Magistrello Camilla (4512554)

Assignment 1: scheduling simulator

In this assignment I had implemented a discrete event simulation.

The simplest case is where I have a single server and a queue of jobs. Jobs are served in a First-In-First-Out (FIFO) fashion.
This case is called M/M/1 queue because jobs arrive in a memoryless fashion, jobs are served in a memoryless fashion, and we have just one server.

In case of multiple servers (n), I need a load balancer to divide the load between servers.
It can assign load to server in three ways:

- Randomly.
- Assign jobs to the least loaded server.
- Supermarket queueing.

I had implemented the supermarket way.

I ask to a small number d of serves, with d < n, their length and then assign the job to the shortest queue.

I created for each server a list of queues (listOfQueueServers) and a list of state (listOfStateServer).

The list listOfStateServer save, for each server, the job.id whose is being scheduled for the completion.
The function min_queue search, for each of d server, the one with the shortest queue and return the server_id.

The inputs of the simulation are:

- λ: average frequency (rete) at which jobs join the queue
- μ: average frequency (rete) at which jobs leave the system when they are complete
- n: number of servers in the system
- d: sub number of server

The simulation can work with infinity n servers, but I test it with 100 of them.

I test the simulation with μ = 1, 100 servers and:

- λ = [0.5, 0.9, 0.95, 0.99]
- d = [1, 2, 5, 10]

For each of the simulations we calculated the average time spent in the system for every job and the theoretical expectation for random server choice.

These are the results I obtained:

- λ = 0.5, d = 1

  Average time spent in the system: 2.0013609955359506

  Theoretical expectation for random server choice: 2.0

- λ = 0.9, d = 1

  Average time spent in the system: 9.360147074369493

  Theoretical expectation for random server choice: 10.000000000000002

- λ = 0.95, d = 1

  Average time spent in the system: 13.192828808842693

  Theoretical expectation for random server choice: 19.999999999999982

- λ = 0.99, d = 1

  Average time spent in the system: 21.759008017288792

  Theoretical expectation for random server choice: 99.99999999999991

- λ = 0.5, d = 2

  Average time spent in the system: 1.2694256634731635

  Theoretical expectation for random server choice: 2.0

- λ = 0.9, d = 2

  Average time spent in the system: 2.742290134501027

  Theoretical expectation for random server choice: 10.000000000000002

- λ = 0.95, d = 2

  Average time spent in the system: 3.428301619646611

  Theoretical expectation for random server choice: 19.999999999999982

- λ = 0.99, d = 2

  Average time spent in the system: 5.064559703282767

  Theoretical expectation for random server choice: 99.99999999999991

- λ = 0.5, d = 5

  Average time spent in the system: 1.0402765785174704

  Theoretical expectation for random server choice: 2.0

- λ = 0.9, d = 5

  Average time spent in the system: 1.668719431124407

  Theoretical expectation for random server choice: 10.000000000000002

- λ = 0.95, d = 5

  Average time spent in the system: 2.0788738737557155

  Theoretical expectation for random server choice: 19.999999999999982

- λ = 0.99, d = 5

  Average time spent in the system: 3.0792776302545133

  Theoretical expectation for random server choice: 99.99999999999991

- λ = 0.5, d = 10

  Average time spent in the system: 0.9988767202907165

  Theoretical expectation for random server choice: 2.0

- λ = 0.9, d = 10

  Average time spent in the system: 1.4279658392487324

  Theoretical expectation for random server choice: 10.000000000000002

- λ = 0.95, d = 10

  Average time spent in the system: 1.6179168257431136

  Theoretical expectation for random server choice: 19.999999999999982

- λ = 0.99, d = 10

  Average time spent in the system: 2.6600309879098787

  Theoretical expectation for random server choice: 99.99999999999991

A notable improvement in the average time jobs spent in the system by jobs was observed.

I also calculated the lengths of the server queues every time the job ID was even, saving the measurements in CSV files called {d}_{λ}.csv

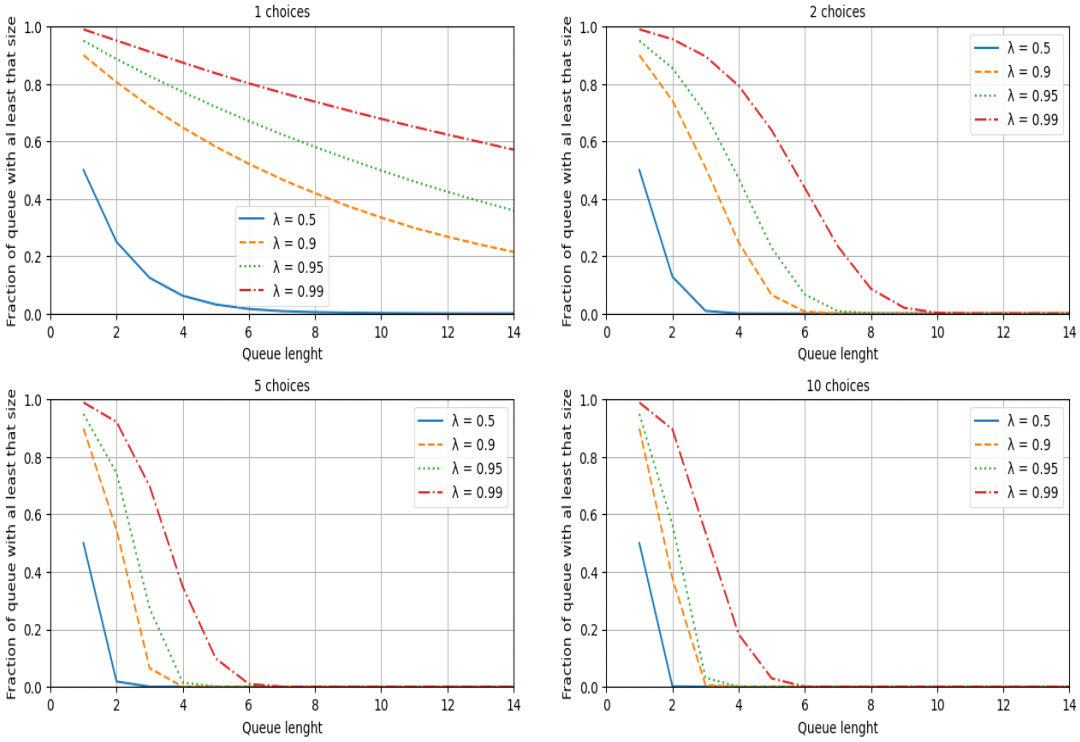
For each element 'd' in the choices list, a MeasurementGraphics object was created, consisting of a 'd' and a list of measurements for that 'd'. Each measurement is a MeasurementLambda object, consisting of λ and the list of normalized measurements.

To normalize the measurements, I have:

- taken all possible indices from 1 to 15
- from the list of measurements, summed all elements greater than or equal to that index
- each sum was divided by the maximum of the sums and multiplied by λ

```
for misureByReference in listaIndici:
    misurationsAdded.append(sum((x >= misureByReference) for x in self.misurations))
    misurationsNormalized = [(float(k)/max(misurationsAdded))*self.lambaData for k in misurationsAdded]
```

In this way, I obtained the following charts that show the relationship between the queue length and the fraction of queue with at least that size which are very similar to shown in the lesson slides (those are theoretical asymptotical results for n = infinity).



More choice I have less is the average queue length.

Simulator extension:

I chose to implement a priority queue because in the real world, when jobs are created, they almost never have the same priority.

To accomplish this, I assigned priorities to jobs at the time of their creation, ranging from 0 to 3, where 3 is the highest priority.

The initial idea was to make the server queues ordered by priority, reordering them when a new job is assigned to a server queue.

But this could lead to concurrency issues in the event that a server queue is reordered at the same time a job is scheduled for completion.

So, a 'fake' priority queue was created, meaning the queue remained unchanged, but when scheduling the job, the one with the highest priority is selected.

In this way, the lengths of the server queues remained the same, but the average time spent in the system by various jobs decreased.

```
def process(self, sim: MMN):
    assert sim.listOfStateServer[self.server_id] is not None
    sim.completions[self.job_id] = sim.t
    if sim.listOfQueueServers[self.server_id]: # if the queue is not empty
        maxPriority = -1
        next_job_id = -1
        indx = -1
        for i, job in enumerate(sim.listOfQueueServers[self.server_id]): # for each job
            indx = i
            if sim.listOfJobs[job] == 3: # if the job have the max priority
                next_job_id = job
                break
            elif maxPriority < sim.listOfJobs[job]:
                maxPriority = sim.listOfJobs[job]
                next_job_id = job

        if indx >= 0:
            del sim.listOfQueueServers[self.server_id][indx]
            sim.listOfStateServer[self.server_id] = next_job_id
            sim.schedule_completion(next_job_id,self.server_id) # schedule its completion
    else:
        sim.listOfStateServer[self.server_id] = None
```

These are the results I obtained:

- λ = 0.5, d = 1

  Average time spent in the system without priority: 2.0013609955359506

  Average time spent in the system with priority queue: 1.8221227802325075

  Theoretical expectation for random server choice: 2.0

- λ = 0.9, d = 1

  Average time spent in the system without priority: 9.360147074369493

  Average time spent in the system with priority queue: 4.546667468721231

  Theoretical expectation for random server choice: 10.000000000000002

- λ = 0.95, d = 1

  Average time spent in the system without priority: 13.192828808842693

  Average time spent in the system with priority queue: 5.444834285525192

  Theoretical expectation for random server choice: 19.999999999999982

- λ = 0.99, d = 1

  Average time spent in the system without priority: 21.759008017288792

  Average time spent in the system with priority queue: 5.761734708868114

Theoretical expectation for random server choice: 99.99999999999991

- λ = 0.5, d = 2

  Average time spent in the system without priority: 1.2694256634731635

  Average time spent in the system with priority queue: 1.272681058709308

  Theoretical expectation for random server choice: 2.0

- λ = 0.9, d = 2

  Average time spent in the system without priority: 2.742290134501027

  Average time spent in the system with priority queue: 2.593197808228961

  Theoretical expectation for random server choice: 10.000000000000002

- λ = 0.95, d = 2

  Average time spent in the system without priority: 3.428301619646611

  Average time spent in the system with priority queue: 3.472068314399358

  Theoretical expectation for random server choice: 19.999999999999982

- λ = 0.99, d = 2

  Average time spent in the system without priority: 5.064559703282767

  Average time spent in the system with priority queue: 5.061087332759245

  Theoretical expectation for random server choice: 99.99999999999991

- λ = 0.5, d = 5

  Average time spent in the system without priority: 1.0402765785174704

  Average time spent in the system with priority queue: 1.0326301996960618

  Theoretical expectation for random server choice: 2.0

- λ = 0.9, d = 5

  Average time spent in the system without priority: 1.668719431124407

  Average time spent in the system with priority queue: 1.6876056241838338

  Theoretical expectation for random server choice: 10.000000000000002

- λ = 0.95, d = 5

  Average time spent in the system without priority: 2.0788738737557155

  Average time spent in the system with priority queue: 2.161856512962556

  Theoretical expectation for random server choice: 19.999999999999982

- λ = 0.99, d = 5

  Average time spent in the system without priority: 3.0792776302545133

  Average time spent in the system with priority queue: 3.2661731268782686

Theoretical expectation for random server choice: 99.99999999999991

- λ = 0.5, d = 10

  Average time spent in the system without priority: 0.9988767202907165

  Average time spent in the system with priority queue: 1.007386379348192

  Theoretical expectation for random server choice: 2.0

- λ = 0.9, d = 10

  Average time spent in the system without priority: 1.4279658392487324

  Average time spent in the system with priority queue: 1.007386379348192

  Theoretical expectation for random server choice: 10.000000000000002

- λ = 0.95, d = 10

  Average time spent in the system without priority: 1.6179168257431136

  Average time spent in the system with priority queue: 1.650938424927781

  Theoretical expectation for random server choice: 19.999999999999982

- λ = 0.99, d = 10

  Average time spent in the system without priority: 2.6600309879098787

  Average time spent in the system with priority queue: 2.282664500007851

  Theoretical expectation for random server choice: 99.99999999999991

It can be observed that in the case of d = 1, the average time of jobs spent in the priority queue significantly decreases, for example, in the case of λ = 0.99, from 21.759008017288792 in the non-priority queue to 5.761734708868114 in the priority one.

While the relationship between the queue length and the fraction of queue remains the same.

The obtained results demonstrate that the priority queue speeds up the average time of jobs spent in the system.