Politecnico di Torino

A.Y. 2024/25

Optimization and Simulation for Industrial Automation

Profs. Alberto Tarable, Marco Ghirardi

Ali Zein Khalifeh: s324962

Ali AlRida Farhat: s331680

Project Description

A small surgery unit is equipped for two kinds of operations: appendectomy and tonsillectomy. Each operation requires three stages:

- The patient is first anesthetized in one of the m1 stage-1 rooms;
- Then, the operation is performed, in one of the m2 stage-2 rooms;
- Finally, the patient recovers from the anesthesia in one of the m3 stage-3 rooms.

All the rooms can contain a single patient at a time, and all the rooms at stage i, i = 1, 2, 3, have the same equipment. A patient that needs to undergo stage i will be assigned to any of the free stage-I rooms. If there is no free stage-i room, for i = 2, 3, the patient will wait in the stage-(i − 1) room in which it is. If there is no free stage-1 room, the patient will wait in a very large (practically unlimited) waiting room.

The time needed for the three stages is an exponentially distributed random variable, with means that depend on the type of operation:

- For the appendectomy, the means (in hours) are $\tau_{A,1} = 1$, $\tau_{A,2} = 2$ and $\tau_{A,3} = 0.5$ for the three stages, respectively.
- For the tonsillectomy, analogously, the means are $\tau_{T,1} = 0.5$, $\tau_{T,2} = 2$ and $\tau_{T,3} = 1.5$, again expressed in hours.

After recovery, patients leave the surgery unit.

1. Emergency scenario :

In this scenario, the two types of patients come from outside according to a Poisson process with rates λ_A = 1 patient/hour and λ_T = 2 patients/hour (i.e., the interarrival times are exponential random variables with means 1/λ_A and 1/λ_T , respectively). Patients are operated according to their order of arrival.

1. Implement the model in Simulink, with parameters m1, m2 and m3 :

We started by the calculations of the number of rooms (m1 , m2 , m3)

For this we started by assumption that We have 3 M/M/m/ Queue systems in which we need to study the Ergodicity of each Stage as a single system using the follwing rule of the trafic intensity , which is

- **$\rho = \frac{\lambda}{m.\mu}$ & in order for the system to be Ergodic ρ must be always less than one**

We considered the following values for **λ :**

- For Stage 1 : 3
- For Stage 2 : 3
- For Stage 3 : 3

We considered the following values for μ **:**

- For Stage 1 : 1/1
- For Stage 2 : 1/2
- For Stage 3 : 1/1.5

Following the rule of ergodicity and the implemented values of the parameters stated above we get the following :

- For Stage 1 : m1 > 3
- For Stage 2 : m2 > 6
- For Stage 3 : m3 > 4.5

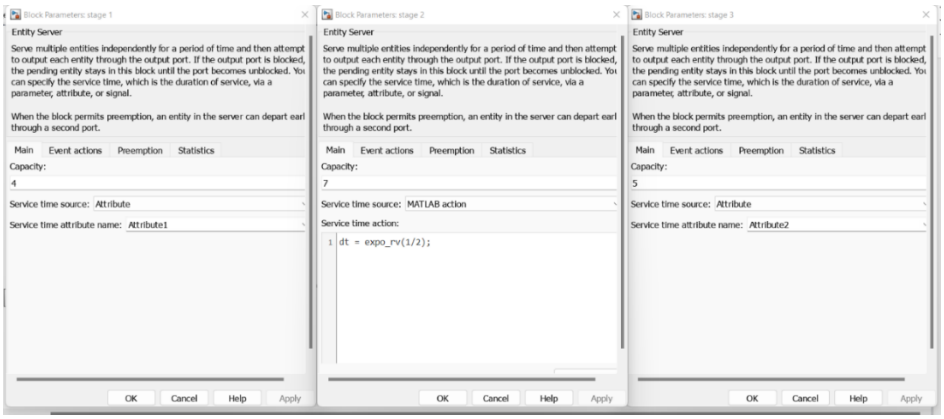

Figure 1: entity servers

Below is the implementation of the queueing system on simulink

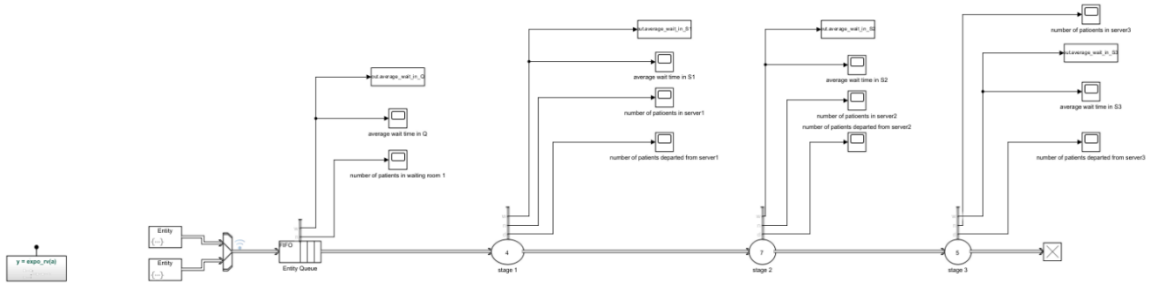

Figure 2 - Simulink Implementation

This diagram represents a Simulink model for generating exponentially distributed random variables in which is used in order to feed the Entity Generators . Here's an explanation of its components:

1. **Inputs**:
    - u: A uniform random variable (commonly generated between 0 and 1).
    - a: A parameter that defines the rate of the exponential distribution (λ\lambdaλ).
2. **Process**:
    - Ln(u): Takes the natural logarithm of u.
    - X : Multiplies ln(u) by a.
    - u: Negates the output of the multiplication to produce the desired distribution.
3. **Output**:
    - y: The resulting exponentially distributed random variable ($y=-\ln(u)/a$)

This implementation is based on the inverse transform sampling method, which uses the property that if $U\sim\text{Uniform}(0,1)$, then $-\ln(U)/\alpha\sim\text{Exponential}(\alpha)$

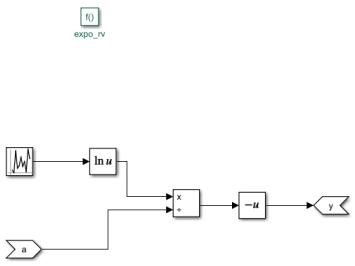

Figure 3 - Exponential Distribution

It's feed by the avergae arrival rate of Patients which is defiened by the following formula :

$$E[Y_K] = 1\ /lamda$$

Where :

- Yk : is the the interarrival time between (k-1) and (k) Patients
- Lamda : Average arrival rate of Patients

The interarrival times are exponential random variables with means 1/λA and 1/λT , respectively

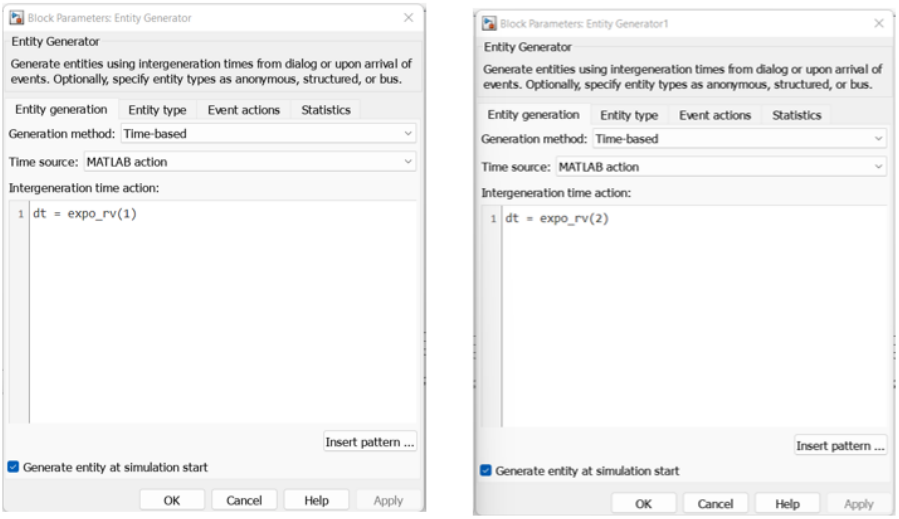

Figure 4: generators for appendectomy and tonsillectomy.

Having 3 Stages in which we have 3 Servers :

- First m1 stage : We have 2 values of the arrival rate depending on the type of patient , where those 2 values will be presented in Attribute 1
- Second m2- stage : We have the same arrival rate regardless of the type of patient so it doesnot need to be specified as Attribute in the Entity type in the Entity Generator
- Third m3 - stage : We have 2 values of the arrival rate depending on the type of patient , where those 2 values will be presented in Attribute 2

Having the means , We consider the arrival rate of Patients as reciprocal of the mean of each stage and considering which type of patient

- Attribute 1 will be used for the First Stage Server:
  - 1/1 correspondes to the appendectomy patient
  - 1/0.5 corresponds to the tonsillectomy patient
- Server 3 has the same processing rate for both patients.
- Attribute 2 will be used for the Third Stage Server:
  - 1/0.5 correspondes to the appendectomy patient
  - 1/1.5 corresponds to the tonsillectomy patient

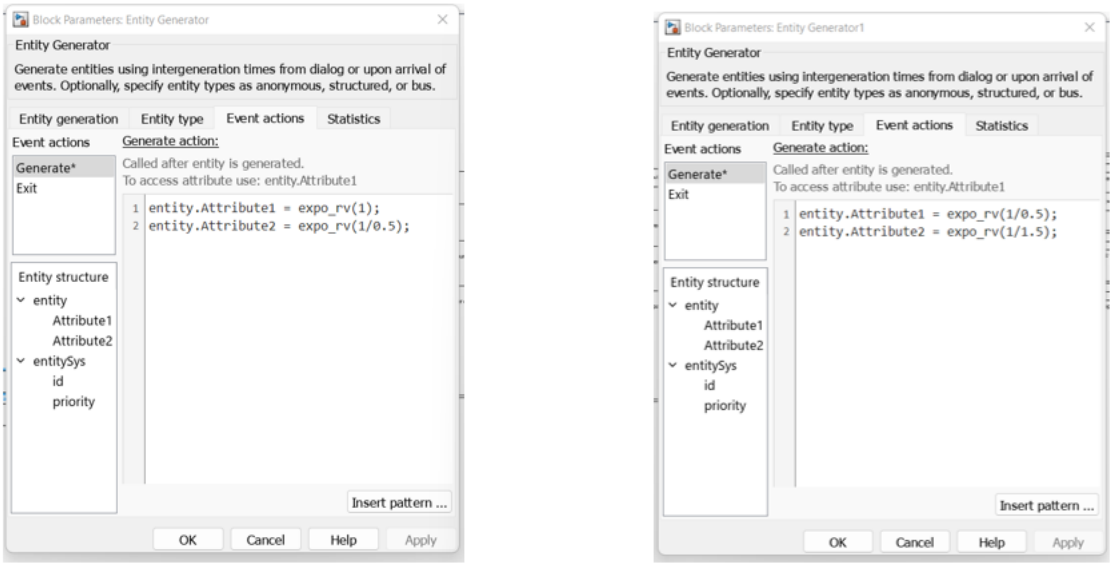

Figure 5: attributes for servers 1 & 3

The Capacity of the Queue is set to infinitly large number (infinite) because If there is no free stage-1 room, the patient will wait in a very large (practically unlimited) waiting room.

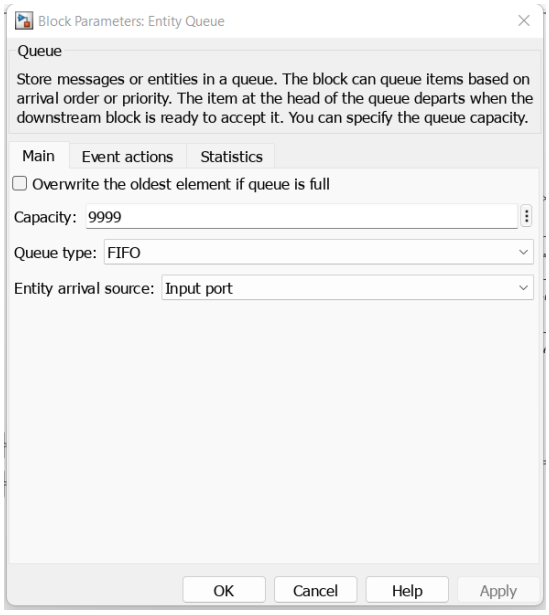

Figure 6: Entity Queue

2. For given values of the parameters, compute the average flow time SA and ST of the two types of patients separately (i.e., the time from the arrival of the patient to the hospital discharge.)

To calculate S_bar_A and S_bar_T:

We used the following method:

- S_bar_A=W_bar+ τ_A1+ τ_A2+ τ_A3 = 10.9157 hrs.
- S_bar_T=W_bar+ τ_T1+ τ_T2+ τ_T3 = 11.4157 hrs.

W_: is the average waiting time in Q + the average weight in server1 and Server 2 and Server 3

All calculations were performed using MATLAB

Ex1.m

```
%% prb2 computing SA & SB
clear all
close all
clc

out=sim("finalproj_emeregency.slx");
w1=out.average_wait_in_Q.data;
w2=out.average_wait_in_S1.data;
w3=out.average_wait_in_S2.data;
w4=out.average_wait_in_S3.data;
W = mean(w1(1:size(w4))) + mean(w2(1:size(w4))) + mean(w3(1:size(w4))) + mean(w4);
S_bar_A= W + (1 + 2 + 0.5);
S_bar_T= W + (0.5 + 2 + 1.5);

z_bar = ((1 + 2*0.5) + (2 + 2*2) + (0.5 + 1.5*2))/3
S_bar= W + z_bar;
```

Figure 7: Average flow time calculation on Matlab

3. Supposing that one stage-1 room costs 10 k€, one stage-2 room costs 50 k€ and one stage-3 room costs 2 k€, find the minimum-cost set of parameters that allow to have S_bar ≤ 20 hours, where S is the average flow time for all patients.

Trial & Error:

- We started with our initial case (m1=4, m2=7, m3=5)
- We began by decreasing m2, as it is the most expensive.
- Following the iterations done on excel we found out that the best scenario will correspond (m1=4, m2=7, m3=5)

cases	A	B	C	D	E	F	G	H	I	J	K	L	M
m1	4	4	4	3	4	5	8	3	2	1	2	2	2
m2	7	6	5	7	7	3	3	4	4	4	4	4	4
m3	5	5	5	5	4	5	5	5	5	5	4	3	2
S bar	11.25	316	>> 20	28	240	>> 20	>> 20	>> 20	>> 20	>> 20	>> 20	>> 20	>> 20
total cost	400 k	350 k	300 k	390 k	398 k	210 k	240 k	240 k	230 k	220 k	228 k	226 k	224 k

Table 1: Trials for different scenarios

2. Hospital scenario :

In this scenario, we consider a single working day and we suppose that there are N patients (N/2 appendectomy and N/2 tonsillectomy). The patients already passed a triage: at time 0, we already know the service times of the three operations for each job.

1. Implement the model in Simulink, with m1 = 1, m2 = 1, m3 = 1 and N = 20.

An Overall view of the Simulink Model

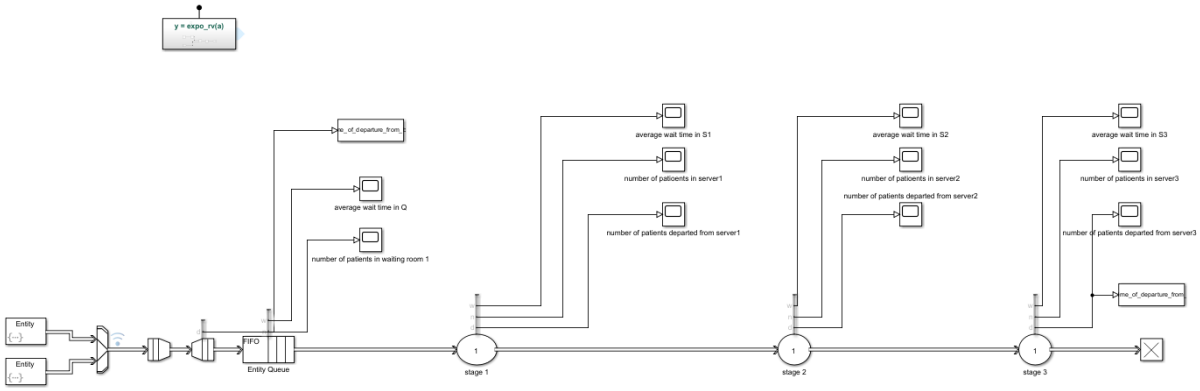

Figure 8: Simulink model for hospital scenario

Having the Entity Generator

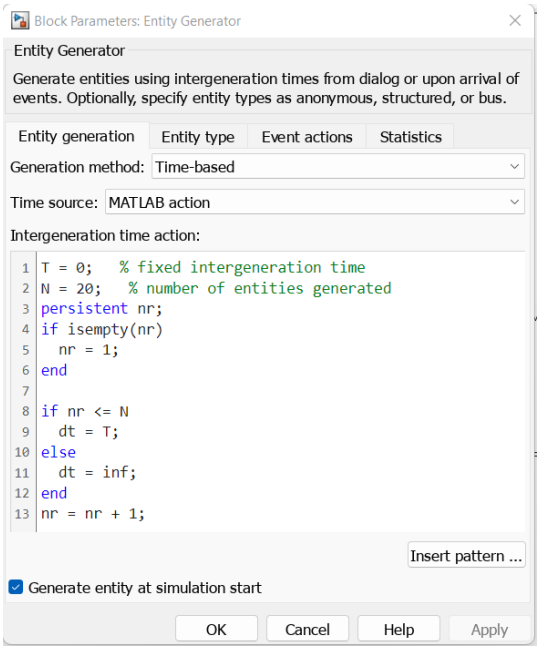

Figure 9: Entity Generator

This entity generates 20 entities at time = 0; and then we put the next entity at dt = inf. This way we are generating only one batch of 20 patients.

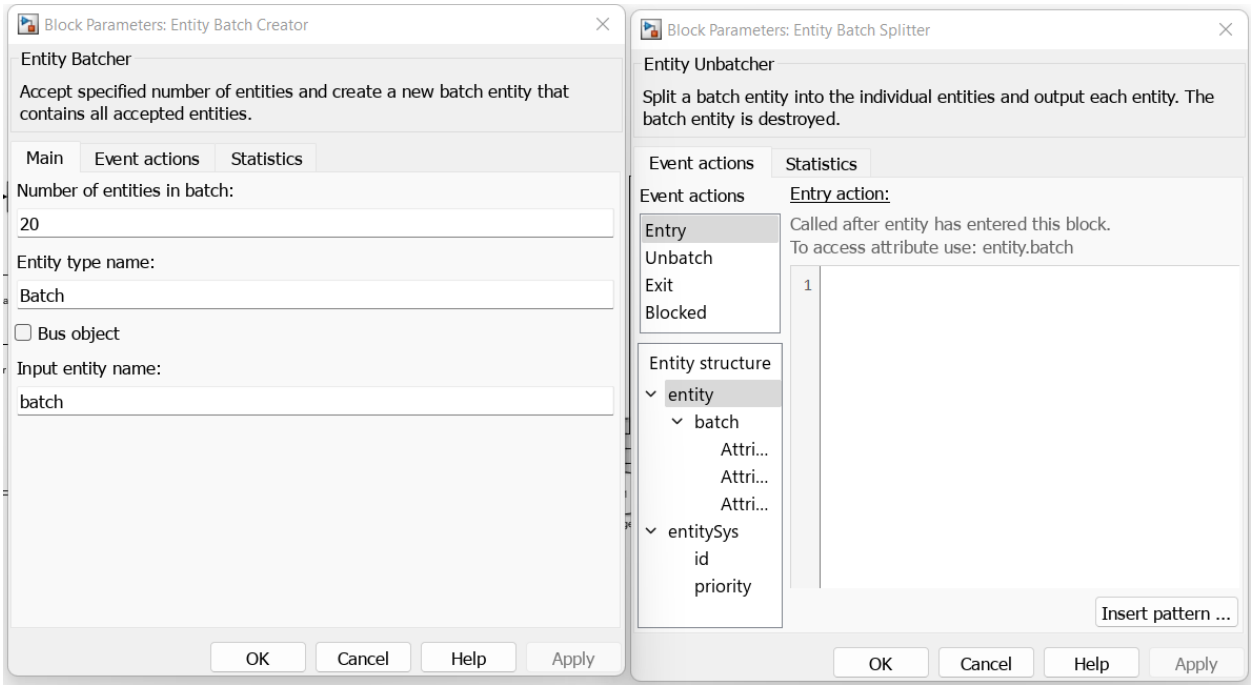

Figure 10: Batch creator and splitter

We calculated S_bar , by differentiating between times of departure from batch splitter with respect to time of departure from S3. This way, we get the process time of each patient, and then we find the average: S_bar = 26.4977 hrs.

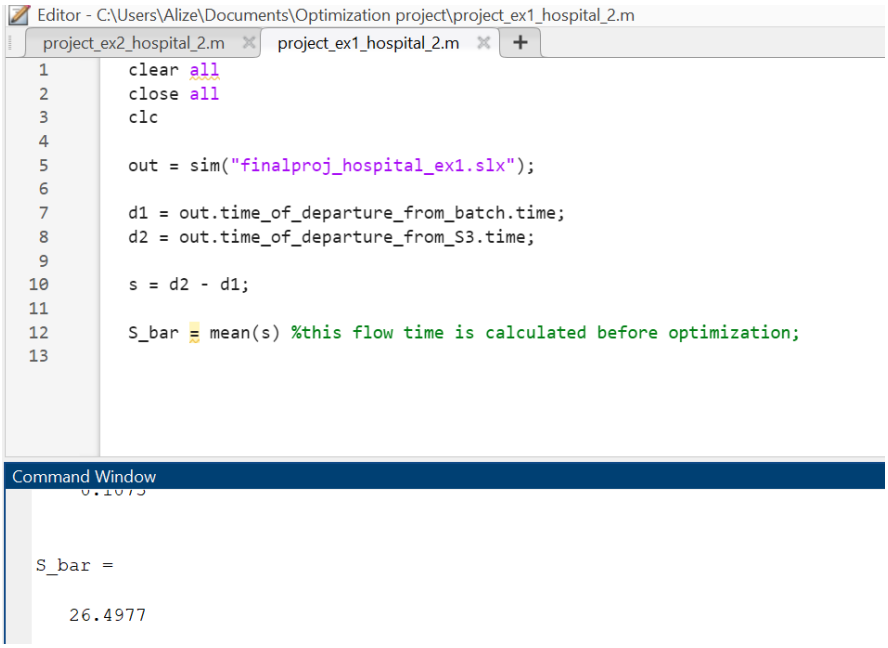

Figure 11: Average flow time calculation before optimization

2. Implement an algorithm of optimization for finding an efficient order of operation for the N patients, where the parameter to be optimized is the average flow time S

Algorithm:

- P(M,N): Matrix containing process times.
- C(M,N) : is the Completion time of job in position j belong N in machine i in M.
- X(N,N): Describes which job is in which position.

Assignments:

- X-matrix should have only one "1" in each row and column.

Sequence:

- Conditions so that each job cannot start if the next machine is working.

Release Time:

- To calculate completion times.

MATLAB OPTIMIZATION CODE

```
clear all
close all
clc

rng('default');
P(1,1:10)=exprnd(1,1,10);
P(1,11:20)=exprnd(0.5,1,10);
P(2,1:10)=exprnd(2,1,10);
P(2,11:20)=exprnd(2,1,10);
P(3,1:10)=exprnd(0.5,1,10);
P(3,11:20)=exprnd(1.5,1,10);

Pnew =
[P(:,20),P(:,19),P(:,9),P(:,4),P(:,1),P(:,16),P(:,17),P(:,7),P(:,5),P(:,11),P(:,8),P(:,3),P(:,13),
P(:,6),P(:,18),P(:,10),P(:,15),P(:,14),P(:,2),P(:,12)];
P1 = Pnew(1,:);
P2 = Pnew(2,:);
P3 = Pnew(3,:);

[M N]=size(P);

% Variables
C=optimvar('C',M,N,'Type','continuous','LowerBound',0);
x=optimvar('x',N,N,'Type','integer','LowerBound',0,'UpperBound',1);
```

```
obj = sum(C(3,:));

% Assignment constraints
for i = 1:N
Assignment1(i) = sum(x(i,:)) == 1;
Assignment2(i) = sum(x(:,i)) == 1;
end

% % Assignment constraints
%     Assignment1 = sum(x,2) == 1;
%     Assignment2 = sum(x,1) == 1;

% Release Time Constraint

RelTime01 = C(1,1) >= (P(1,:))*x(1,:)';
RelTime02 = C(2,1) >= C(1,1) + (P(2,:))*x(2,:)';
RelTime03 = C(3,1) >= C(2,1) + (P(3,:))*x(3,:)';

for j=2:N
RelTime1(j) = C(1,j) >= C(1,j-1) + (P(1,:))*x(j,:)';
RelTime2(j) = C(2,j) >= C(1,j) + (P(2,:))*x(j,:)';
RelTime3(j) = C(3,j) >= C(2,j) + (P(3,:))*x(j,:)';
end

for j=2:N
Sequence(j,1) = C(1,j) >= C(2,j-1);
Sequence(j,2) = C(2,j) >= C(3,j-1);
end

% Define Problem Structure

SingleMachineProblem=optimproblem;
SingleMachineProblem.Objective=obj;
SingleMachineProblem.Constraints.Assignment1=Assignment1;
SingleMachineProblem.Constraints.Assignment2=Assignment2;
SingleMachineProblem.Constraints.RelTime1=RelTime1;
SingleMachineProblem.Constraints.RelTime2=RelTime2;
SingleMachineProblem.Constraints.RelTime3=RelTime3;
SingleMachineProblem.Constraints.RelTime01=RelTime01;
SingleMachineProblem.Constraints.RelTime02=RelTime02;
SingleMachineProblem.Constraints.RelTime03=RelTime03;
SingleMachineProblem.Constraints.Sequence=Sequence;

% Run the LP solver

[Jobs fval]=solve(SingleMachineProblem,'solver','intlinprog');
```

After running, we get X-matrix and C-matrix as the optimal solution.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2: X_matrix

0.020675	0.137141	0.294487	1.139269	1.344175	2.320539	2.752197	4.030541	4.624303	5.548109	6.244258	8.30793	8.635018	11.20305	11.83751	14.13458	17.66459	22.32796	28.47875	35.1433
0.122795	0.294487	1.139269	1.310961	2.188155	2.709025	3.753662	4.624303	5.399351	6.244258	8.175154	8.635018	11.20305	11.83751	14.13458	17.66459	22.32796	28.47875	35.1433	42.03782
0.294487	1.098016	1.310961	1.425542	2.600074	3.753662	3.815295	5.028485	6.238015	8.175154	8.393389	8.76862	11.83751	12.19442	15.7511	17.80531	25.52094	31.20333	35.62505	42.61697

Table 3: C_matrix

3. Include the optimization algorithm in the model, running it at time 0 and then sending the patients to their operations in an optimized order. Compare the obtained S with the one obtained with random ordering of the patients.

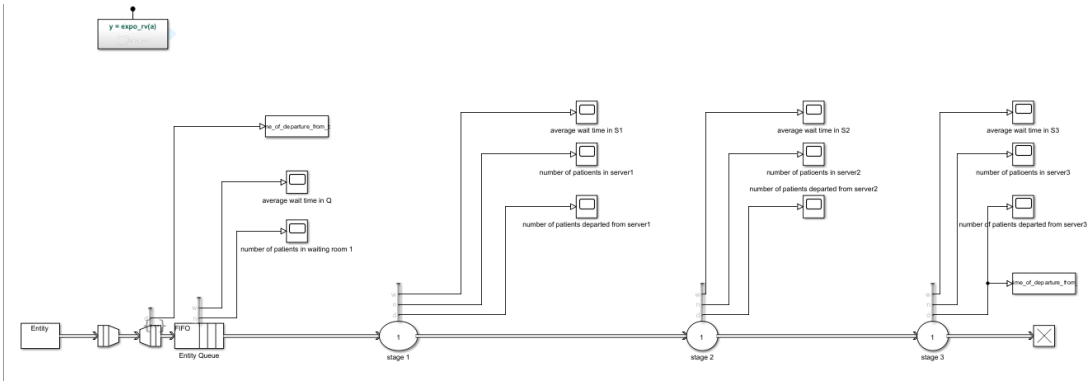

Figure 12: Simulink Model

We created three attributes for each entity.

In the Entity Queue, each entity's attributes are changed with respect to a pattern which corresponds to the matrix P1, P2 & P3 that we obtained from the optimization.

Below you can see how we inserted the pattern in the Queue

```
Pnew =
[P(:,20),P(:,19),P(:,9),P(:,4),P(:,1),P(:,16),P(:,17),P(:,7),P(:,5),P(:,11),P(:,8),P(:,3),P(:,13),
P(:,6),P(:,18),P(:,10),P(:,15),P(:,14),P(:,2),P(:,12)];
P1 = Pnew(1,:);
P2 = Pnew(2,:);
P3 = Pnew(3,:);


% Pattern: Repeating Sequence
persistent SEQ;
persistent idx;
if isempty(SEQ)
SEQ = P1;
idx = 1;
end
if idx > numel(SEQ)
idx = 1;
end
entity.Attribute1 = SEQ(idx);
idx = idx + 1;


% Pattern: Repeating Sequence
persistent SEQ1;
persistent idx1;
if isempty(SEQ1)
SEQ1 = P2;
idx1 = 1;
end
if idx1 > numel(SEQ1)
idx1 = 1;
end
entity.Attribute2 = SEQ1(idx1);
idx1 = idx1 + 1;

% Pattern: Repeating Sequence
persistent SEQ2;
persistent idx2;
if isempty(SEQ2)
SEQ2 = P3;
idx2 = 1;
end
if idx2 > numel(SEQ2)
idx2 = 1;
end
entity.Attribute3 = SEQ2(idx2);
idx2 = idx2 + 1;
```

Computation of flow time after optimization

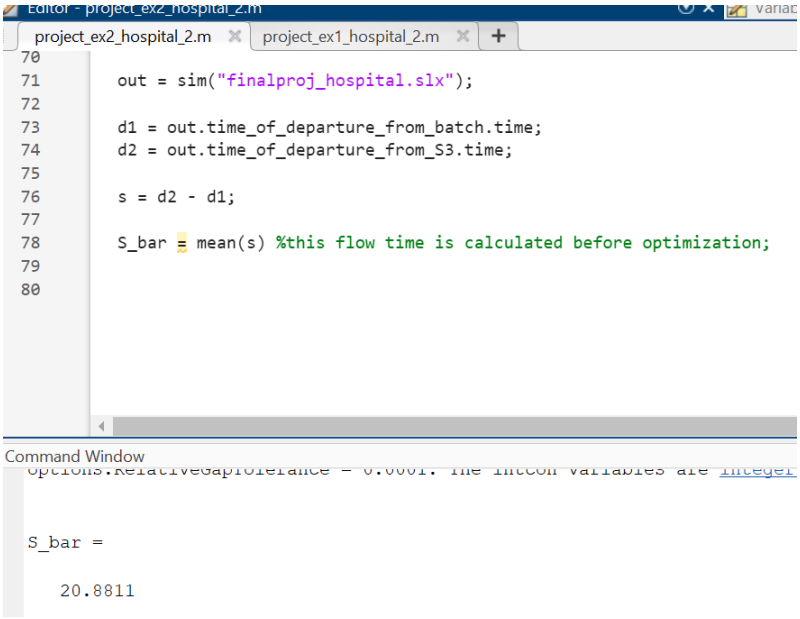

Figure 13: Average flow time after optimization

As we can see here, we obtained a lower S_bar after optimization.

Conclusion

This project presented a comprehensive analysis and simulation of a surgical unit operating under two scenarios: an emergency-based arrival model and a scheduled hospital day. Using Simulink for system modeling and MATLAB for optimization, we evaluated the performance of the system in terms of average patient flow time and resource allocation. Key outcomes include: • Identification of the minimum number of rooms required at each stage to ensure system stability (ergodicity). • Simulation of patient flow under realistic assumptions, including stochastic interarrival and service times. • Evaluation of cost-effective configurations that maintain acceptable flow times ($\leq$ 20 hours). • Implementation of a scheduling optimization algorithm that significantly reduced average patient time in the hospital from 26.5 hours to a lower optimized value, demonstrating the potential of data-driven scheduling. These results highlight the importance of integrating discrete-event simulation with mathematical optimization for healthcare operations. The modeling approach can be extended in future work by considering additional constraints (e.g., staff availability, emergency priority levels) or incorporating real patient data to further enhance decision-making