OBP Assignment 3

Fergus Hathorn (2722167), Elham Wasei (2693347)

April 2022

a

Simulation procedure

Lognormally distributed times were generated by using the `numpy.random.lognormal(`μ, σ`)` command in Python. Parameters mean μ and standard deviation σ differ from the observed mean x and standard deviation y. Parameter μ is calculated by $\mu = \ln(x^2/\sqrt{x^2+y^2})$, while $\sigma^2 = \ln(1+y^2/x^2)$. For $x = 13$ and $y = 5$ to obtain $\mu = 2.496$ and $\sigma = 0.371$. A simulation model was programmed to read a schedule of the form $(x_1, \ldots, x_{12})$, where x_j is the scheduled start time in minutes of patient j. For each patient j, first the (actual, can deviate from scheduled) start time S_j is computed, with $S_j = \max(F_{j-1}, x_j)$, with F_j the finish time of patient j, where we initialize by setting $F_0 = 0$. Then the waiting time of patient j is calculated as $\max(0, S_j - x_j)$. Lastly, the finishing time for patient j is calculated as $F_j = S_j + X$, where $X \sim \text{Lognormal}(\mu, \sigma^2)$. Waiting times for each patient are stored up until the last patient, at which point the tardiness is calculated as $\max(F_{12} - 180, 0)$.

Calculating mean waiting time, tardiness and confidence intervals

To estimate the expected mean waiting time over all patients μ_w and expected tardiness μ_t for a particular schedule, we simulate the schedule N times and record mean waiting time and tardiness over each simulation. The Central Limit Theorem (CLT) will be used to support the following estimates. We want the confidence interval (CI) for μ_w and μ_t to be small, so N must be sufficiently large. We created n simulation batches consisting of 100 simulations, making $N = 100n$. The mean waiting time and tardiness for each batch $i \in \{1, \ldots, n\}$ is denoted as $\bar{W}_i$ and $\bar{T}_i$, respectively. Let $\bar{W}$ be the average of the waiting time batch means, i.e. $\bar{W} = \sum_{i=1}^{n} \bar{W}_i/n$. Likewise, we let $\bar{T}$ be the average of the tardiness batch means, i.e. $\bar{T} = \sum_{i=1}^{n} \bar{T}_i/n$. Using CLT, the batch means $\bar{W}_i$ and $\bar{T}_i$ are approximately $N(\mu_w, \sigma_w^2)$ and $N(\mu_t, \sigma_t^2)$ distributed, respectively, with σ_w^2 and σ_t^2 unknown. Thus, we have that, approximately, $\sqrt{n}(\bar{W} - \mu_w)/s_w \sim t_{n-1}$ where s_w is the sample standard deviation of $\{\bar{W}_i\}_{i=1}^{n}$, and $\sqrt{n}(\bar{T} - \mu_t)/s_t \sim t_{n-1}$ where s_t is the sample standard deviation of $\{\bar{T}_i\}_{i=1}^{n}$. It follows that a 95% confidence interval for μ_w is given by $\bar{W} \pm \frac{s_w}{\sqrt{n}} t_{n-1;\,0.975}$, and for μ_t is given by $\bar{T} \pm \frac{s_t}{\sqrt{n}} t_{n-1;\,0.975}$. Using Python we make use of a while loop that continues producing new batches until n is large enough so that $2 \cdot \frac{s_w}{\sqrt{n}} t_{n-1;\,0.975} \leq \bar{W}/5$ and $2 \cdot \frac{s_t}{\sqrt{n}} t_{n-1;\,0.975} \leq \bar{T}/5$. For the individual schedule, we find an $\mu_w = 2.76$ minutes with 95% CI [2.607, 2.920] and $\mu_t = 4.01$ minutes with 95% CI [3.627, 4.402]. The same procedure is applied to the objective value in Section (c) in order to define a CI.

b

Defining the neighbourhood

Now we work with schedules of the form $(n_1, \ldots, n_{36})$ where n_i is the number of patients scheduled at time slot i. Given a particular schedule S_1, it can be seen as a point in a space and we need to define the neighbourhood of this point. We define it as follows. Another schedule S_2 is called a neighbour if we can obtain S_2 from S_1 by moving one patient from one slot to a neighbouring slot, i.e. changing n_i to $n_i - 1$ (this must be positive) and n_j to $n_j + 1$ for some i and j, $j = i \pm 1$. But S_2 must satisfy additional conditions. We cannot obtain S_2 by moving a patient from the last two slots of S_1 or to the last two slots of S_1. Furthermore, S_2 is not allowed to have 0 patients scheduled in 4 consecutive time slots (except for the last 10 slots) and it cannot have ≥ 3 patients scheduled within 3 consecutive slots (except the first 8 slots). A schedule that violates these conditions can be interpreted as having infinite distance to all other possible schedules, which prevents them from being visited. Using this neighbourhood, a neighbour S_2 from S_1 can be generated by picking time slot i uniformly random from the time slots where at least one patient is scheduled in that time slot, and moving that patient randomly to either the later or earlier time slot (move deterministically at $i = 0$ or $i = 34$), and checking the conditions above are satisfied (and else generate another candidate neighbour).

Sim-opt

We start the algorithm with the individual schedule and we generate a neighbour. Both schedules are simulated 2000 times (which is a slight modification to the original sim-opt algorithm), and we store the following information for both schedules: the count (both updated to 2000), the sum of objective values (defined as $2\times$ tardiness + waiting time) across all simulations done so far for the schedule, and the mean objective value, which is this sum divided by the count. We make a jump to this neighbour, and set it as the 'current' schedule, if its mean objective value is lower. Then we repeat this procedure iteratively, but we make another modification to the original algorithm: each time a neighbour is generated that hasn't been simulated before, we simulate both the current and the neighbour 2000 times, else we simulate both schedules once. This ensures we only make jumps when a neighbour has a moderately high probability of being better (since approx. 2000 simulations are needed to get an idea about the mean, which we learned when computing confidence intervals), but not a too high probability since then we converge too quickly to a local optimum. After the exploration stage we do single simulations to try to make many visits to the optimal schedule.

We have a budget of 1000000 simulations and we subtract 4000 from the remaining budget if we generate a new neighbour and we subtract 2 if we generate a neighbour that has already been simulated. We iterate until we don't have enough budget to make the required simulations.

c

The optimal schedule, according to the Sim-opt, was found to be [1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0], yielding an expected objective value of 7.23. The method detailed in Section (a) was used to calculate a CI for the objective value, that is, we made use of batch-means and CLT. The stopping condition was adjusted to ensure that that the width of the CI was less than 5% of the mean objective value. In this case, the optimal schedule has a 95% CI of [7.13, 7.33].

d

The percentile waiting times $(10\%, 20\%, \dots, 100\%)$ for each of the 12 patients for our optimal schedule is shown in Figure 1. We opted to use a log-scale on the y-axis, as the 100th percentile waiting time was significantly higher all other 10th percentiles. This is not surprising, as the 100th percentile represents the largest waiting time for a particular patient. As expected, the first patient never experiences any delay, thus all percentile waiting times are shown to be zero. The objective function places more weight on tardiness, therefore good solutions do not typically schedule many patients towards the end of the three hour period. Accordingly, the percentile waiting times are consistently lower towards the end of the schedule. Patients 2-5 are the only ones that show a strictly positive waiting time for any percentile below the median, with patient 3 being the only patient with waiting times larger than zero for the 20th and 30th percentile. No patients exhibited a waiting time of greater than zero for the 10th percentile.

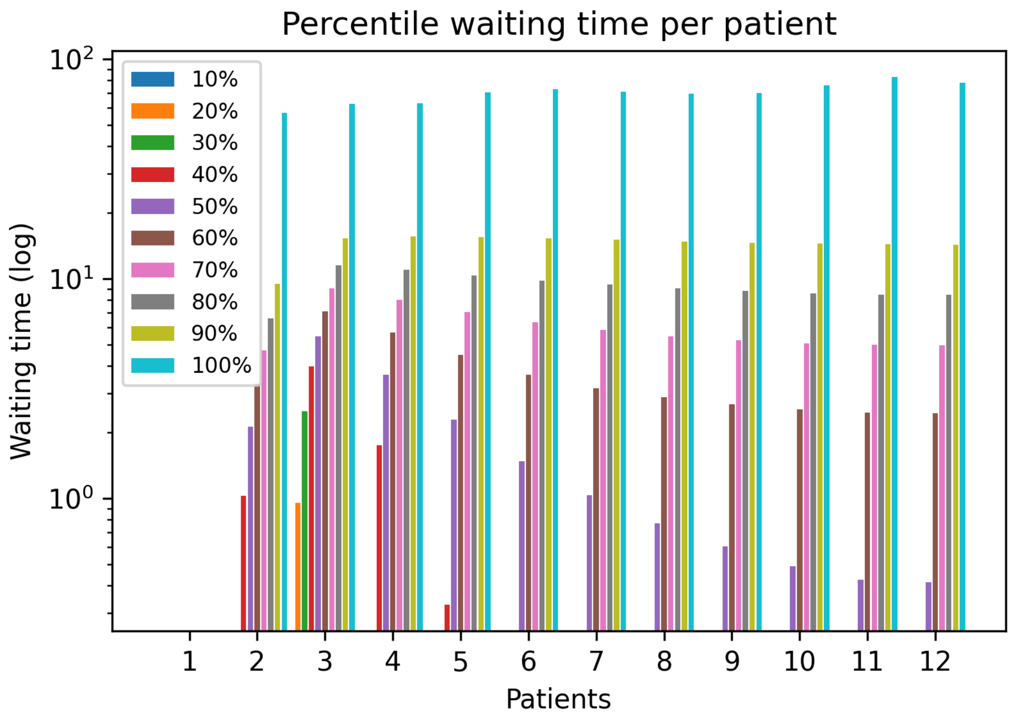

Figure 1: Plot showing waiting time for each 10th percentile for all 12 patients.