Project Title: Do not crash, please!

Alessandro Minoli 20202A

Università degli Studi di Milano - Dipartimento di Informatica

Course "Simulation" - Exam Project, A.Y. 2022-2023

Contents

1 Project specifications

Do not crash, please! [1]

In this report I will analyze a system of machines that need frequent maintenance. To be operational, the system needs N machines to be working at any given time and s additional machines are kept available as spares. Whenever a machine breaks down it is immediately replaced by a spare and is itself sent to the maintenance facility, which consists of w workers who bring machines (one at a time per worker) from an arrival area to a repair area. The maximum number of machines that can be simultaneously repaired is C.
Once a failed machine has been repaired, it becomes available as a spare to be used when the need arises. The system is said to "crash" when a machine fails and no spares are available. Our goal is to try to reduce the frequency of crashes: when the system crashes it needs some minutes to reboot, in that time it is completely unusable and, obviously, this translates into lower profits.
Currently the system is running with $s = 7$ and $w = 2$.
The average time a machine can run before needing maintenance has been observed to be between 90 and 150 minutes. The repair time of a machine, in a huge number of cases, has been observed to be approximately 10 minutes; it has never been less than 3 or more than 30 minutes.
The most interesting outcome of our experiments is the time at which the system crashes. However, we will also consider other useful quantities and we will look at how they vary with respect to changes in s and w.

2 Model

The number of machines that must always be working is $N = 20$.
The number of machines that can be simultaneously repaired is $C = 3$.
The number of spare machines is a parameter that currently has value $s = 7$.
The number of workers is a parameter that currently has value $w = 2$.

Modeling of machine failures

The time a machine functions before breaking down is modeled using an exponential random variable Y.
This RV is suitable because of its memoryless property: if a machine has already been active for 30 minutes is not more likely to break in the subsequent one than a machine that has been active since only 2 minutes ago.
Let's define Z as a truncated normal RV having mean 0, standard deviation 0.3, minimum value -1 and maximum value 1. Let z be a realization of Z we compute at the beginning of every simulation run and q be:

[1] The project specification is inspired by the model *A Repair Problem*, at page 124 of [1]

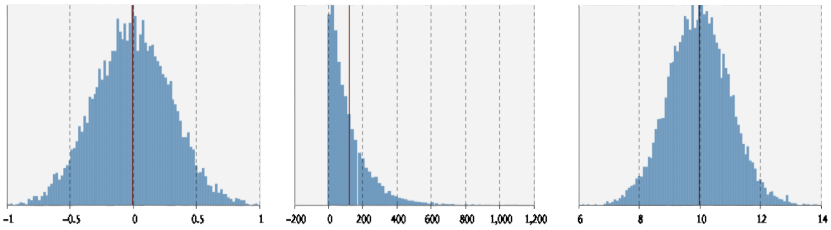
$$q = \begin{cases} z \cdot 0.1, & \text{if } z \leq 0 \\ z \cdot 0.16, & \text{otherwise} \end{cases}$$

We assume each machine to have, on average, $0.5 + q$ failures per hour.
The time scale we consider for our model is minutes. Thus, we set the rate of Y to be $\lambda_Y = (0.5+q)/60$ in order for itself to have a mean of $1/\lambda_Y = 60/(0.5+q)$. For example: if $q = 0$ then the average interfailure time will be 120 minutes, if $q = -0.1$ it will be 150 minutes and if $q = 0.16$ it will be 90 minutes.
The fact that the failure rate depends on a realization of Z makes each experiment having all machines a bit less or more failure-prone.

Modeling of maintenance process

When a machine is sent to maintenance it is initially placed in an arrival area within the facility. Once a worker becomes available, if it wasn't already, the machine is manually moved towards a repair point.
Only C machines can be simultaneously repaired, if all the repair points are occupied then the worker waits nearby until one of them becomes free.
The repair time of the machines has a mean of 10 minutes, however it is influenced by many external noises that are difficult to examine. We assume these noises, which can cause an increase or decrease of the overall repair time, to be all independent and identically distributed. If we sum them up, as the *Central Limit Theorem* states, we obtain an overall noise that is normally distributed. If we add this noise to the average repair time we finally get a normally distributed repair time.
We model it using a truncated normal RV X having mean 10, standard deviation 1, minimum value 3 and maximum value 30.

All PDFs of the discussed random variables are shown in figure 1.



(a) Z (b) $Y \mid z = 0$ (c) X

Figure 1: Probability density functions of the used RVs

Simulation paradigm

The simulation paradigm I chose is a hybrid between agent-based and discrete events models. Following the agent-based paradigm, I consider the machines as individuals that can be in different states (namely: *working* state, *spare* state or *maintenance* state). The most convenient feature of this paradigm that I exploited is that individuals are allowed to interact with each other sending messages. When some machine breaks, indeed, it must immediately send a message to a spare one in order to make it transition to the *working* state.
The discrete events paradigm has been used to model the maintenance process. Every machine is an agent that steps through the process blocks. The workers of the maintenance facility are treated as resources that the agents must acquire to flow from the arrival area to the repair area.

3 Implementation

The model has been implemented using the simulation software *Anylogic*.

The quantities N, C, z, q and the parameters s and w have been initialized in the simulation environment as explained previously.

I created a population of $N + s$ *Machine* agents that behave according to the statechart in figure 2. N individuals are initialized in *working* state and the other s in *spare* state. The transition from *working* to *maintenance* state is triggered by rate λ_Y, as discussed in section 2. Once an agent enters in *maintenance*, a *WakeUp* message is sent to one of the machines currently in *spare* state to make it transition to *working*. If none of the machines are in *spare*, the simulation finishes. After the message has been sent, the *Machine* enters the maintenance process, which is represented in figure 3.
The maintenance facility is organized as shown in figure 4.
When a *Machine* agent enters the process it is graphically placed in an arrival area. Logically, it is pushed into a FIFO queue waiting to get a worker that moves it to the repair area. When it arrives to destination, if there is a free repair point, it is released there by the worker and gets repaired for the already mentioned normally distributed time X. If all the C repair points are occupied, the worker waits in a FIFO queue for its turn to release the machine. When the maintenance is completed the agent messages itself with a *MaintenanceDone* message to transition to the *spare* state. When the workers are not busy moving machines, they wait for new tasks in a waiting area.

During the execution of the simulations it is displayed a time stack chart, like the one in figure 5, to see at any moment how many machines are available as spares and how many are in maintenance.

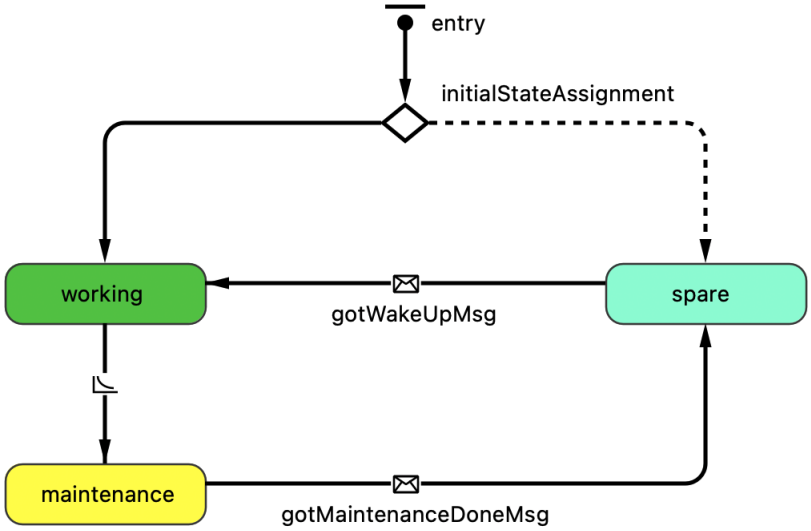


Figure 2: Statechart of the *Machine* agent

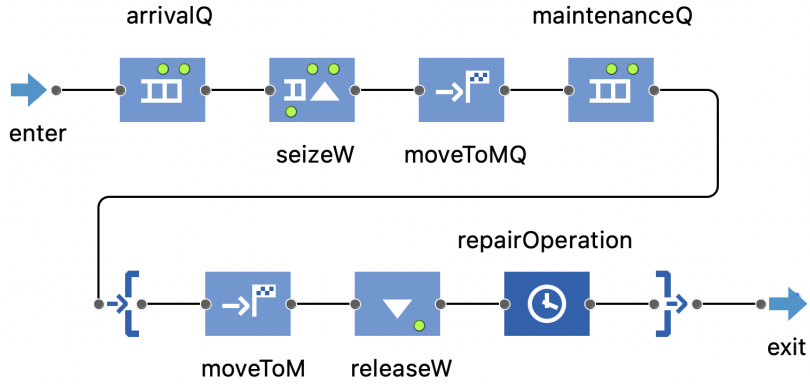


Figure 3: Structure of the maintenance process

4 KSP and KPI

Key System Parameters

The most important parameter of my model is the number of initial spare machines s. Currently the system works with 7 spares but we will see how the performances change along with some variations of this parameter. Also the number of workers w, normally equal to 2, will be subject to a similar inspection. Of course the system's performance is influenced by the rate at which the

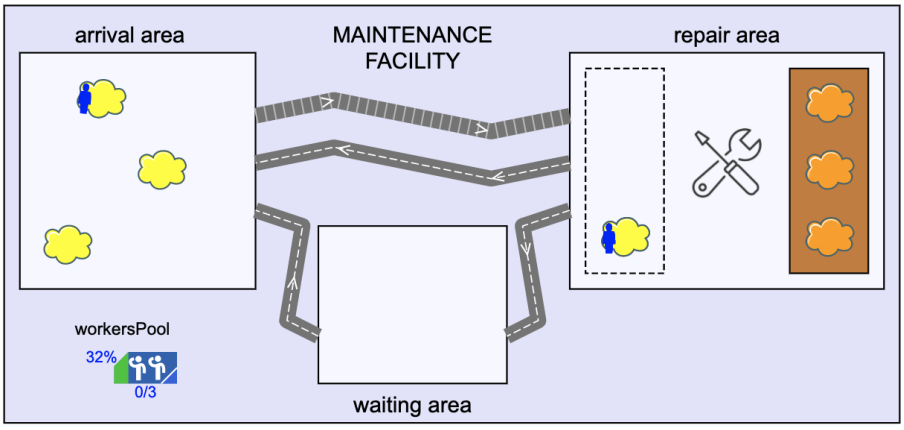


Figure 4: Maintenance facility

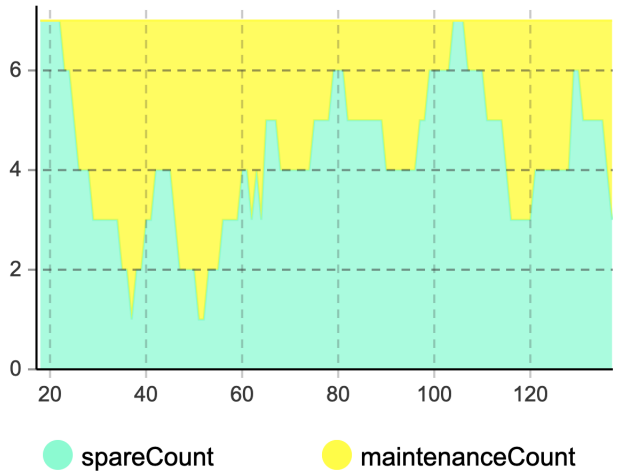


Figure 5: Time stack chart of the machines' state

machines need maintenance and by their repair time, however these are random variables whose parameters are provided to us as a result of experiments conducted by others so it's not wise to change them. The only thing we did was handle some uncertainty in the maintenance rate by introducing the variable q. N and C are constants, we're not allowed to modify their value.

Key Performance Indicators

The fundamental performance indicators of the system are :

- T_c - the time at which the system crashes
- T_q - the time a worker spends in the maintenance queue
- U - the mean utilization of the workers in a simulation run (i.e. the fraction of time they were busy)

5 Experimental analyses

All the analyses described in this section are performed in the Python notebook *experimental_analyses.ipynb*. The outcomes of our simulation runs are collected in a file called *results.txt*. Our model has been executed 500 times for each pair of values of $s \in \{6, 7, 8\}$ and $w \in \{1, 2, 3\}$. We can analyze what happens if the standard parameters of our system are increased or decreased by 1.

Estimate of the expected crash time of the system

The quantity that we wanna estimate is $\theta = \mathbb{E}[T_c]$, the expected time at which the system crashes. We do an unbiased estimate of θ by sample mean :

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{1}$$

where n is the number of simulation runs and X_i is the outcome of the i-th run. We define the sample standard deviation as :

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2} \tag{2}$$

By making interval estimates, we can give a range in which we are confident the parameter lies. The interval

$$\bar{x} \pm z_{\alpha/2}\frac{s}{\sqrt{n}} \tag{3}$$

is an approximate $100(1-\alpha)$ percent confidence interval estimate of θ.
The quantity z_α is such that $\mathbb{P}[Z > z_\alpha] = \alpha$, where Z is a standard normal RV.

We obtained that, performing 500 runs for each parameter's configuration, with a probability of 95% the results on $\mathbb{E}[T_c]$ in figure 6 hold.

We observe that if, for any reason, we have to run our system with only 1 worker then we have to expect a high decrease in $\mathbb{E}[T_c]$, independently of the value of s. Having one more spare, meaning $s = 8$, can definitely help our system run for a much longer time. On the contrary, losing a spare has a very negative effect on the estimates. It's not clear if assuming another worker is convenient or not, but surely it doesn't have a strong effect on the outcomes.

# workers	# spares = 6	# spares = 7	# spares = 8
1	148.69 ± 12.27	230.95 ± 18.07	343.60 ± 32.22
2	267.05 ± 22.20	**681.94 ± 65.13**	1471.05 ± 149.88
3	294.77 ± 26.84	646.07 ± 64.71	1634.82 ± 157.11

Figure 6: Estimates of $\mathbb{E}[T_c]$

Estimate probability that the system crashes before 8 hours of activity

The number of runs we performed is sufficiently large to try to build an ECDF of the crash time of the system for each parameter's configuration.
The results of this process are shown in figure 7.

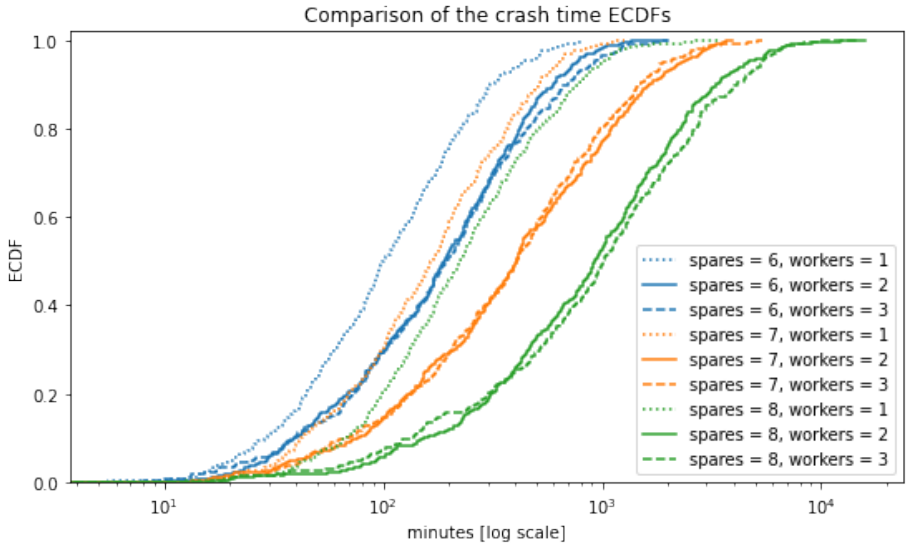


Figure 7: ECDFs of the crash time of the system

Having these functions, we can easily estimate $\mathbb{P}[$ "the system crashes before k hours of activity" $]$. We are interested in knowing the probabilities for $k = 8$.
Our estimates are shown in figure 8.

# workers	# spares = 6	# spares = 7	# spares = 8
1	0.95	0.88	0.78
2	0.85	**0.57**	0.31
3	0.81	0.56	0.28

Figure 8: Estimates of $\mathbb{P}[$ "the system crashes before 8 hours of activity" $]$

Distribution of the crash times

If we look at the empirical PDF of the crash times of our current system, with $s = 7$ and $w = 2$, we see that it seems to have an exponential decay.
If we compare it with some known theoretical distribution and rate them by the sum of square errors between our data and their value, we obtain that the 5 distributions that best match our observations are the ones in figure 9.

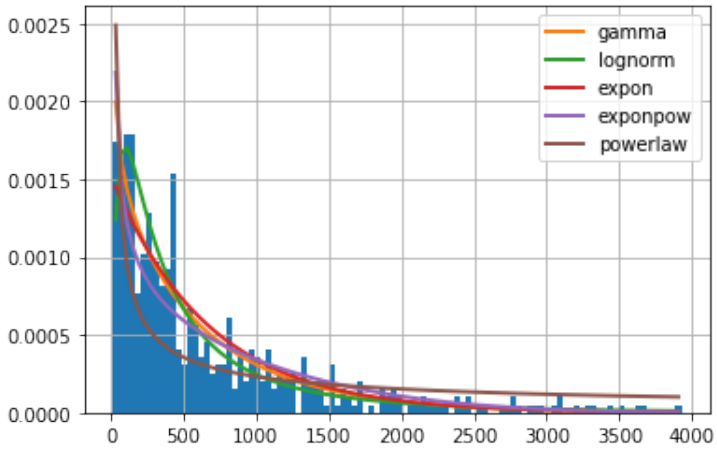


Figure 9: Theoretical distributions similar to our data distribution

Variance reduction by control variates on the estimate of the expected crash time of the system

Currently, our system is expected to crash after 681.94 ± 65.13 minutes.
We can have an unbiased estimate of $\theta = \mathbb{E}[T_c]$ also by sample mean on the RV $Y = X + c \cdot (Z - \mu_Z)$. We defined Z in section 2 and we know that $\mu_Z = 0$.

If c is chosen to be $c = -COV[X,Y]/VAR[Y]$, then one can prove that $VAR[Z] \leq VAR[X]$. We know that $VAR[Y] = 0.09$, $COV[X,Y]$ needs to be estimated.
If we apply this technique to our data we obtain another estimate of θ that is equal to 670.54 ± 59.45 minutes. As expected, now the confidence interval is tighter than the one in the basic estimate.

Estimate of the expected waiting time in the maintenance queue

We don't want our workers to waste too much time in the queue to release machines in a repair point. We define the Bernoulli RV P_q to be :

$$P_q = \begin{cases} 1, & \text{if worker has to wait in the queue because all the repair points are occupied} \\ 0, & \text{otherwise} \end{cases}$$

We obtained that, considering only 300 runs for each parameter's configurations, with a probability of 98% the results on $\mathbb{E}[P_q]$ and $\mathbb{E}[T_q \mid P_q = 1]$ in figure 10 hold.

	# spares		
# workers	6	7	8
1	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
2	0.17 ± 0.01 2.03 ± 0.10	**0.22 ± 0.01** **2.08 ± 0.07**	0.28 ± 0.01 2.13 ± 0.04
3	0.21 ± 0.02 2.85 ± 0.14	0.27 ± 0.02 3.27 ± 0.10	0.33 ± 0.01 3.50 ± 0.07

Figure 10: Estimates of $\mathbb{E}[P_q]$ (1st line) and $\mathbb{E}[T_q \mid P_q = 1]$ (2nd line)

We observe that if $w = 1$ then, obviously, the worker never waits in the queue. If we increase s to 8 or, independently of s, we put $w = 3$ then both the probability to wait and the waiting time in the queue get bigger. This result is explained by the idea that increasing the number of agents in the system has the natural consequence of making it more congested.

Estimate of the expected utilization of the workers

We obtained that, considering only 100 runs for each parameter's configuration, with a probability of 99% the results on $\mathbb{E}[U]$ in figure 11 hold.

# workers	# spares = 6	# spares = 7	# spares = 8
1	0.75 ± 0.03	0.80 ± 0.03	0.80 ± 0.03
2	0.49 ± 0.03	$\mathbf{0.48 \pm 0.03}$	0.51 ± 0.02
3	0.34 ± 0.02	0.35 ± 0.02	0.37 ± 0.03

Figure 11: Estimates of $\mathbb{E}[U]$

We observe that the values of utilization depend on w and seem to be independent of s. Continuing to run the system with 2 workers seems to be the wisest choice, doing so we maintain the workers' utilization around 50%.
Decreasing the number of workers makes the only remaining worker too busy. Increasing it to $w = 3$ makes them unoccupied for too much time.

Bootstrapping technique application

We define the RV L to be the number of times in a simulation run in which the number of spares becomes < 3. In figure 12 we can see an empirical PMF of L for our system. We do a bootstrap estimate, considering only 50 runs, of $MEDIAN[L]$ and $STD[L]$.
The resulting estimate of the median by sample median is $f(l) = 11.00$
The resulting estimate of the std by sample std is $g(l) = 15.37$
By resampling from the ECDF of L, we estimate the true value of our quantities of interest to finally obtain :
$\mathbb{E}_F[\text{MSE between } f(l) \text{ and } \theta(F)] = 0.171$.
$\mathbb{E}_F[\text{MSE between } g(l) \text{ and } \theta(F)] = 0.142$.

6 Conclusions

After the model has been designed, implemented and the results of many simulation runs have been analyzed, we can finally draw some conclusions about the behavior of our system. We saw how its performances look, in the current state, with respect to various indicators. The analyses in section 5 suggest that a possible decrease in the number of workers or in the number of spares can be very harmful for our system. This is the reason why it seems reasonable to increase the number of spares to 8 and to hire one more worker as soon as possible. The additional worker will not remarkably improve the performance of our system, it's only an act of prevention. Instead, having one more spare will also result in a performance gain.

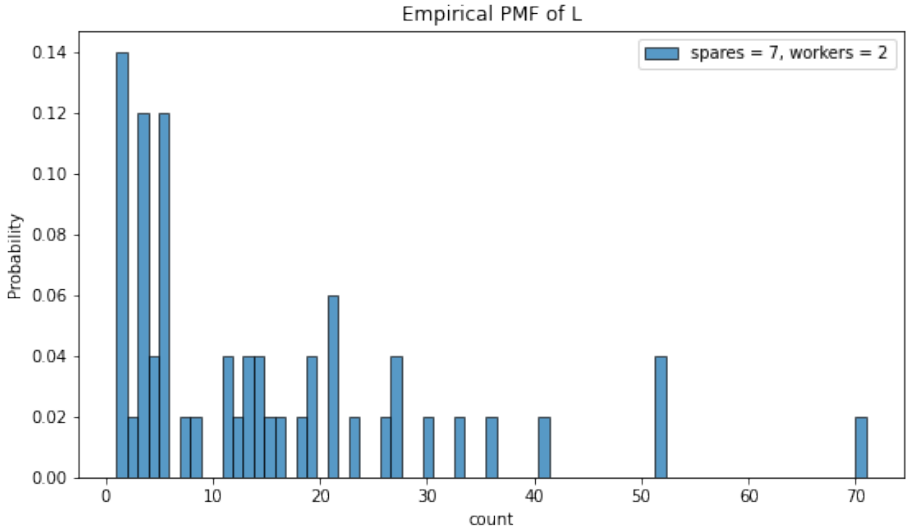


Figure 12: Empirical PMF of L

References

[1] Sheldon M. Ross, *Simulation*, 5th edition (2013)

[2] Alberto Ceselli, *Simulation*, lecture notes (2022-23)