Final Exam - Due by Monday 5/11

You are the CTO of the AI company "ClosedAI" providing Large Language Models (LLMs) inference on-demand. You are receiving constant pressure from your CEO to deliver the inference of the models at minimum cost. You are tasked with testing a pilot to control the scheduling of servers k_t in a data center. The historical time-stamped arrivals and service times to the data center are recorded in the file *past_arrivals.csv*. Physicists and engineers at the company estimate the following cost function for the operation of the data center for k active servers and N jobs in the system:

$$c(k,N) = \underbrace{\gamma\left[kP_i + N\Delta P\right]}_{\text{Linear Electricity}} + \underbrace{\gamma\left[\mu N^{\alpha}\right]}_{\text{LLM Memory Overhead}} + \underbrace{\lambda\left(\frac{(N-k)^+}{k}\right)^{\beta}}_{\text{SLA/Congestion Penalty}}, \tag{1}$$

The parameters are the effective energy cost $\gamma = \$0.33/kWh$, the idle power floor $P_i = 6kW$, the active power delta $\Delta P = 4kW$, the complexity exponent $\alpha = 1.2$ due to LLMs self-attention memory and KV Cache. A memory factor $\mu = 0.35kW$ and a Service Level Agreement of $\lambda = \$50/h$ with a congestion elasticity $\beta = 2$. The data center has a state-of-the-art controller that allows an algorithm to control the number of servers for the next hour. For example, given a vector $(0, t_1, t_2)$ with $0 < t_1 < t_2 < 1$ and an active servers plan (k_0, k_1, k_2), the data center for the next hour (the interval $[0, 1)$) will start activating k_0 servers between $[0, t_1)$, k_1 servers between $[t_1, t_2)$ and finally k_2 servers between $[t_2, 1)$ at which point the server scheduling algorithm is called again to decide the server schedule for the next hour. The scheduling algorithm *server_schedule* lives in a script called *scheduling_algorithm.py*. The workflow is as follows every hour:

- The function *server_schedule* receives numpy array with the timestamps of all the past arrivals of customers up to that hour.
- Your function *server_schedule* outputs two lists with the server plan for the next hour. One list with the activation times $[0, t_1, \ldots, t_j]$ and other with the number of servers for each interval $[k_0, k_1, \ldots, k_j]$. Remember that $0 < t_1 < t_2 < \cdots < t_j < 1$, you can choose any j, including the plan $[0], [k_0]$ which means there will be k_0 servers active for the full hour.
- Your plan remains active (and fixed) for the full hour, receiving jobs as they arrive and processing as a queueing system with processing times distributed the same as the historical data, each server is a cluster that can service 20 jobs concurrently. At the end of the hour, the cycle starts again.

To ensure that the script works as intended you are given a unit test script *unit_tests.py* ensuring that the output of your model runs within the infrastructure of the company. For the same reason, you are also constrained that the script can only use packages contained in the file *requirements.txt*. The performance of your strategy is evaluated over out-of-sample future hours continuing the time series of arrivals and service times you received.

$$\frac{1}{T}\sum_{t=1}^{T}\int_0^1 c(k_t, N_t)dt,$$

Task: Come up with a scheduling algorithm that performs best according to this metric.