Norges Handelshøyskole

BAN403: Simulation of Business Processes

Professor Julio Cesar Goez

Project 1:
Basic simulation and queueing theory

March 11^{th}, 2024

1 Monte Carlo Simulation

The objective of this task was to identify portfolio weights that maximized the Sharpe-Ratio of a portfolio. The Sharpe-Ratio is a measure of risk-adjusted returns. The data consisted of five years of daily closing prices for Apple (AAPL), Google (GOOG), Amazon (AMZN), Microsoft (MSFT) and Tesla (TLSA). All gathering of data, simulations, and computations was done in R (see attachment 1).

To find the Sharpe-maximizing weights, we relied on Monte Carlo simulation to randomly generate weights of the stocks in the portfolio. We simulated 100,000 sets of weights, under the assumption that this number of simulations would enable us to identify the Sharpe-maximizing weights, while keeping the computation time at an acceptable level. The weights were simulated by drawing random numbers from a uniform distribution with a min of zero and max of 1, which we then normalized by dividing each set of weights on the sum of set. The uniform distribution was selected under the assumption that only long positions were allowed, meaning all weights had to be non-negative. Furthermore, the normalization of the weights, causing the sum of each set to equal one, was done under the assumption that the portfolio had to be fully invested and non-leveraged.

The Sharpe-Ratio is computed by dividing the difference of the expected return of the portfolio and risk-free rate on the standard deviation of the portfolio. We used the US 10-Year Treasury bill as a measure of the risk-free rate, with an annual rate on the 03/04/2024 of 4.22% (YCharts, n.d.) which we converted to a daily-rate to coincide with the daily expected return and standard deviation of the portfolio. The portfolio expected return was computed as the sum of the products of the five year average of daily returns for each stock and its corresponding simulated portfolio weight. The portfolio standard deviation was computed as the matrix product of the weights and the covariance matrix of the portfolio, and the matrix product of the transposed weights.

After computing the Sharpe-Ratio for each set of simulated weights, we identified the following Sharpe-maximizing set: AAPL: 27.15%, GOOG: 0.37%, AMZN: 0.55%, MSFT: 38.67% and TSLA: 33.26%, resulting in a Sharpe-Ratio of 0.0707. The positive ratio implies a positive excess return compared to the risk-free rate, accounting for the additional risk investing in the market.

The left of figure 1 illustrates the efficient frontier of the simulation. Each point represents the expected return and risk of a portfolio with different asset allocations. The set of weights that maximized the Sharpe-Ratio is highlighted. The right of figure 1 shows the distribution of the Sharpe-Ratios computed.

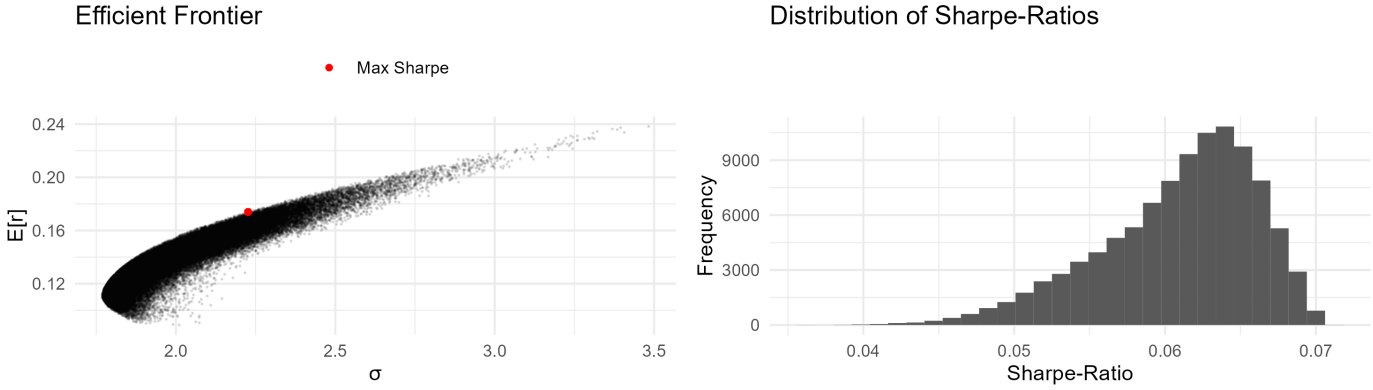

Figure 1: Resulting efficient frontier and distribution of Sharpe-Ratio from the Monte Carlo Simulation.

2 Queuing Theory

We first describe the implementation in JaamSim (JS) including the most important settings and functionalities used. Note that this is not a complete documentation of the JS model, but mentions only the most important configurations.

The JS model is displayed in figure 2. We use three different simulated entities and three different entity generators to keep track of the machines that are currently in the repair process. This is also important because each individual machine breaks down on average every 8 hours, independently of the other two machines. Each

entity has the attributes "Machine1ID", "Machine2ID", and "Machine3ID", one of which is set to 1 if this entity is the corresponding machine. This is important because a machine that is currently serviced by one of the teams cannot be generated as a simulated entity before it is repaired and running again. We incorporate this logic using thresholds for every machine-generator. The threshold for one machine is closed as long as this machine is in the repair process.

Furthermore, the objective is to simulate this process with different general distributions for inter-arrival times. To simulate all distributions in one run, we make use of the scenario logic in JS. A tab-delimited text file is used as input data (see attachment 3) and for each scenario the type of distribution and the run duration is taken from this file. We choose to simulate with exponential, normal, uniform, and gamma distribution.

We collect times spent in each state (0, 1, 2 or 3 machines nonworking) through an entity system object in JS, where an expression is used to evaluate and change the state of the whole entity system. The "Waiting"-state is assigned at the queue object and the "BeingRepaired"-state at each service team. Total time spent in the system on average is collected at the statistics object. All metrics demanded by the PaperCo management are saved as a .dat-file and included in the attachments to this report as an Excel-file (attachment 4). Average values over 10 replications per scenario are displayed in table 1.

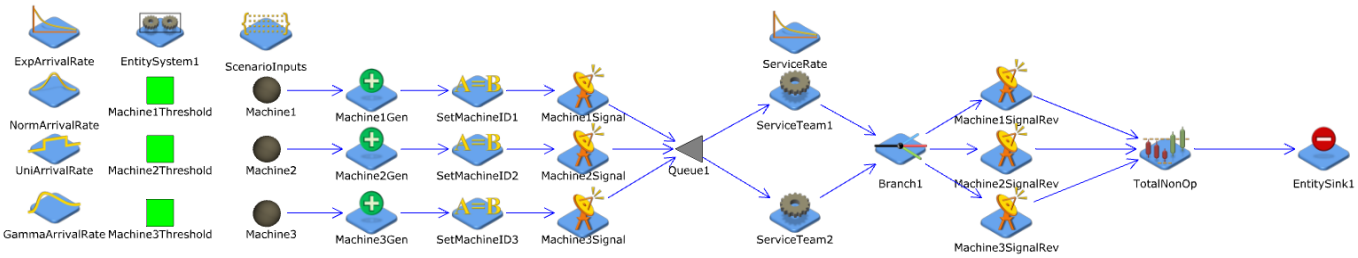

Figure 2: JaamSim Model for machine repair process.

Table 1: Average values for metrics (rounded) from the machine repair process simulation.

Metric	Analytical	Exponential Distr.	Normal Distr.	Uniform Distr.	Gamma Distr.
P_0 **(0 machines nonworking)**	0.2909	0.29	0.29	0.29	0.29
P_1 **(1 machines nonworking)**	0.4364	0.44	0.44	0.44	0.44
P_2 **(2 machines nonworking)**	0.2182	0.22	0.22	0.22	0.22
P_3 **(3 machines nonworking)**	0.0545	0.055	0.054	0.054	0.054
W **(Avg total time)**	4.22 (h)	4.22 (h)	4.22 (h)	4.22 (h)	4.22 (h)
W_q **(Avg queue time)**	0.22 (h)	0.22 (h)	0.22 (h)	0.22 (h)	0.22 (h)
L **(Avg no. of jobs)**	1.0363	1.04	1.04	1.04	1.03
L_q **(Avg no. of jobs in queue)**	0.0545	0.055	0.054	0.054	0.054

It is worth noting that our model theoretically has no restriction on the queue length (∞), however stops new simulated entities being generated when three machines are broken down (N = 3). Consequently, since there are two service teams, the queue length never exceeds 1. The conceptual notation of $M/M/c/\infty/N$ is thus misleading in this particular scenario.

Firstly, table 1 shows small deviations of the simulated metrics from the analytically calculated metrics, when simulating with exponentially distributed arrival rates. Minor deviations are expected in a simulation. All values are rounded and exact values can be seen in attachment 4. Secondly, we obtain very similar (not to say the same) values for other general distributions. This clearly validates the remark made in the book. The formulas for metrics in the $M/M/c/\infty/N$ model can also be applied to systems with normally, uniformly, and gamma distributed arrival rates.

3 General Analytical Models

a) Steady-State Probabilities: In steady-state, the expected number of departures from a state in a unit of time equals the expected number of entrances to the state. This characteristic can be used to identify the

balance of flow equations needed to compute the steady-state probabilities in the drive-through system. Using the transition diagram, eight equations corresponding to the eight states in the system can be formulated:

$$\pi_{0,0,0} \cdot \lambda = \pi_{0,0,1} \cdot \mu_2 \tag{1}$$
$$\pi_{0,1,0} \cdot (\lambda + \mu_1) = \pi_{0,0,0} \cdot \lambda + \pi_{0,1,1} \cdot \mu_2 \tag{2}$$
$$\pi_{0,0,1} \cdot (\lambda + \mu_2) = \pi_{0,1,0} \cdot \mu_1 + \pi_{0,b,1} \cdot \mu_2 \tag{3}$$
$$\pi_{1,1,0} \cdot \mu_1 = \pi_{0,1,0} \cdot \lambda + \pi_{1,1,1} \cdot \mu_2 \tag{4}$$
$$\pi_{0,1,1} \cdot (\lambda + \mu_1 + \mu_2) = \pi_{0,0,1} \cdot \lambda + \pi_{1,1,0} \cdot \mu_1 + \pi_{1,b,1} \cdot \mu_2 \tag{5}$$
$$\pi_{0,b,1} \cdot (\lambda + \mu_2) = \pi_{0,1,1} \cdot \mu_1 \tag{6}$$
$$\pi_{1,1,1} \cdot (\mu_1 + \mu_2) = \pi_{0,1,1} \cdot \lambda \tag{7}$$
$$\pi_{1,b,1} \cdot \mu_2 = \pi_{0,b,1} \cdot \lambda + \pi_{1,1,1} \cdot \mu_1 \tag{8}$$

To compute the steady-state probabilities, we used the Solver in Excel (see attachment 5) on the balance of flow equations. The solver objective was to sum the probabilities to one, as all steady-state probabilities in a system must equal one. The solver returned the following results:

Table 2: Computed steady-state probabilities for the drive-through system.

State	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(0,b,1)	(1,1,0)	(1,1,1)	(1,b,1)
Probabilities (%)	4.67	5.84	10.65	15.46	5.15	32.47	11.04	14.72

b) Steady-State Analysis: The average number of clients in the system can be computed by summing the products obtained by multiplying the number of clients in each state with their corresponding steady-state probability (here rounded to simplify calculation):

$$L = (0.05) \cdot 0 + (0.06 + 0.11) \cdot 1 + (0.15 + 0.05 + 0.32) \cdot 2 + (0.11 + 0.15) \cdot 3 = \underline{\underline{1.99 clients}}$$

The system is limited to three clients at a time, thus any further arriving clients will be rejected. As a result, the average rate of clients entering the system (λ^*) and the inter-arrival rate of clients(λ) will differ. The average rate of arrivals is the sum of the products obtained by multiplying the steady-state probabilities of states allowing for client arrivals and the inter-arrival rate:

$$\lambda^* = 50 \frac{clients}{hour} \cdot (0.05 + 0.11 + 0.06 + 0.15 + 0.05) = \underline{\underline{21 \frac{clients}{hour}}}$$

The average time a client spends in the system (W) is computed using Little's law:

$$W = \frac{L}{\lambda^*} = \frac{1.99}{21} = 0.095 hours = \underline{\underline{5.7 minutes}}$$

c) Assessing System Stability For the drive-through system to be stable, the inflow-rate of clients and the outflow-rate must equal. Clients enter the system in the queue before the microphone, and leave the system after receiving their order. Because the path every client follows through the system is known, the average rate of clients leaving the system (μ^*) can be computed using the same procedure as the the average arrival rate. The average rate of clients leaving is: $\mu^* = 40 \cdot (0.06 + 0.15 + 0.05 + 0.11 + 0.15) = 21 \frac{clients}{hour}$. As the computed λ^* equals μ^*, this indicates that the drive-through system is in fact stable.

d) Drive-Through Model in JaamSim Figure 3 shows the JS model for the drive-trough example. We incorporate the distributions for the food preparation rate in different scenarios in the same model.

Indeed, when simulating for a sufficiently long duration (exponential distribution), the steady-state probabilities, L, and λ^* converge to the same values as calculated. We get small expected deviations due to randomness in the simulation, but these can be neglected. Things look different when we have a normally distributed food preparation and collection rate. Both scenarios are displayed in table 3 along with the differences between normal

and exponential distribution. The deviations range up to -7.24 percentage points for the (1,b,1)-state. Considering the balance of flow equations used to calculate the steady-state probabilities, this should come as no surprise. The equations assume exponentially distributed service times. Contrary to changing the inter-arrival-rate as in Task 2, changing the service time to be normally distributed has a bigger effect on the mathematical properties of the system. When we loose the memoryless property of the exponential distribution for the service time, the mathematical expressions for steady-state probabilities no longer hold and we cannot analyse the model as a M/M/[...] model. Thus, deviations of these metrics, as seen from the JS model, are expected.

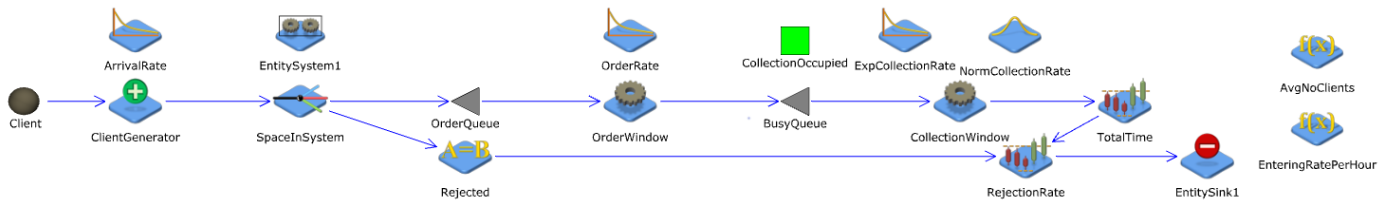

Figure 3: JaamSim Model for machine repair process

Table 3: Steady-state probabilities for the different states of the simulated drive-through system.

Distribution	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(0,b,1)	(1,1,0)	(1,1,1)	(1,b,1)
Exponential (%)	4.68	5.83	10.64	15.44	5.16	32.50	11.01	14.73
Normal (%)	2.94	7.30	6.39	21.89	6.77	34.04	13.18	7.49
Diff Norm-Exp (%-points)	-1.74	1.47	-4.26	6.44	1.61	1.54	2.17	-7.24

4 Airline Ticket Counter

a) JaamSim Model with different queues: In the first model, we simulate the airline ticket counter with different queues for each agent. The critical JS implementation is to determine the shortest queue at arrival of the customer, since the customer always chooses the shortest queue. We calculate this using an expression entity. It is assumed that a customer currently being served is included in the queue length, for logical reasons. Furthermore, our model simulates 5 different scenarios with 1, 2, 3, 4, and 5 Agents working simultaneously. The thresholds (green in figure 4) are opened or closed depending on the current scenario. We make the following model specific assumption: When two queues have the same length, the branch always routes the customer to the first available agent from the top, while in reality this decision would be more random. This is the reason why the working rate for Agent1 is always the highest, no matter how many agents are staffed. We included a different model in figure A2 (attachment 14) that simulates a random choice between two queues if they have the same queue length.

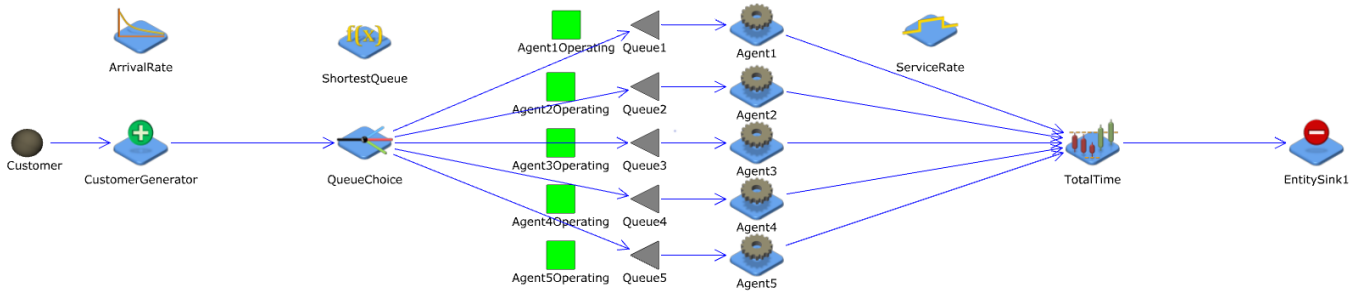

Figure 4: JaamSim Model for airline ticket counter with different queues

Results. When only one Agent is working, length of Queue1 tends to increase infinitely, thus the waiting time also tends to increase infinitely. When running the simulation for 500,000 minutes, we record an average waiting time of 41,966 minutes. We can clearly say that the system is not stable when only Agent1 is working.

In Scenario 2, Agent2 is "unlocked" and the average waiting time decreases to 2.28 minutes independent of the run duration of a simulation. The system now converges to steady state and the two agent's work is enough for the queue to not increase to infinity. Three agents working further decreases the average waiting time to 0.49 minutes, however more agents do not cause significant improvements. Thus, two agents are needed to achieve a maximum waiting time of 5 minutes.

b) JaamSim Model with one queue: The JS model with only one queue for all agents is almost identical to the model in a), except for one queue routing all customers to the first available agent (see Figure A1). Furthermore, the expression entity is not needed anymore. It goes without explanation that the average waiting time when only one agent is working is the same as in model a) (41,966 minutes). An additional agent decreases the average waiting time to 2.0 minutes, converging the system to steady-state.

c) Comparison of the models in a) and b): Both models need the same number of agents to achieve a maximum waiting time of 5 minutes. Model b), however, reduces the waiting time with two agents to 2.0 minutes, while model a) achieves only 2.28 minutes. Interestingly, model b) is also able to reduce the time further to 0.3, 0.05, and 0.009 minutes for 3, 4, and 5 agents respectively, while model a) achieves no noteworthy improvements staffing more than 3 agents (0.49, 0.48, 0.48 minutes). The logical explanation for this is the neglection of remaining working time at the agents when determining the shortest queue. It does not matter for the queue length whether an agent has 2 minutes or 10 minutes working time left before the next customer can move up, both count as +1 in queue length. When a customer has chosen a queue/agent and another agent comes available earlier, this customer cannot change the queue or agent in model a). Model b) works more efficiently, routing customers to whatever agent becomes available first.

d) JaamSim model with different counters: For the last model we have to make a few changes in JS. Assuming that we start from model b), the two counters have one queue each for all agents and customers are routed to the agent in each counter that is available first (see figure 5). We simulate three scenarios: 1 agent on check-in and purchase each, 2 on check-in and purchase each, and 2 on check-in and 3 on purchase. One could also simulate different staffing levels of the two counters, however, we see that the average waiting time for one agent on each counter 2.32 minutes, which is already below 2 minutes. Whith 15% of customers purchasing tickets, approximately 1 in 7 customers purchases a ticket, which increases the average interarrival time for purchasing clients to approximately 35 minutes, which is not a problem for a uniformly distributed service time between 12 and 18 minutes. Moreover, the shorter check in service time between 2 and 4 minutes easily deals with an average interarrival time of approximately 6 minutes. Hence, our results of only one agent per counter are as analytically expected. We thus suggest for the airline to have one agent on each counter.

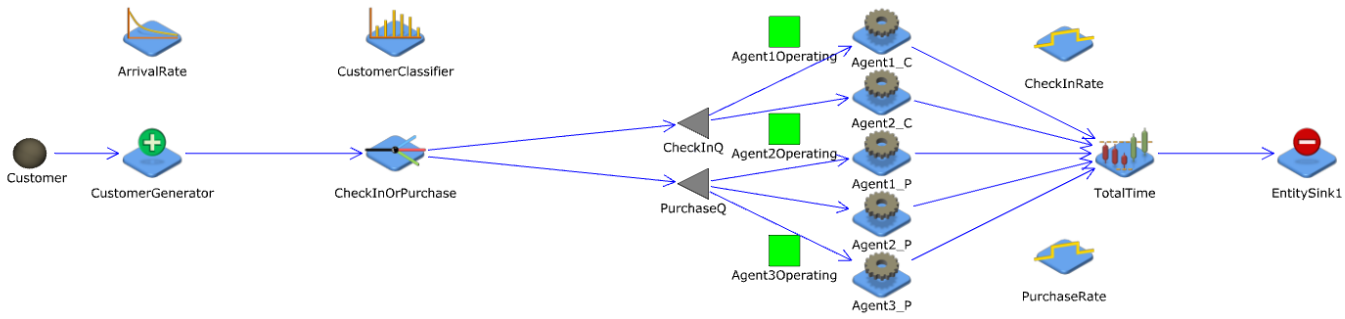

Figure 5: JaamSim Model with different counters

References

YCharts. (n.d.). 10-Year Treasury Rate [Accessed: 03/08/2024]. https://ycharts.com/indicators/10_year_treasury_rate?

Appendix

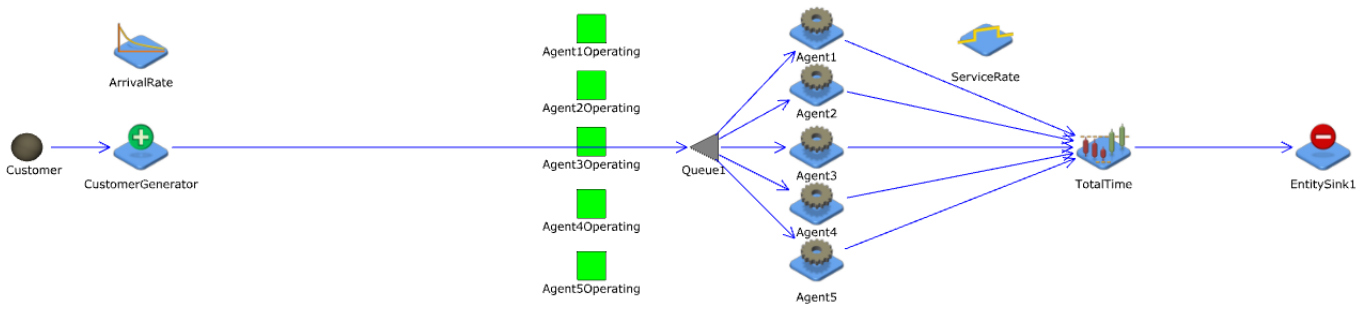

Figure A1: JaamSim Model for airline ticket counter with one queue.

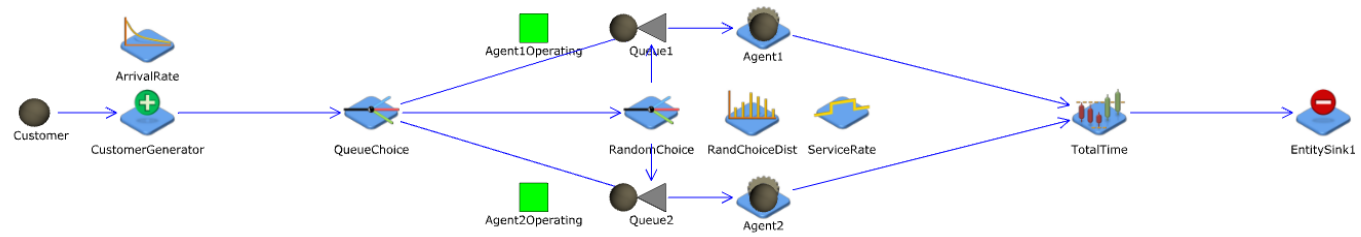

Figure A2: JaamSim Model for airline ticket counter with different queues and random assignment at same queue length.

Attachments

Attachment 1: Task1.R. R coding file for monte carlo simulation of portfolio weights.

Attachment 2: BAN403_Project1_QueuingTheory.cfg. JaamSim model for machine repair process.

Attachment 3: BAN403_Project1_QueuingTheory_InputData.txt. Input data file for multiple scenarios in JaamSim model for machine repair process.

Attachment 4: BAN403_Project1_QueuingTheory_output.xlsx. Nicely formatted output file for JaamSim machine repair process metrics.

Attachment 5: task3_solver.xlsx. Excel-file to solve the balance of flow equations for steady state probabilities.

Attachment 6: BAN403_Project1_DriveThrough.cfg. JaamSim model for drive through system.

Attachment 7: BAN403_Project1_DriveThrough_output.xlsx. Nicely formatted output file for JaamSim drive through process metrics.

Attachment 8: BAN403_Project1_AirlineTicketCounter_a.cfg. JaamSim model for Task 4 a).

Attachment 9: BAN403_Project1_AirlineTicketCounter_a_output.xlsx. Nicely formatted excel file for airline ticket counter metrics with different queues.

Attachment 10: BAN403_Project1_AirlineTicketCounter_b.cfg. JaamSim model for Task 4 b).

Attachment 11: BAN403_Project1_AirlineTicketCounter_b_output.xlsx. Nicely formatted excel file for airline ticket counter metrics with one queue.

Attachment 12: BAN403_Project1_AirlineTicketCounter_d.cfg. JaamSim model for Task 4 d).

Attachment 13: BAN403_Project1_AirlineTicketCounter_d_output.xlsx. Nicely formatted excel file for airline ticket counter metrics with two different counters.

Attachment 14: BAN403_Project1_AirlineTicketCounter_a_option2.cfg. JaamSim model for Task 4 a) option with random assignment on same queue length.