to see when the ED works more according to each triage color. It's easy to notice that from 00:00am to 8:00am the number of events decreases for each color, but in particular for the blue and white ones. That's reasonable since the blue and white triage colors are the ones with the highest percentage of autonomously triage and people, even if they feel very bad but not too bad to call an ambulance, tend to wait for the morning. From 8:00am to 11:59pm the distributions look very similar, excepting for yellow and green colors where the number of events is always very high, while in red, blue and white ones the number of events decreases at the end of afternoon.

Such distributions were obtained calling the method in pm4py `pm4py.view_events_distribution_graph(log_[COLOR], distr_type="hours")` for each color log.

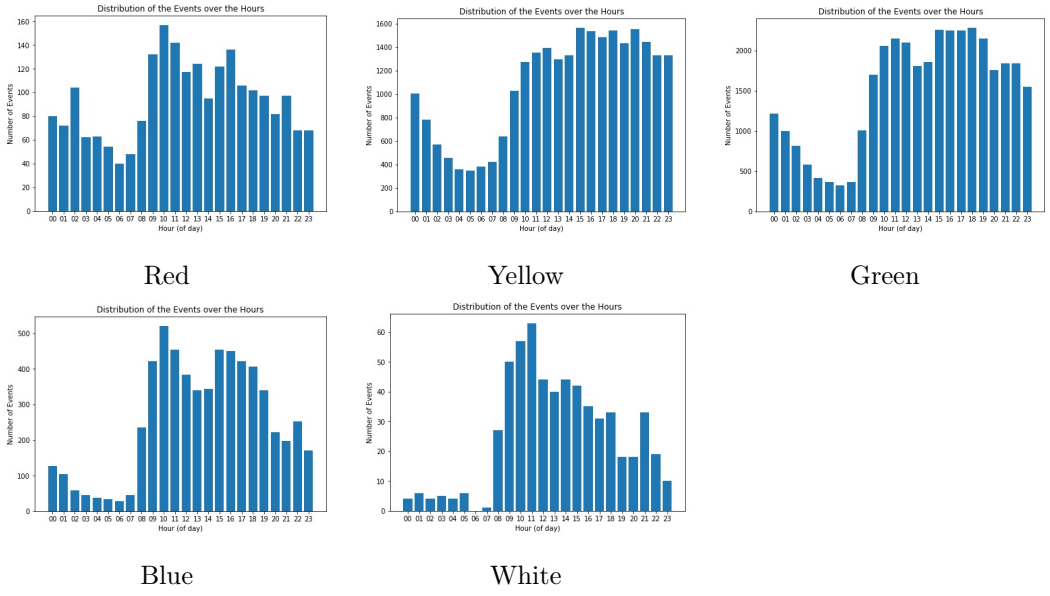

Figure 9: Distribution of events over the hours of the day for each triage color

2.2 Answer (b)

The Key Performance Indicator (KPI) is defined as the time elapsed between the triage and the beginning of the patient's treatment, therefore what we have to do is to divide up the log in one log for each triage color and then measure whether KPI of each trace is conforming to the requirements in Table 1. To cope with this question I used only pm4py.

First of all, let's filter out only the traces which start with a triage activity. I could have used the filtered log obtained in the previous answer, but doing everything in pm4py is easier. Then, I split the filtered log via `pm4py.filter_event_attribute_values` and applied a custom method named `percentage_non_compliant_cases/2`,

which given a log and a number corresponding to minutes, it checks how many traces have KPI higher than the number of minutes in input. The code snippets show how to do it for the red triage color, but you can easily adapt them for other colors by changing the value of values parameter of `pm4py.filter_event_attribute_values`. Here below the percentage of non compliant cases regarding the red triage color:

```
filtered_log = pm4py.filter_start_activities(log, ['AUTONOMOUSLY TRIAGE', 'AMBULANCE TRIAGE'])
log_red = pm4py.filter_event_attribute_values(filtered_log, "triage color", values=["RED"], level='event')
round(percentage_non_compliant_cases(log_red, time=10), 2)
```

Listing 3: Percentage of non compliant cases by color

To be more precise, the method `percentage_non_compliant_cases/2` is defined as follows:

```
def percentage_non_compliant_cases(log, time):
    non_compliant_cases = 0
    for trace in log:
        arrival_time = trace[0]['time:timestamp']
        response_time = trace[1]['start:timestamp']
        diff_minutes = (response_time - arrival_time).total_seconds() // 60
        if diff_minutes > time: non_compliant_cases += 1
    return non_compliant_cases / len(log) * 100
```

Listing 4: `percentage_non_compliant_cases` method

As you can see, the KPI is computed taking the difference between the end time of the first event, which in our case is for sure a triage activity since we are using the filtered log, and the start time of the next one. Such difference is converted in minutes and compared with the number of minutes given input. If it's higher, the counter `non_compliant_cases` increases by 1. At the end, the method returns the percentage of non compliant cases.

After having repeated the above code for each triage color, you can see the results in Table 2.

Triage Color	**Non compliant cases (%)**
Red	17.47%
Yellow	19.73%
Green	21.17%
Blue	23.04%
White	18.67%

Table 2: Percentage of non compliant cases for each triage color

3 Question 3 - Process Simulation

The hospital is unsatisfied with the costs: there are certainly too many resources at the ED. Use Process Simulation to reduce the number of resources while keeping similar case duration. First, build a simulation model in BIMP that defines case arrival rate, resource roles, task duration, branching probabilities. Then, work with different resources allocated per role and reduce them in number while keeping similar case duration.

3.1 Answer

To compute all the parameters that we need to build a simulation model in BIMP, we can use both pm4py and ProMLite. In particular, ProMLite is useful just to compute the branching probabilities while pm4py for getting the other parameters.

Case arrival rate The case arrival rate is defined as the mean among the time passed between one trace and its next of the same day. It can be computed through the following method, which is very simple and intuitive:

```
def get_case_arrival_rate(log):
    case_arrival = []
    for i in range(1, len(log)):
      arrive1 = log[i-1][0]['start:timestamp']
      arrive2 = log[i][0]['start:timestamp']
      if arrive1.date() == arrive2.date():
        case_arrival.append(arrive2 - arrive1)
    return np.mean(case_arrival)
```

Listing 5: `get_case_arrival_rate` method

In our case, the case arrival rate is 12 minutes.

Resource roles Table 3 shows the discovered organizational roles computed exploiting the method `pm4py.discover_organizational_roles(log)` in pm4py.

Activities distribution In order to build a simulation, we need to find out the distribution for each activity and that's possible through the following code:

```
def plot_waiting_distribution(log, activity):
    waitings = []
    for trace in log:
        for event in trace:
            if event['concept:name'] == activity:
                start = event['start:timestamp']
                end = event['time:timestamp']
```

Role Name	Activities	Quantity
Role1	[AMBULANCE TRIAGE]	1
Role2	[ANGIO REPORT]	36
Role3	[ANGIO, TC, X-RAY]	27
Role4	[AUTONOMOUSLY TRIAGE]	1
Role5	[BLOOD SAMPLING, EMERGENCY SERVICE]	30
Role6	[CONSULTATION]	34
Role7	[DISCHARGE, VISIT]	15
Role8	[ECHO, TC REPORT, X-RAY REPORT]	39
Role9	[LABORATORY]	96
Role10	[MRI]	5
Role11	[MRI REPORT]	5
Role12	[OBSERVATION]	12

Table 3: Discovered organizational roles

```
                time = end - start
                waitings.append(time.seconds // 60)
    mean = np.mean(waitings).round(2)
    std = np.std(waitings).round(2)
    var = np.var(waitings).round(2)
    min = np.min(waitings)
    max = np.max(waitings)
    title = "Avg " + str(mean) + ", Std: " + str(std) + ", Var " + str(var) + ", Min " + str
    (min) + ", Max " + str(max)
    plt.figure(figsize = (10, 5))
    plt.suptitle(activity)
    plt.title(title)
    plt.hist(waitings, bins = 50)
    plt.axvline(np.mean(waitings), c='red', label='avg')
    plt.legend()
    plt.xlabel("Minutes")
    plt.ylabel("Frequence")
    plt.show()

data_table = pm4py.convert_to_dataframe(log)
activities = list(sorted(set(data_table['concept:name'])))
for activity in activities:
  plot_waiting_distribution(log, activity)
```

Listing 6: `plot_waiting_distribution` method

which plots the distribution of how many times x minutes are required to execute an activity. Moreover, some useful information is attached to the plot, like mean, standard deviation, variance and min and max values of the distributions in minutes. Table 4 shows the distribution of each activity with the parameters for that particular distribution.

Activity	**Distribution**	**Parameter values (min.)**
AMBULANCE TRIAGE	Fixed	value: 0
ANGIO	LogNormal	Mean: 70.82, Var: 9389.68
ANGIO REPORT	Normal	Mean: 7.47, Std: 5.64
AUTONOMOUSLY TRIAGE	Fixed	value: 0
BLOOD SAMPLING	LogNormal	Mean: 25.28, Var: 973.52
CONSULTATION	Exp	Mean: 35.66
DISCHARGE	Exp	Mean: 1.43
ECHO	Exp	Mean: 16.81
EMERGENCY SERVICE	Exp	Mean: 6.95
LABORATORY	LogNormal	Mean: 83.73, Var: 5919.61
MRI	Uniform	Min: 10, Max: 200
MRI REPORT	Uniform	Min: 0, Max: 1
OBSERVATION	Uniform	Min: 0, Max: 1400
TC	Exp	Mean: 18.05
TC REPORT	Exp	Mean: 5.1
VISIT	Exp	Mean: 4.62
X-RAY	Exp	Mean: 21.06
X-RAY REPORT	Exp	Mean: 0.47

Table 4: Activities distribution

Branching probabilities Regarding branching probabilities, I used the ProMLite plugin *Multi-Perspective Process explorer* applied on the PetriNet that I drew and on the original log. In Figure 10 you can see the result obtained by computing alignment and setting the three measures to: Percentage (local), Frequency and Frequency.

After having got all information, I generated the BPMN diagram exploiting the ProMLite plugin *Convert PetriNet to BPMN diagram* and uploaded it on BIMP filling any field with the needed data. Moreover, I chose to use an exponential distribution for Inter arrival time with 5% of tail. Figure 11 shows what I got running the simulation and it's clear that there are too many people working, since the waiting times is 0 but the resource utilization is very low.

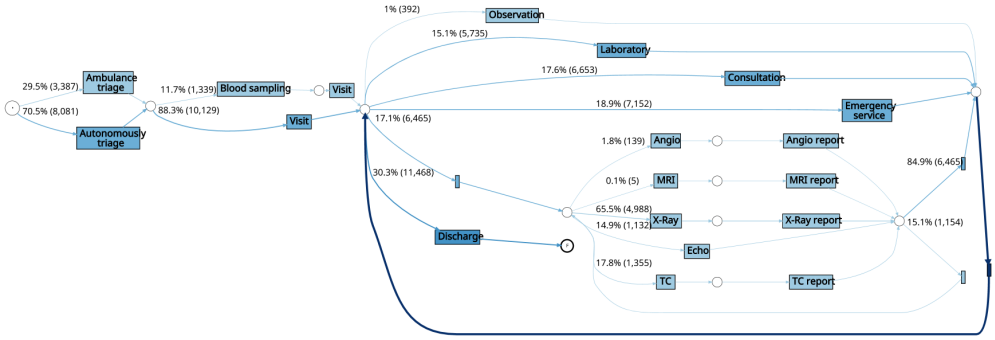

Figure 10: Branching probabilities diagram

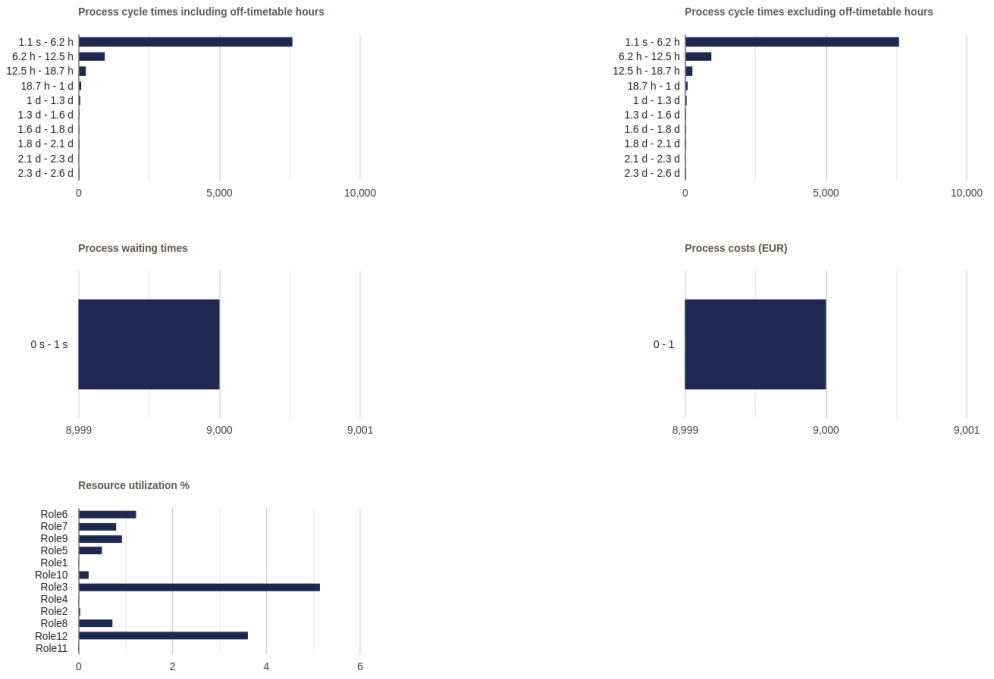

Figure 11: Original BIMP Simulation

At first glance to resources, you can notice that Role2, Role5, Role6, Role8 and Role9 are in a very high number, so first of all I cut them very aggressively and then I continued to reduce a little for each role until the wait time was no longer 0. After many tries, in my opinion the following scenario is the best one since it reduces the resources as much as possible keeping the waiting time to 0 and raising a lot the resource utilization. Table 5 shows the number of resources for each role and Figure 12 shows what I got playing the simulation. In the original simulation there was 301 resources, while in this scenario I reduced them to 74, with a cut of 75.42%.

Role Name	**Quantity**
Role1	1
Role2	3
Role3	16
Role4	1
Role5	8
Role6	7
Role7	6
Role8	8
Role9	13
Role10	3
Role11	2
Role12	6
Total number	**74**

Table 5: Number of resources for the improved scenario

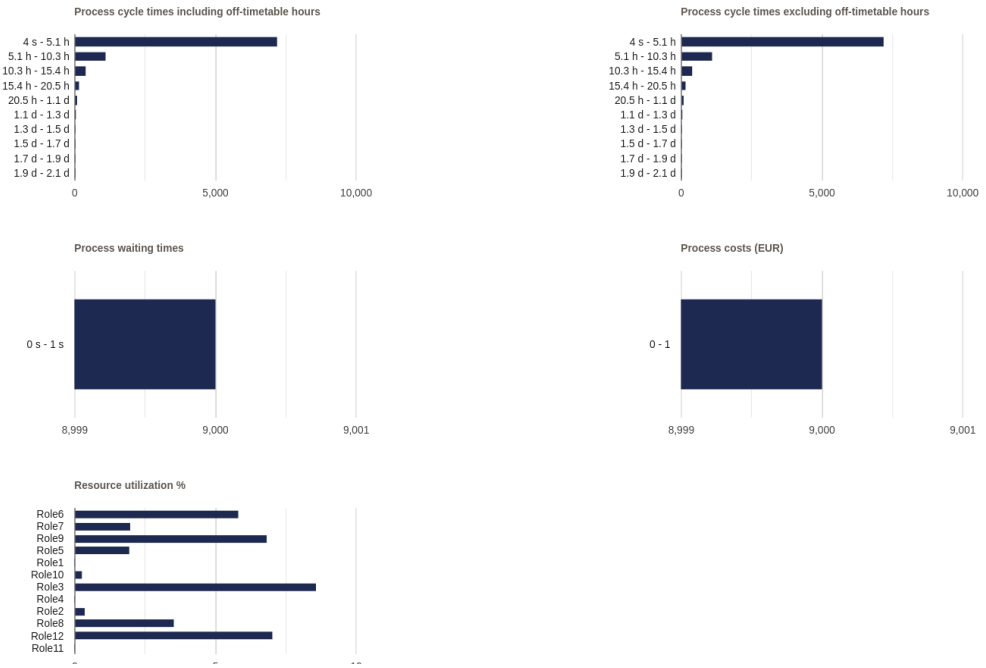

Figure 12: Improved BIMP Simulation