Business Simulation A3 Report

By Wesley van der Horst and Nando Beijaard

This report aims to elaborate on the optimization of a single server queue with infinite capacity and recommend our best findings. This single server has changing service times depending on two threshold variables k and K. The values of k and K must be determined by us depending on the average cost of the server. The starting (and low) service time of the server is distributed exponentially with a rate of 2. Big K determines the number of customers at which the server's service time switches to be exponentially distributed with a rate of 6. This higher rate is the service time of the server until there are (small) k customers left in the queue; from small k the server returns to work at the lower rate of 2. These service times have certain costs, the lower service time has a cost of 5 euros per time unit whereas the higher unit has a cost of 10 euros per time unit. To optimize the queue of this single server we must find the threshold (k, K) with the lowest average cost.

CRN Implementation

First, we will compare the difference between thresholds (5,20) and (9, 20) using the variance reduction technique called common random numbers (CRN). This technique is used to improve the comparison between scenarios by using the same randomly generated numbers. Using CRN, the arrival time RNG used the same seed for both thresholds. This is also the case for the service time RNG. The following values are found for the average cost of the server of both thresholds:

Average cost for (5,20)	3.7932365...
Average cost for (9,20)	3.7964696...

The values found for the average cost imply that using a threshold that consists of a small k of 5 and a big K of 20 will result in a smaller average cost per time unit than a threshold of (9,20). This is not surprising since the smaller small k implies that a server would stay in the state of a higher service time rate for a longer time. Since the cost of a higher rate only doubles whereas the service times are cut to 1/3 of the original service times. So, a smaller average cost per time unit is expected. Furthermore, since CRN was used to obtain these results we can almost assume that (5, 20) is the better solution, which will be tested in the following chapter.

Impact of CRN

Secondly, the impact of CRN is tested by comparing the average differences in costs between the two previously mentioned thresholds of (5, 20) and (9, 20) over 100 simulations for both normal simulations and CRN simulations. These differences, by deducting the costs of the latter from the former, were added to a TallyStore for every simulation and resulted in the following table:

Method	Num. Obs.	Min	Max	Average	Variance	Std. Dev.
Normal	100	-0.169	0.145	-0.016	4.5E-3	0.067
CRN	100	-4.3E-3	1.0E-3	-1.7E-3	1.1E-6	1.0E-3

As expected, there is a great difference between the results. All values for the CRN are considerably smaller than the normal method. This is caused by the effect of the increased correlation between the simulation results for the CRN, resulting, most importantly, in smaller variance and standard deviation, which would also affect a p-value if one would consider a test for equality of these solutions.

However, both methods do favour thresholds (5, 20) as before, since the averages turn out negative which means that the costs of (9, 20) were on average larger than the costs of (5, 20). These results are thus still in line with our previous assumptions, although the normal method could have also shown different results since they are (pseudo) completely random.

Finding the best solution

In order to find the best thresholds, we must simulate the model multiple times for a whole array of 55 possible thresholds, the thresholds can have the following values:

$$k \in \{5, 6, \dots, 9\}$$
$$K \in \{10, 11, \dots, 20\}$$

Since we only have a budget of 5000 simulation runs, we must spend each simulation carefully, we cannot simulate every threshold thousands of times to find a threshold with a low average cost. To save valuable simulation space we will use two methods of comparing multiple scenarios. These methods are local search and ranking and selection.

Local Search

To keep the limited budget in consideration the first method starts with picking a random threshold: π. This solution is then used to retrieve a random neighbor: π' for comparison. The random neighbor was found by using an RNG to decide how the (k, K) threshold values had to change by either 1, 0 or -1. From a grid perspective, any non-border solution has 8 neighbors, while single- and double-bordered have 5 and 3 respectively. While the new random (k, K) solution was the same as π or out of bounds for the thresholds this process was repeated until an existing random neighbor was found.

In a repeating cycle, the average costs of the solution π and a random neighbor π' were simulated and added to their respective tallies. The value of π was then updated to the overall best performing of the two and then 2 was deducted from the total budget. This process from picking a random neighbor π' of π until deducting from the budget was then repeated until the budget ran out, which resulted in threshold (5, 10), which was simulated 2050 times with an average cost of 3.692, coming out on top.

Ranking and selection

First, every solution is simulated $n = \frac{budget/2}{\#Solutions}$ times, since the total budget was 5000 simulations all 55 solutions were simulated $n = 45$ times. Using these simulations averages and variances are calculated for each solution by updating the Tallies. These averages and variances are used to check whether certain solutions are outperformed by other solutions. Outperformed means that another solution's average + a certain factor is smaller than the average of the solution that is being checked as depicted in the formula below. This factor is composed of a student t distribution, the variances of both the solution being compared and the solution that is being compared to and n.

$$I = \left\{ \pi \middle| \bar{r}_n(\pi) < \bar{r}_n(\pi') + t_{1-a^*}^{n-1} \frac{\sqrt{s_n^2(\pi) + s_n^2(\pi')}}{\sqrt{n}}, \forall\, \pi' \neq \pi \right\}$$

A selection I of 13 solutions was found over which the rest of the budget was evenly spread:

$$I = \{(5,10),(5,11),(5,12),(5,13),(6,10),(6,11),(6,12),(7,10),(7,11),(8,10),(9,10),(9,11),(9,13)\}$$

Ranking and selection found the following threshold: (5, 10) simulated a total of 237 times. This is threshold has an average cost per time unit of 3.694 and is the same threshold found as in the local search.

Overall solution (5, 10) which are the lower bounds for both thresholds seems to be the most cost-effective to use, which coincides with our assumptions that it would be beneficial to have the servers at the higher rate as much as possible.