MSCI512 Group Coursework

1. (a)

To calculate the probability of a call being answered immediately without any wait, we must sum the probabilities of the system being in a state in which the number in the system is **less** than the number of servers.

$$P(Immediate\ Call) = \sum_{n=0}^{s-1} \pi_n$$

The formula used to calculate the probability of being in state π_n is the steady state distribution for an M/M/S queue.

$$\pi_n = \frac{\lambda^n}{n!\ \mu^n}\pi_0 \quad if\ n \leq s$$

$$\pi_0 = \left[\sum_{n=0}^{s-1} \frac{(s\rho)^n}{n!} + \frac{(s\rho)^s}{s!\,(1-\rho)}\right]^{-1}$$

$$\left(\rho = \frac{\lambda}{s\mu}\right)$$

To obtain the value for ρ we will calculate **λ** and **μ** from our data. Averaging the arrival counts over the 14-day period gives a mean arrival rate of **λ = 44.616**. From the enquiry duration data, we are given the service time in minutes. To calculate the service time in hours the average of these values is taken and divided by 60. The service rate is calculated by taking 1 over the service time, giving a value of **μ = 5.143**.

To estimate the number of servers required for this desired percentage, the calculation for the probability of an immediate call stated above is used for each server amount, starting with the number of servers which satisfies steady state, until the probability of an immediate call is above or equal to 0.4.

The number of servers required to satisfy steady state for this system is the number that in which **λ/sμ < 1**. Rearranging this for s gives a value of s needing to be greater than 8.675. The nearest whole number which satisfies this is **9**.

The calculations for the steady state and immediate call probabilities are done using Python. The probabilities of an immediate call are:

Number of Servers	P(Immediate Call)
9	0.121
10	**0.427**

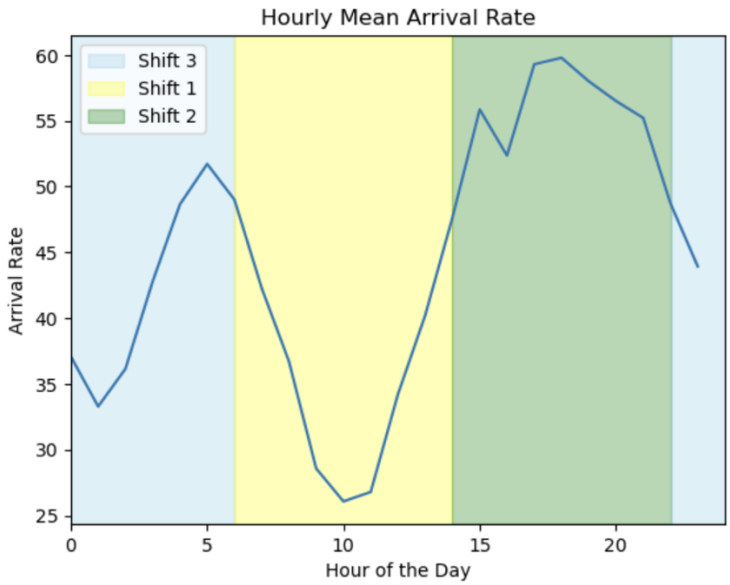
The probability of an instant call with 10 servers is 0.427 or 42.7%, which is above our desired percentage meaning this number of servers is sufficient.

1. (b)

The total number of servers we must allocate is 10 * 3 = 30. To have an idea where to allocate these, both the average arrival rate per **shift** and per **hour** should be calculated.

The values for these are:

Hour of Shift	Shift 1	Shift 2	Shift 3
1	49.000	47.643	48.714
2	42.286	55.857	43.929
3	36.714	59.357	37.071
4	28.571	59.286	33.286
5	26.071	59.786	36.143
6	26.786	58.000	42.786
7	34.214	56.500	48.643
8	40.214	55.214	51.714
Shift Average	**35.482**	**55.580**	**42.786**



From both the hourly and shift arrival rates we can likely see that the greatest number of servers need to be allocated into shift 2 followed by shift 3 and then lowest to shift 1.

We will use numerical integration in Python to identify the following metrics of the system at different times to determine the ideal allocation of servers:

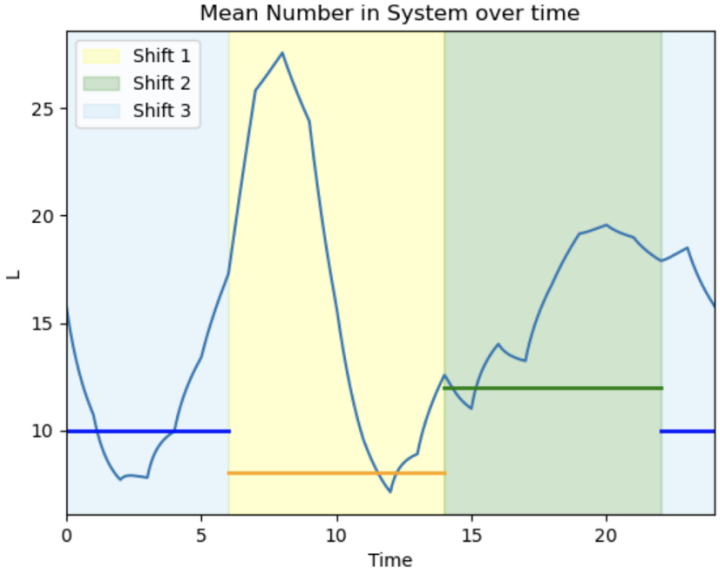
- Number of customers in the system / queue (L / L_q)
- Waiting time in the queue (W_q)

The numerical integration model takes the parameters of λ, μ and s at different times. For these examples, we will use the arrival rates per hour rather than per shift for increased insight. The time step used is a low value of 0.001 due to the frequent changes in the parameters. At every time step, the probability of each state is calculated for which the mean number in the system can be calculated across many repetitions. The mean waiting time can then be calculated using Little's Law. As the model requires a maximum size for the queue, using a value high enough such as 200 is almost sufficient to approximate an infinite queue which is one of the assumptions being made, yet this is a limitation.

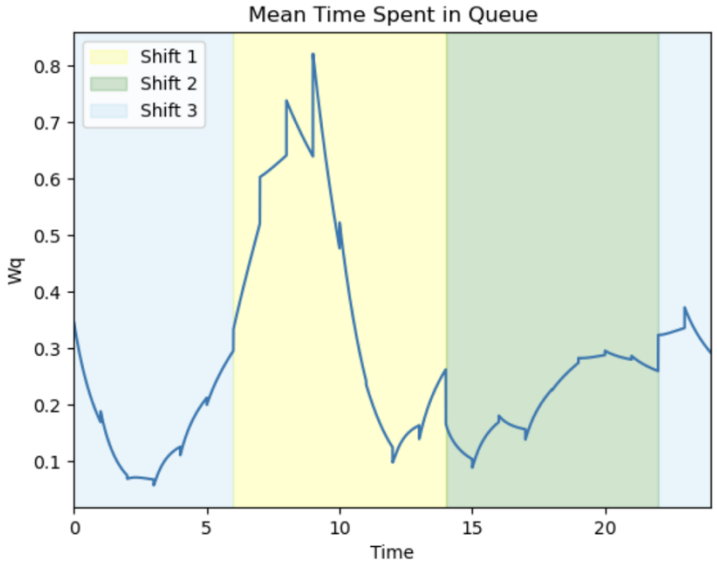
Allocation 1:

- Shift 1 – 8 servers
- Shift 2 – 12 servers
- Shift 3 – 10 servers

Shift	Mean in System (L)	Mean Time in Queue (W_q)
Shift 1	16.721	0.408
Shift 2	16.054	0.214
Shift 3	12.663	0.201
Overall Average	15.146	0.274



The coloured horizontal lines represent the servers assigned in the enclosing shift (**server capacity**). From this integration we can see that shift 1 has the highest average L and the highest peak despite having the lowest average arrival rate. This is due to the arrivals being very high early in the shift in comparison to the rest of the shift. This leads to the number of customers in the system growing very fast in this period. The arrival rates at 6am and 7am are 49.0 and 42.286 respectively, which 8 servers cannot sufficiently handle, with ρ being above 1 in both periods causing the queue to grow quickly.



This can also be seen in the waiting time for the queue in this system, with a very high peak around 7am/8am. As for shifts 2 and 3, they have similar average wait times which are more consistent across the shift due to the arrival rate per being more manageable for the server count.

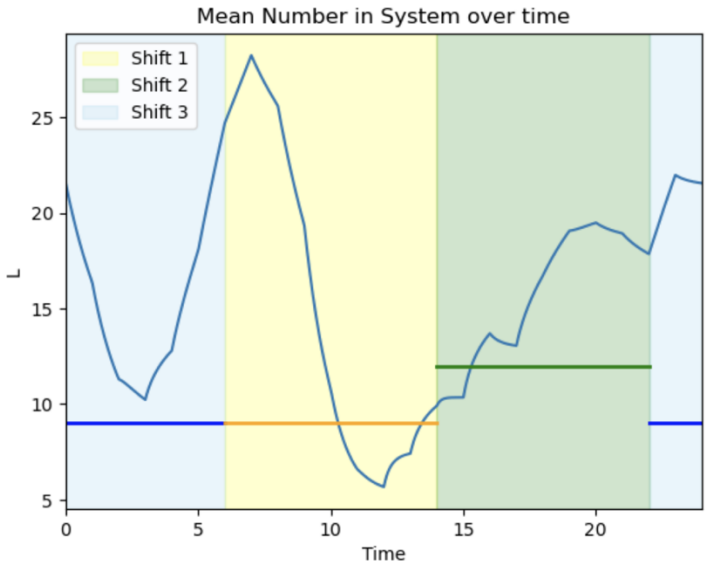
Seen in the mean number in the system integration, shift 3 goes under server capacity for a significant amount of time and shift 2 doesn't, therefore an extra server assigned to shift 1 from shift 3 could be an improvement to the waiting times.

Allocation 2:

- Shift 1 – 9 servers
- Shift 2 – 12 servers
- Shift 3 – 9 servers

Shift	Mean in System (L)	Mean Time in Queue (W_q)
Shift 1	15.04	0.314
Shift 2	15.735	0.205
Shift 3	16.688	0.335
Overall Average	15.821	0.285

In comparison to the first allocation, both L and W_q are slightly higher, but the maximum waiting time decreased, and the spread is much lower.



From this integration it still stands that the peak in the system is around the same time, just slightly earlier. This is because the arrival rate at the end of shift 3 at 5am is very high at a value of 51.714 which even 9 servers cannot handle which the queue size growth continues into the early hours of shift 1.

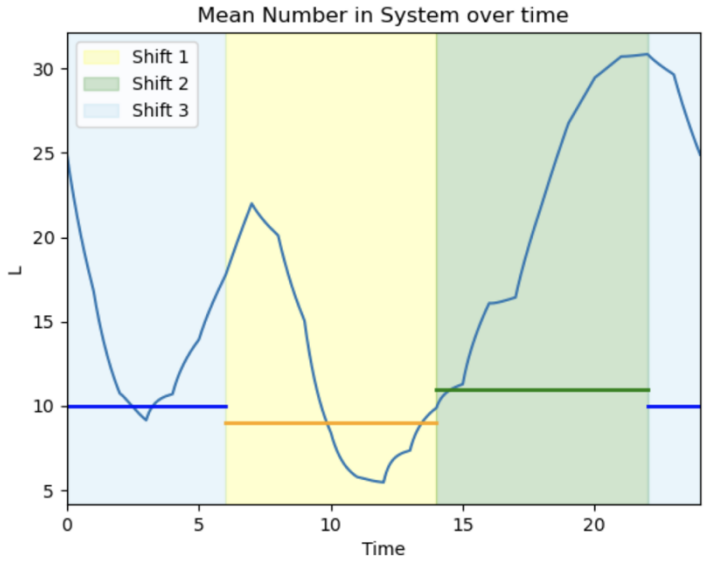
The waiting time decreases slightly in shift 1, but the increase in shift 3 is noticeable. The period in-between shift 3 and shift 1 is difficult to create a solution for because increasing the servers causes the other parts of both shifts to fall below capacity on average, while at the same time having very long waiting times at other parts of the shift. To try to alleviate this situation taking a server from shift 2 and giving it to shift 3 may help to stabilize this peak by decreasing the queue before entering shift 1.

Allocation 3:

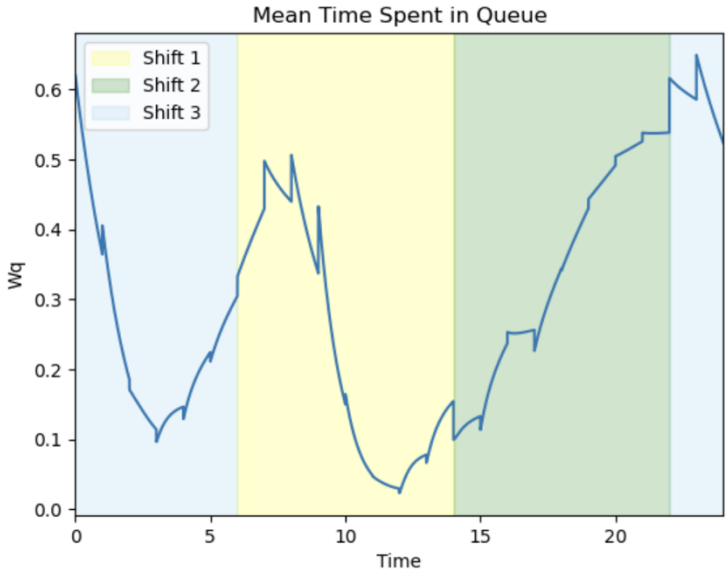
- Shift 1 – 9 servers
- Shift 2 – 11 servers
- Shift 3 – 10 servers

Shift	Mean in System (L)	Mean Time in Queue (W_q)
Shift 1	12.191	0.230
Shift 2	21.729	0.344
Shift 3	17.15	0.334
Overall Average	17.15	0.303

This change causes both the average L and W_q to rise noticeably. Simply by these numbers it appears to be a bad decision, however looking at the integration in detail may give some more insight:



This change in the number of servers improves the early hours of shift 1, at the cost of shift 2 having a faster growing queue. This also leads to shift 1 being under capacity more on average during the second half.



As shift 2 has the highest number of servers, the waiting time is closer to average despite the higher number in the system. This allocation of servers handles the transition between shifts the most appropriately, with the biggest downside being the later end of shift 2 being very busy.

The decision on the best allocation depends on how much priority you give to having a lower average waiting time, compared to a slightly higher average waiting time with lower peaks. All 3 allocations could be the most effective with different goals, but due the difference in peak waiting time being small – as well as the it being at capacity more frequently – the 2nd allocation of servers (9,12,9) is what we believe to have the best balance for controlling the systems calls and waiting times the best way.

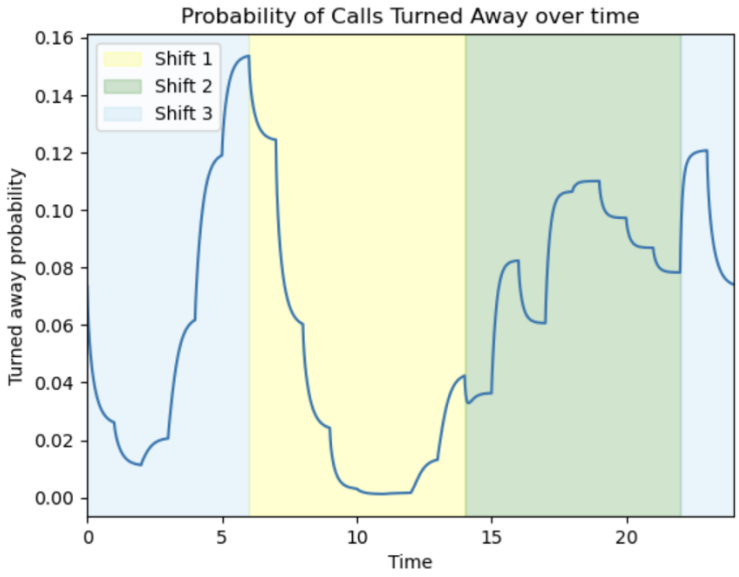
1. (c)

Using the same model, numerical integration can be used while adding a maximum limit to the queue. The probability of calls being declined can be defined as the probability of the system being in its maximum state, which can also be graphed. Allocation 2 (9,12,9) of servers from part (b) will be used for the tables and graphs.

15 Capacity:

Shift	Mean in System (L)	Mean Time in Queue (W_q)
Shift 1	7.535	0.084
Shift 2	10.389	0.066
Shift 3	9.2	0.127
Overall Average	9.041	0.092

The L and W_q are significantly lower than before the queue limit were introduced, with the means of both shift 1 and 2 being lower than their assigned servers.



The probability of a call being declined reaches its maximum at the very end of shift 3, which would be 6:00am to the nearest minute. The probability at this time is 0.154.

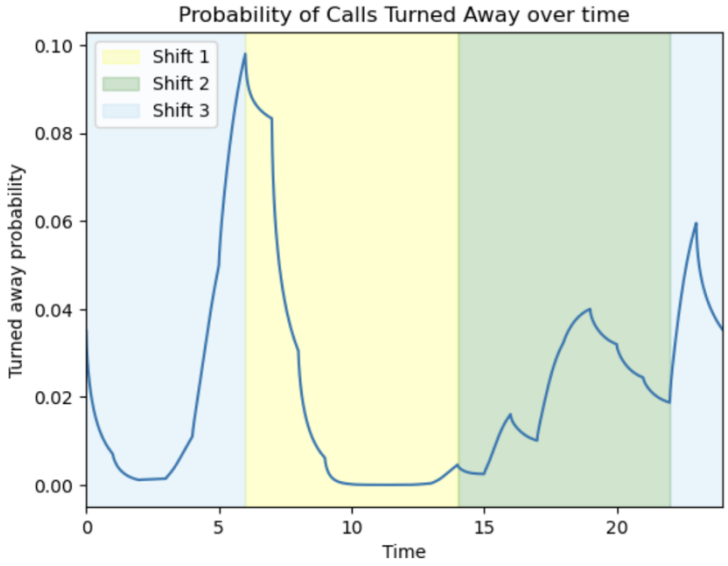
- Allocation 1 (8,12,10) - Maximum at 7:00am with value 0.186
- Allocation 3 (9,11,10) - Maximum at 7:00pm with value 0.140

When exploring these values for the other allocations, it appears that allocation 3 may be most appropriate for this maximum capacity due to its balancing of the maximum waiting time as previously mentioned in part (b).

25 Capacity:

Shift	Mean in System (L)	Mean Time in Queue (W_q)
Shift 1	9.738	0.154
Shift 2	13.287	0.154
Shift 3	12.27	0.217
Overall Average	11.765	0.175

The values for L are now all above server capacity, and the values for W_q almost doubled for this increase in capacity. These values are still noticeably lower than without the capacity, meaning there are still many calls being declined.



Similarly to the previous capacity, the turned away probability reaches its maximum value at 6:00 am - with a value of 0.098.

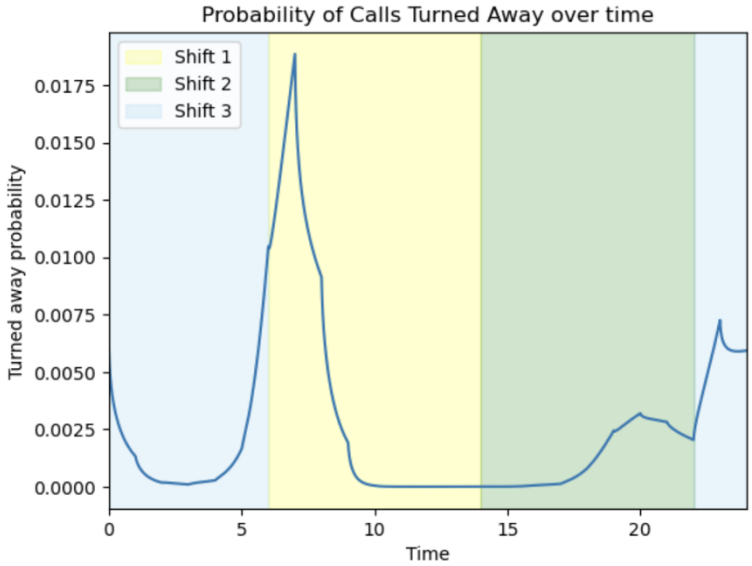
- Allocation 1 (8,12,10) - Maximum at 7:00am with value 0.131
- Allocation 3 (9,11,10) - Maximum at 7:00pm with value 0.080

This increase in capacity does not actually decrease the maximum declined probability as much as expected, with it still being more than half the value of 15 capacity. However, values outside this maximum decreased quite noticeably as seen on the graph, with certain segments having a probability of almost 0.

50 Capacity:

Shift	Mean in System (L)	Mean Time in Queue (W_q)
Shift 1	13.923	0.279
Shift 2	15.58	0.202
Shift 3	15.891	0.315
Overall Average	15.132	0.265

The increase in capacity this time brings L and W_q up to a value which is fairly close to the approximated infinite queue. It is also at this capacity at which this allocation of servers performs the most strongly on these metrics.



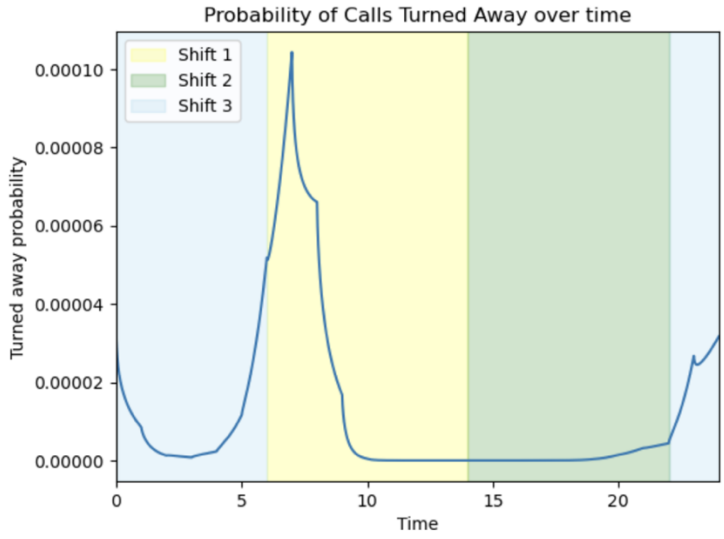
This new capacity has a different time where the probability of calls being turned away is at its maximum – 7:00am rather than 6:00am, potentially due to the system needing more time to grow the queue with this larger capacity. The value decreases significantly here, down to just 0.0188.

- Allocation 1 (8,12,10) - Maximum at 8:00pam with value 0.0169
- Allocation 3 (9,11,10) - Maximum at 8:00pm with value 0.0205

100 Capacity:

Shift	Mean in System (L)	Mean Time in Queue (W_q)
Shift 1	15.03	0.314
Shift 2	15.733	0.205
Shift 3	16.685	0.335
Overall Average	15.816	0.285

The values of L and W_q for this capacity are almost identical to those when assuming an infinite queue size, due to the almost zero probability of the system getting to this size.



With this final capacity, the probability of a call being declined is almost 0 for the entire day. The peak is once again at 7:00am but this value is only 0.000104. In comparison to a capacity of 50, the difference is so small that the linear increase in cost is very likely not worth the investment.

- Allocation 1 (8,12,10) - Maximum at 9:00am with value 0.000124
- Allocation 3 (9,11,10) - Maximum at 11:00pm with value 0.000355

Out of these capacities – the two extremes of 15 and 100 seem to both be very unfit for the system. For 15, the probability of calls being declined is too high and there are often times where this harms efficiency when there are large changes in arrival rate. 100 capacity is not worth the doubling of cost of 50 capacity just for a 1-2% maximum decrease in declined calls.

For the choice between 25 and 50, the decision comes down to the preference of controlling waiting time vs minimizing declined calls. The 50 capacity has in our opinion the best tradeoff between cost and declined calls, however it will still have fairly long waiting times during certain times of the day due to calls very rarely being declined. 25 capacity controls the waiting time very well, at the cost of declining calls during busy times. For the choice we made, we chose to prioritize decreasing calls declined therefore our recommendation would be 50, however 25 is still also a very valid option.

Assumptions made in question 1:

- The queue is infinite (parts (a) and (b) only).
- All servers handle calls at the same rate.
- Customers do not leave the queue.
- Customers arrive according to exponential distribution, which is memoryless.
- The transition of servers between shifts is instant.
- As soon as a server completes a call, the next call starts

2. (a)

To calculate long-run profit, we will have to find the steady state of the system. Each service can be modelled as a Markov chain where the state is equal to the number of bookings. The transition of states is equal to:

$$State = \min(5, Last\ state - \min(Vans, Last\ state) + Calls)$$

The last state is subtracted by the jobs done, which is the minimum of the last state (bookings) and the vans available for that service. The calls received are then added, while keeping the state to a minimum of 5 due to the maximum bookings limit.

Assuming the number of requests/calls arrives as a Poisson process, we can determine the probability distribution of these calls according to the mean of the data.

The mean number of calls (λ) for each service are:

- Self-Drive: 2.892
- Full Removal: 2.046

To now calculate the probability of calls, the formula for Poisson probability distribution can be used:

$$P(Calls = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

One thing to note about these services is that since the maximum number of bookings is 5, receiving more than 5 calls would be equivalent to receiving 5 in all cases. Therefore, the probability of this many or more calls is equal to (1 – P(Calls <= 4)) The probabilities of calls for each service were calculated in Python using this method:

Calls	Self-Drive Probability	Full Removal Probability
0	0.0554	0.129
1	0.160	0.264
2	0.232	0.271
3	0.236	0.185
4	0.162	0.0944
5 +	0.167	0.0569

The transition matrix for both services can be defined using these probabilities. The probability of going from state i to state j can be defined as:

If j <= 4:

$$P(i, j) = P\left(Calls = \left(j - (i - jobs\ done)\right)\right)$$

If j = 5:

$$P(i, j) = P\left(Calls \geq \left(j - (i - jobs\ done)\right)\right)$$

Using this formula, the transition matrix for each service are as follows:

Self-Drive

State	0	1	2	3	4	5
0	0.0554	0.160	0.232	0.236	0.162	0.167
1	0.0554	0.160	0.232	0.236	0.162	0.167
2	0.0554	0.160	0.232	0.236	0.162	0.167
3	0.0554	0.160	0.232	0.236	0.162	0.167
4	0	0.0554	0.160	0.232	0.236	0.329
5	0	0	0.0554	0.160	0.232	0.552

Full Removal

State	0	1	2	3	4	5
0	0.129	0.264	0.271	0.185	0.0944	0.0569
1	0.129	0.264	0.271	0.185	0.0944	0.0569
2	0.129	0.264	0.271	0.185	0.0944	0.0569
3	0	0.129	0.264	0.271	0.185	0.151
4	0	0	0.129	0.264	0.271	0.336
5	0	0	0	0.129	0.264	0.606

Using these transition matrices, the steady state distribution of both services can be calculated. This is done in Python.

Self-Drive

π_0	π_1	π_2	π_3	π_4	π_5
0.0266	0.0879	0.161	0.205	0.197	0.323

Full Removal

π_0	π_1	π_2	π_3	π_4	π_5
0.0403	0.108	0.163	0.202	0.196	0.290

Now that we have the steady state distribution, the long run profit can be calculated using the expected profit in each state, multiplied by the steady state probability of that state.

$$Expected\ \mathrm{Pr}o\,fit\ =\ \sum_{n=0}^{5}(Income_n\ -\ Discount_n)\pi_n$$

Where the income and discount are defined as:

$$Income\ =\ \min(vans,\ n)\ \cdot\ income\ per\ job$$

$$Discount\ =\ (n\ -\ \max(0,\ n\ -\ vans\ ))\ \cdot\ discount\ amount$$

Using this formula in python gives an expected profit per day of:

- Self-Drive: £90.69
- Full Removal: £79.35
- **Total Expected Profit: £170.05**

To calculate the proportion of days that all the vans are busy, we need to look at the probability that the state of each service is greater than or equal to the number of vans.

$$P(Vans\ Busy)\ =\ \sum_{n\ =\ vans}^{5}\pi_n$$

- Self-Drive: (0.205 + 0.197 + 0.323) = 0.725
- Full Removal: (0.163 + 0.202 + 0.196 + 0.290) = 0.851

We can calculate the probability of all vans being busy by multiplying these two.

P(All Vans Busy) = 0.725 * 0.851

P(All Vans Busy) = 0.617 (61.7% of days)

Finally, to calculate the proportion of days' work might be turned away, we need to look at the probabilities of the calls being at an amount equal or greater to the amount that would bring the state after completing jobs above 5. For each service this would be:

State	Calls Required (Self-Drive)	Calls Required (Full Removal)
0	6	6
1	6	6
2	6	6
3	6	5
4	5	4
5	4	3

To calculate the probability of this requirement, we will again use Poisson distribution formula, taking 1 minus the cumulative sum of call probabilities lower than this requirement.

State	Turned Away Probability (Self-Drive)	Turned Away Probability (Full Removal)
0	0.0735	0.0183
1	0.0735	0.0183
2	0.0735	0.0183
3	0.0735	0.0569
4	0.167	0.151
5	0.329	0.336

Taking the sum of these probabilities multiplied by the steady state probability of the corresponding state gives values of:

- Self-Drive: 0.174
- Full Removal: 0.133

For the probability of work being turned away for at least one service, we will find the union of these two probabilities.

P(Work Turned Away) = 0.174 + 0.133 - (0.174 * 0.133)

P(Work Turned Away) = 0.144 (14.4% of days)

2. (b)

Considering this is a finite time horizon, we can use a Markov decision process to identify the policy at each stage which maximizes the overall profit. The policy will be defined by a 6x6 matrix, with the value of policy[i, j] being the number of vans hired when self-drive is at state i (i bookings) and full removal is at state j (j bookings). This allows us to look at the combination of bookings, since the decision to hire a van for one service should not only depend on its own bookings, but the other services' bookings – due to the fact that the hired vans are limited and shared.

The method to find the maximized profit at each stage would be Bellman's optimality equation. Defined as:

$$V_n(i) = \max_a \left\{ \sum_{j \in S} p_{ij}(a)[r(i,a,j) + V_{n-1}(j)] \right\} \forall i \in S,\ n \in \{1,2,\dots,T\}$$

To calculate p_{ij} (a), we will need to use a similar method for the transitional probabilities from before, however the number of vans hired will affect the calls required to transition into state j. We simply need to add the number of hired vans to the current number of vans when

calculating the number of jobs done in state i. Since we are now working with both services at the same time, the transitional probabilities will be calculated for all combinations of bookings of each service, with probabilities being multiplied to calculate the probability of both transitions happening.

To calculate r(i, a, j) we need to consider not only the profit made for completed jobs minus the van hire cost, but the discounts caused by being over the van count. If we hire vans, we will end up in a lower state j on average, therefore it is also less likely for discounts to build up in the future. The reward function is defined as followed:

$$r(i,\ a,\ j) = (\min(i,\ vans + a) \cdot income\ per\ job) - (a \cdot hiring\ price) - discount$$

$$discount = \left(\max\left(0,\ (i - vans + a)\right) \cdot discount\ amount\right)$$

One potential limitation of this is that since we are treating this problem as a Markov process, we only look at the current state and previous state, therefore it is difficult to fully account for the expected loss in discounts while only looking at these subsequential days.

The model made in Python uses these calculations, while iterating through every possible combination of action for each service which doesn't exceed 2 (hired van limit). We start from V_0 in which the rewards are set as 0 (terminal rewards). For every state combination of self-drive and full removal, the reward is calculated using the formula (for each action combination). The reward is stored whenever it becomes the new maximum, and the action for each service corresponding to the maximum reward is defined in their respective policy matrices for this stage. The value of this maximum is stored as V_n for each combination of states. Finally, this iterates for 28 days and the policy matrix for both self-drive and full removal is output.

An assumption made in this model is that, at the final stage (28th day in this case) there is no more work to be done in the future, therefore the discounts will be set to 0.

With the original van hire cost, the optimal policy found for these 28 days is the following:

(Rows: Self-Drive bookings, Columns: Full Removal bookings, Value: Hired vans for this service)

Self-Drive
Days 1 – 28

State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Full Removal
Days 1 – 28

State	0	1	2	3	4	5
0	0	0	0	1	2	2
1	0	0	0	1	2	2
2	0	0	0	1	2	2
3	0	0	0	1	2	2
4	0	0	0	1	2	2
5	0	0	0	1	2	2

From this we can see it is never worth hiring a van for self-drive. For full removal you should hire as many vans as possible, likely because the income from jobs is higher than the hiring cost, while for self-drive the income is lower than the hiring cost with the future potential gain not overcoming this.

2. (c)

At the end of the data period on day 130 there are 5 bookings remaining for self-drive and 2 bookings remaining for full removal. Now that we have a policy for 28 days, we can simulate this period alongside this initial 5 days before the policy is introduced.

To simulate the activity of these services, a model built in excel will be used. This model not only tracks the profit of each day, but van usage and calls turned away. The calls are distributed from a Poisson approximation using binomial with high n. We can update the bookings per service using the state calculation mentioned in part (a). After determining how many vans should be hired using a grid search based on our pre-defined policy, the jobs completed, and their income can be calculated. The status of these jobs – and whether they are 1 or 2 days late - is also taken into account so we can add appropriate discounts.

The results of these simulations come from averages taken over 1000 replications using a macro made in VBA. The “expected operating profit” is taken from the days where the policy is **active,** meaning the average of the 28 days where the policy is defined.

Result of simulation:

Expected Operation Profit = £183.20

2. (d)

For the range of the hiring price, we will decrease in increments of £5 and identify the changes in policy and profit.

Van Hire cost: £45

As expected, the full removal policy remains the same, however the self-drive policy changes as so:

Self-Drive

Days 1 – 27

State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	1	1	1	1	0	0

Day 28

State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

From this we can see that after this decrease in price, it is now worth hiring a van when we have 5 bookings – but only in the case where full removal also has 3 or less bookings. This is due to full removal having higher priority, so it will take the hired vans when both services have high bookings. Despite the van hiring price still being higher than the income, it appears that the potential gain from ending up in a lower state (more jobs possible to take or less discounts incurred) outweighs this short-term loss. On the final day, it is no longer worth hiring any vans for the self-drive service. This is due to the potential future gain being negated by the horizon ending.

Expected Operating Profit: £186.95

Van Hire cost: £40

Both policies remain the same as £45 for this price, however there is one thing to consider. The hiring cost is equal to the income of self-drive, therefore on the last day when assuming no further discounts, hiring a van would give the same expected reward as not hiring a van. Therefore, we could potentially hire as many vans as possible for self-drive if full removal has less than 3 bookings, but it would be equivalent to not doing so in terms of profit. But in a real-world situation, completing jobs rather than not could be worth it if you want to satisfy a customer considering equal profit outcomes.

Expected Operating Profit: £191.79

Van Hire cost: £35

This new hiring cost changes the policy for both self-drive and full removal, as follows:

Self-Drive

Days 1 – 26

State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	1	1	1	1	0	0
5	2	2	2	1	1	0

Days 27-28

State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	1	1	1	1	0	0
5	2	2	2	1	0	0

Full Removal

Days 1 – 26

State	0	1	2	3	4	5
0	0	0	0	1	2	2
1	0	0	0	1	2	2
2	0	0	0	1	2	2
3	0	0	0	1	2	2
4	0	0	0	1	2	2
5	0	0	0	1	1	2

Days 27-28

State	0	1	2	3	4	5
0	0	0	0	1	2	2
1	0	0	0	1	2	2
2	0	0	0	1	2	2
3	0	0	0	1	2	2
4	0	0	0	1	2	2
5	0	0	0	1	2	2

We have now reached the threshold for self-drive where it is always worth hiring a van whenever over capacity and full drive is not. The difference in policy for the last 2 days in comparison to the rest is subtle, but interesting. When self-drive has 5 bookings and full removal has 4 bookings – we originally chose to split the hired vans 1 to 1 per service for the first 26 days. However, for the final 2 days, it is best to continue to give both 2 hired vans to full removal. The reason for this could be because towards the end of our horizon, we want to prioritize having a low number of full removal bookings, so we do not lose their income after the horizon ends.

Expected Operating Profit: £195.10

Van Hire cost: £30

At this price the policy does not change noticeably, except for the policy of the cost of £35 which lasted between days 27-28 being extended to days 26-28. Since there are no more significant changes to be made, no lower hiring costs will be examined.

Expected Operating Profit: £199.70

Assumptions made in question 2:

- The system follows the Markov property; therefore, the future state depends on only the current state.
- Calls arrive according to a Poisson distribution process.
- Call probabilities are fixed for both services and are equal on all days.
- There are no additional costs aside from hiring a van or giving out a discount.
- Jobs are completed in the order of the day that they were received, meaning late jobs will have priority over new jobs.
- Customers can not cancel their requests if they are late.
- Discounts are not counted at the final stage of the time horizon when carrying out the decision process.

3. **We need to determine:**
    a. How many couriers should be employed?
    b. How many self-service order machines should be installed?

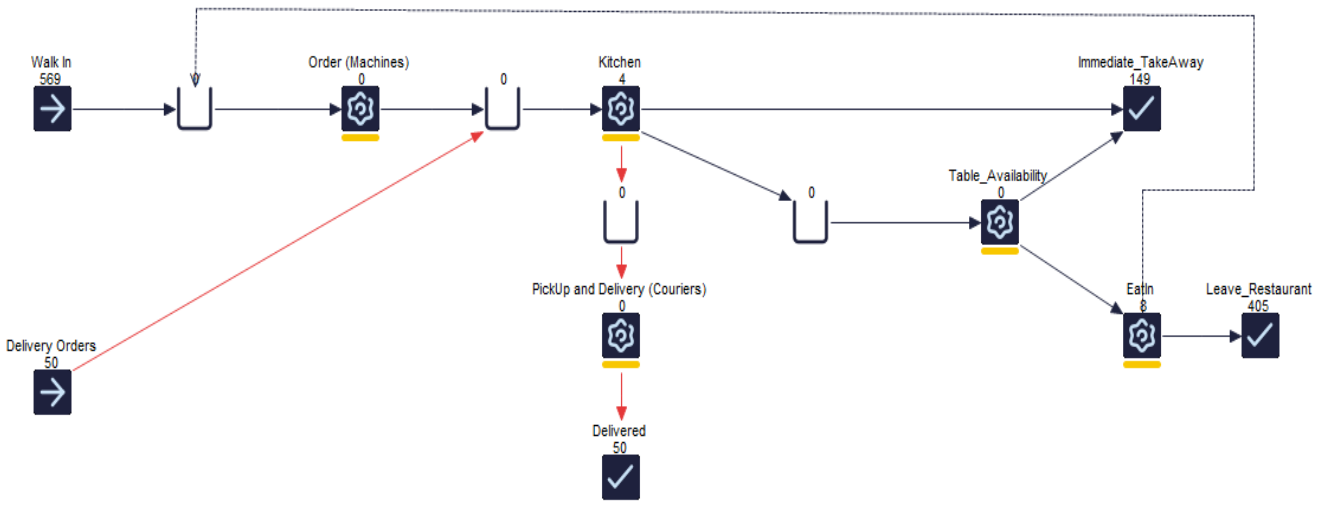
KPI's:

a. Percentage of delivery food orders delivered within 1hour of ordering
b. Time between arrivals and receiving food for walk in customers
c. Expected number of orders completed per day

Assumptions made for this model are as follows:

a. All calculations and simulations are run at 95% significance level.
b. We assume that the inter-arrival times follow an exponential distribution.
c. Customers do not leave the ordering queue, and the queue does not extend till infinity.
d. Since the restaurant wants to retain some tables for the sit in arrangement, the maximum number of self-service machines we can install is 14.
e. The queue for all orders going to the kitchen is the same, however, the queue for delivery orders to be collected from and the queue for orders to be collected from for in restaurant is different.
f. One courier only makes one delivery at a given time.
g. We assume that after delivery of an order, the courier just returns to the restaurant which is the same as the time taken by him to deliver the order. When the courier is needed to pick up another order - only then do they walk in to pick up the order and walk back to the vehicle - i.e. 5 minutes. Hence per order he would follow: **5 minutes + (2*time taken) to deliver the order**.
h. Collecting food and eating it is considered under 'Sit in' times.
i. The activity block for delivery accounts for the pick-up time, time to walk back to the vehicle, delivery time and return to the restaurant.
j. Each table in the restaurant seats only 1 customer.
k. The first takeaway point named "Takeaway" is used for KPI selection because we assume that it indicates that people have received their food and now split into "Sit in" or "Takeaway".

Conceptual Model:



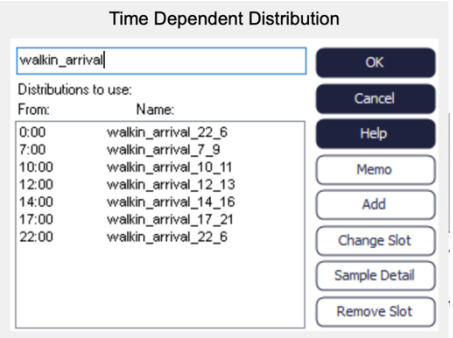
a. **Distributions:**

- **Walk In Arrivals:**

We begin by defining the different distributions in the model that we will be using. For walk in arrivals, we take the hourly average across all 14 days and notice that for certain time periods the arrival rates are quite close to one another. We group these time periods together and find the arrival rate per shift. We now use this to find the average in minutes. Following are the shifts we determined, their arrival rates per hour and their average in minutes:

Shifts	Arrival Rate (Per Hour)	Average Time (Minutes)
7am – 10am	32.80952	1.829
10 am – 12pm	14.21429	4.221
12pm – 2pm	46.78571	1.282
2pm – 5pm	14.14286	4.242
5pm – 10pm	47.9	1.253
10pm – 7am	10.5	5.714

We build a “Named Distribution” following exponential distribution with the average per minute for each of the shifts. We then create a “Time Dependent Distribution” and based on the shift start time, configure each named distribution in it.



We now use set the distribution of the "Walk In" block to "walkin_arrivals", our time dependent distribution.

- **Delivery Order Arrivals**

For the delivery orders arrivals, we know that it is open from 4pm-9pm so we first take the average of the total number of arrivals per day for all 14 days. Since the number of arrivals for delivery orders is 10% of the number of arrivals for walk in orders, we find 10% of the average arrival rate between 4pm-9pm for 14 days. Thus, we find that the following about the interarrival time for delivery orders.

Shifts	Arrival Rate (Per Hour)	Average Time (Minutes)
4pm – 9pm	11.93714	5.025

So, for the "Delivery Orders" block we set the distribution in the timing section to exponential with an average of 5.025 minutes.

- **Food Preparation Time:**

We use Stat::Fit to evaluate the distribution followed by the time taken to prepare food when a customer orders for the first time i.e. 'Food Prep time 1'. Stat::Fit recommends Pearson 5 and Lognormal distribution, however the rank, fit and aicc probability for the Pearson 5 model are more accurate and higher, hence we choose to use it. We therefore build a "Named Distribution" called "prep_time1" which follows this Pearson 5 Distribution.

Similarly, we use Stat::Fit to evaluate the distribution for the time taken by the kitchen to prepare the food if a person reorders. i.e. 'Food Prep time 2'. As per Stat::Fit's analysis, we use Pearson 6 distribution for modelling 'Food Prep time 2'. We build a "Named Distribution" called "prep_time2" which follows a Pearson 6 Distribution.

Following are the parameters for both distributions:

	Distribution	Parameters
Food Prep Time 1	Pearson 5 Distribution	Alpha (α): 4.09 Beta (β): 2.56
Food Prep Time 2	Pearson 6 Distribution	Alpha1 ($\alpha 1$): 12.0845 A lpha2 ($\alpha 2$): 5.91903 Beta (β): 2.10614

- **Routing delivery orders to couriers and in restaurant orders to takeaway/sit in**

To make sure that all the orders being processed in the kitchen which have been ordered online get routed to the couriers, we define another distribution with a probability profile such that 100% of the orders being ordered online are routed to the couriers. We call this distribution "delivery_prob".

If a customer is ordering for the first time, we see that 18.88% of customers takeaway their order and 81.12% customers prefer to sit in. So, we create a Probability Profile Distribution to account for that and call it "eat_in_or_take_away"

Similarly, if a customer has reordered, we see that 17.57% of customers take away their second meal and 82.43% of customers sit in for the second time. For this as well we use a Probability Profile Distribution to account for this and name this "second_eat_in_or_take_away".

Following is the description of the distributions:

	Distribution	Parameters
Delivery_prob	Probability Profile Distribution	1 (Takeaway): 0% 2 (Sit In): 0% 3 (Couriers): 100%
eat_in_or_take_away	Probability Profile Distribution	1 (Takeaway): 18.88% 2 (Sit In): 81.12% 3 (Couriers): 0%
second_eat_in_or_take_away	Probability Profile Distribution	1 (Takeaway): 17.57% 2 (Sit In): 82.43% 3 (Couriers): 0%

- **Delivery Time:**
  Based on the data we have; it is said that the delivery takes 15 minutes and is usually between 8-12 minutes. For this reason, we use a 'Named Distribution' called "delivery_distribution" which follows a Beta Distribution.

	Distribution	Parameters
delivery_distribution	Beta Distribution	Alpha1 ($\alpha 1$): 2.5 A lpha2 ($\alpha 2$): 3 Min: 6 Max: 15

- **Sit in Time:**

We use Stat::Fit to evaluate the distribution followed by the time customers spend in the restaurant to sit in. Stat::Fit recommends Weibull and Beta distributions but the rank and aicc probability of Weibull is better, so we conclude that 'Sit in time 1" follows a Weibull distribution named 'first_eatin'.

Similarly, when we evaluate the time for which customers sit in the restaurant after they reorder, Stat::Fit recommends various models but the best one we use is Weibull distribution and we name it as 'second_eatin'.

	Distribution	Parameters
first_eatin	Weibull Distribution	Alpha (α): 5.07 Beta (β): 39.6 Min: 0
second_eatin	Weibull Distribution	Alpha (α): 5.72581 Beta (β):20.1689 Min: 0

- **Reordering:**
  Based on our data we can see that almost 24% of the customers reorder while 76% of customers do not. To record this, we create a probability profile distribution, and we call this 'second_order'.

	Distribution	Parameters
second_order	Probability Profile Distribution	1 (Reorder): 24.3% 2 (Leave the Restaurant): 75.7%

We now use these distributions to create labels and assign them to the various blocks in our model:

b. Labels:

We will now be defining some labels to define their function in a block.

- **lbl_eatin_time:**
  This label is assigned to the element related to entry point for in restaurant walk ins. This label action is set to the sit in time **(distribution 'first_eatin')** when customers are ordering for the very first time. We set this to walk in entry point because that's the only way that the customers will use first time sit in time.
  Under the same label, it is also associated with the element related to the activity where customers sit in the restaurant after reordering. The action related to this object is the sit in time **(distribution 'second_eatin')** when customers are reordering. We assign this to eat in activity because people route back into the ordering queue after they have already sat in and eaten in the restaurant.
  We will be using this label to set the timing section for the activity associated with customers eating in.

- **lbl_prep_time:**
  This label is associated with three objects – Entry point for walk ins, Entry point for delivery order and activity related to eating in the restaurant.
  For the entry point for walk ins, we set the action to the time distribution related to the kitchen food preparing time for the people ordering for the first time (**distribution 'prep_time1').**
  For the entry point for delivery, we set the action to the time distribution related to the kitchen food preparing times for those customers who have ordered for the first time (**distribution 'prep_time1').** We use this since people who have ordered online are also ordering for the first time.
  For the eating in action, we set the action to the time distribution related to the kitchen food preparing times for those customers who have reordered (**distribution 'prep_time2').** We do this cause people who reorder are routed back into the queue after they have eaten in for the first time.
  We use this label to set the timing section for the block associated with food preparing i.e. kitchen.

- **lbl_reorder:**
  For this label, it is associated with the same objects as lbl_prep_time.
  For the walk in entry point, we set the action to deciding whether they want to sit in or takeaway **(probability distribution 'eat_in_or_take_away').**
  For the delivery entry point, we set the action to route all the delivery orders to the couriers after it is processed in the kitchen (**probability distribution 'delivery_prob')**
  For the eat in activity, we set the action to deciding whether the customer wants to sit in or takeaway after they have reordered (**probability distribution 'second_ eat_in_or_take_away').**
  We use this label to set the routing out for orders that are in the restaurant and those that are for delivery.

c. Working:

There are two entry points for this model – **"Walk In"** and "**Delivery Orders".** The "walk in" handles the orders from people who come into the restaurant and the timing is set to the time dependent distribution of "walkin_arrival". The "Delivery Orders" handle the orders coming in from online orders. Since the orders for delivery are equal to 10% of the total number of orders received by walk-ins, we find the total arrivals per day for 14 days, average it and divide by 5 because the option is only open from 4pm-9pm (5hours). Then we use that value and find its 10% and take its reciprocal and multiply by 60 to get a timing of ~5.025 minutes.

$$x = Average(daily\ sum\ of\ hourly\ arrival\ rate\ for\ 14\ days)$$

$$y = \frac{0.1 * x}{5}$$

$$timing = \left(\frac{1}{y}\right) * 60\ minutes = \mathbf{5.025\ minutes}$$

Now we use this to set the timing section for the delivery entry point with the distribution of 'exponential distribution'.

When customers come in from walk in, they get routed to the queue before they can order. In the "**Order (Machines)"** activity block, we set the timing of the block to follow Weibull (0,4.09,2.56) as we use stat fit to evaluate the 'self-service times'. We also use 'replicate' to set the number of resources for this block. Since we need to find the number of machines needed, we change this value iteratively and compare it with the KPIs to conclude on the number of machines that should be installed.

After orders are received from the customers, they are queued before they enter the kitchen block. This queue also receives the orders from delivery. So, in conclusion, all orders queue up in this block before they go to the kitchen for preparation.

For the "**kitchen"** block, the timing is set to the label 'lbl_prep_time'**.** This label checks the entry point of the order and accordingly uses the food preparation time. If the order is coming from "walk in block" and "delivery orders block", it sets the value to 'food_prep1' and if the orders are from "eat in block" it sets it to food_prep2. Further, for the routing out, it is done by label "lbl_route**"** to ensure that all orders that are made by online order are routed out to the couriers and do not enter the in-restaurant system. This label also ensures that based on the probability distribution 'eat_in_or_take_away', customers are routed to directly taking their order and leaving the restaurant or eating in.

From the kitchen the orders that get routed out to the couriers go to the delivery queue. From here we set up the activity block "**PickUp and Delivery (Couriers)"** which accounts for the time taken by the courier to pick up the order and walk to his

vehicle, deliver to the recipient and drive back to the restaurant. Further, we also set the number of couriers for this block using replicate. We need to find the ideal number of couriers, so we simulate over different number of couriers and pick the one for which the KPIs are the best performing. Once an order is delivered and the courier returns to the restaurant, it is recorded as the end and the resource 'courier' is freed up to work on the other orders in the queue. Based on the data, the pick-up time is 5 minutes, and delivery follows a distribution named '**delivery_distribution".** We also need to consider that the courier will drive back to the restaurant. So, we set the timing for this section to be **"5+2*(delivery_distribution)".**

Coming back to the in-restaurant. Once people have been split by probability, they either go to the end **'Takeaway'** block or they queue up to sit in. Right after the queue, the system checks if there are any available tables in the "**Table_Availability"** block. Since this block is a decision block, the time for this block is set to 0 so it is solely used to check conditions. This block has its routing out options set to priority which indicates that the priority is given to eating in and if the resources in eating in are full then people get routed out to **"Takeaway".**

If tables are free in the next block "**Eat In**", customers are routed in. The capacity of this block is set to the number of tables using replicate. Since the number of tables depends on the number of machines we install, we also iterate over this value using the equation 30-2*number of machines. As for the time of this section, it depends on the label 'lbl_eatin_time'**,** which allocates 'first_eatin' distribution to customer coming in from walk in and 'second_eatin' distribution to customers who were routed out from 'eatin'. Further, based on the data, we also find that 75.7% of customers leave after eating in and only 24.3% reorder. We include this by setting the routing out by percentage. The orders that get routed out to reorder go and join the ordering queue again.

The last block is the '**Leave_Restaurant'** and this block has all the people who sit in and eat and directly leave and the people who sit in, reorder and then sit in again before leaving.

This shows the working of the entire model.

d. Calculating Results

We run this above model while iterating over the number of couriers, number of machines and number of tables and for each combination we run the trails with 100, 500 and 1000 replications. To iterate over different values, we simply change the "Replicates" in the "PickUp and Delivery (Courier)" block, "Order (Machines)" block and "EatIn" block respectively.

To record the KPIs,

- **% of food delivered within 1 hour:** We look at the "Delivered" end block results and set the time limit to 60 minutes and record the percentage within the limit.

- **Percentage of order completed (%):** We sum over the orders delivered from the "Delivered block", "Immediate_Takeaway" and "Leave_Restaurant" and divide it by the sum of "Walk In" and "Delivery Orders".
- **Time between arriving and receiving the food:** For this, we cannot directly do this in simul8, so we created an alternative model where we add an extra exit for reorders who are directly taking away after their food is prepared. This ensures that the time from arriving and receiving is not affected by the first sit in time and purely depends on the first orders.

Based on the replications and trials, we form a KPI table and base our analysis and results on it.

Analysis of the simulation

We need to determine how many couriers should be employed and how many machines should be installed. From KPI's, the percentage of orders completed per day is used instead of the expected number of orders completed per day, as percentage value will be easier to understand and make decision. Therefore, the tables below show information about the percentage of food delivered within 1 hour, the percentage of order completed in 1 day, and the time between arriving and receiving the food with 100, 500, and 1000 replications.

Machine	Replications	1	1	1	1
Couriers		4	5	6	7
% of food delivered within 1 hour	100	39.78%	72.71%	93.40%	98.62%
Percentage of order completed (%)		71.00%	71.08%	71.08%	71.08%
Time between arriving and receiving the food		109.48	109.48	109.48	109.48
% of food delivered within 1 hour	500	40.32%	73.43%	93.13%	98.30%
Percentage of order completed (%)		71.07%	71.13%	71.13%	71.13%
Time between arriving and receiving the food		110.07	110.07	110.07	110.07
% of food delivered within 1 hour	1000	40.07%	74.11%	93.21%	98.24%
Percentage of order completed (%)		71.11%	71.18%	71.18%	71.18%
Time between arriving and receiving the food		109.19	109.19	109.19	109.19

This table shows the results using 1 machine and 4 to 7 couriers. For 1 machine, the percentage of orders completed per day is around 71%, but the average time between arriving and receiving the food goes up to 110 minutes which is almost 2 hours of waiting. So, the restaurant will not consider using only 1 machine.

Machines	Replications	2	2	2	2
Couriers		4	5	6	7
% of food delivered within 1 hour	100	39.88%	72.06%	92.96%	98.21%
Percentage of order completed (%)		97.64%	97.71%	97.71%	97.71%
Time between arriving and receiving the food		22.94	22.94	22.94	22.94
% of food delivered within 1 hour	500	40.15%	72.88%	92.62%	97.87%
Percentage of order completed (%)		97.85%	97.91%	97.91%	97.91%
Time between arriving and receiving the food		22.56	22.56	22.56	22.56
% of food delivered within 1 hour	1000	39.92%	73.50%	92.64%	97.81%
Percentage of order completed (%)		97.82%	97.88%	97.88%	97.88%
Time between arriving and receiving the food		22.71	22.71	22.71	22.71

If the restaurant uses 2 machines and employs 4 to 7 couriers, the percentage of orders completed per day is almost 98%, and the average time between arriving and receiving the food is around 23 minutes. For the couriers, like the previous table, when employing 7 couriers, the number of orders that will be delivered within 1 hour is around 98%. However, 6 couriers can deliver 93% of the orders within 1 hour, which is a high value yet also saves some costs for the restaurant. This number of machines looks good, but we will consider using more machines to check the accuracy.

Machines	Replications	3	3	4	4
Couriers		6	7	6	7
% of food delivered within 1 hour	100	89.76%	94.95%	89.26%	94.37%
Percentage of order completed (%)		98.55%	98.55%	98.58%	98.58%
Time between arriving and receiving the food		15.72	15.72	15.32	15.32
% of food delivered within 1 hour	500	89.55%	95.02%	89.44%	94.87%
Percentage of order completed (%)		98.57%	98.57%	98.59%	98.59%
Time between arriving and receiving the food		15.55	15.55	15.22	15.22
% of food delivered within 1 hour	1000	89.68%	95.02%	89.58%	94.96%
Percentage of order completed (%)		98.58%	98.58%	98.59%	98.59%
Time between arriving and receiving the food		15.54	15.54	15.23	15.23

If the restaurant uses a higher number of machines, using 3 or 4 machines, the percentage of completion order within 1 day is around 98%, and the time between arriving and receiving the food is around 15 minutes. We can see that the value stays stable after 3 machines. When hiring 6 couriers, food can be delivered within 1 hour 89% of the time. When hiring 7 couriers, this value increases to 95%

Sensitivity of the Analysis to the Assumptions:

Since, we make certain assumptions to support our model, it does have its implications on the results.

- Since the model follows a 95% confidence interval, the results for the KPIs are accurate 95% of the times that the simulation is run. Meaning that if the simulation is run for 100 runs, then the KPIs will be in the confidence interval range for 95 runs.
- In a real-world scenario, a customer might leave the ordering queue in a restaurant if they see that the queue is long. This possible scenario is not considered as part of the model and thus might affect the number of people in the system which in turn might affect the KPI for % of orders completed.
- We assume that only 1 order can be delivered at a time by 1 courier. However, if we allocate couriers by region i.e. group orders whose delivery address are close to one another/ in the same region and let 1 courier pick up all couriers for that area at the

same time, it will save on the time spent in delivering. In the current model, a great amount of time between ordering and delivering is spent on the courier travelling back. Thus, allowing multiple deliveries at once for the orders of the same region will reduce that and more orders will be delivered within an hour.

- Each table can have a larger sitting capacity to accommodate more people. This ensures that even with lower number of tables the capacity is not very low.

Conclusion

In conclusion, the restaurant should use **2 machines**, as the completion order is around 98%, which is less than 3 machines by only around 1%, which is not a very significant value. Similarly, the difference in the average time until receiving the food for using 2 machines and 3 machines is only around 8 minutes. Thus, the restaurant can consider using 3 machines if they want the average waiting time until customers receive their food to be less, but this will add more cost to it. Moreover, **6 couriers** should be employed because the percentage of orders delivered within an hour is around 93% with using 2 machines (89% with using 3 machines), which is more than using 5 couriers by around 20%. Also, hiring 7 couriers improves the percentage of food getting delivered within an hour by only 5%.

Thus, if the restaurant does not mind bearing the extra cost, then they can employ 7 couriers and add 3 machines for the best KPIs. However, if the restaurant is conscious of the cost, then employing 6 couriers and using 2 machines still gives very high and efficient KPIs.