# Reinforcement Learning to reduce Reneging and Cost of Operation in Queuing Systems

**Deepraj Dutta**
*Dept. Of Science and Humanities*
*National Institute of Technology Nagaland*
Chumukedima, Nagaland, India
deepraj.dutta299@gmail.com

**Dr. Prem Prakash Mishra**
*Dept. Of Science and Humanities*
*National Institute of Technology Nagaland*
Chumukedima, Nagaland, India
maths.prem79@gmail.com

***Abstract*—Queuing systems have been studied using various tools of optimization where we try to balance two contrary costs, the waiting cost and the cost of service. The decrease in service rate causes loss of customers through reneging. On the other hand, the increase in service rate increases the cost of operation. In this paper, we develop a total cost function that takes into account the cost incurred from reneging and the cost incurred due to maintaining the server operation. We then employ this as a reward function for our Q-learning model. The Q-learning model in this paper takes integer service rates as states and the increase or decrease of service rates by some step size as action. This algorithm gives us the optimal service rate. Furthermore, we use a numerical problem to demonstrate the same. Through this paper we show the reader how machine learning can be used to solve optimization problems in queuing models.**

***Keywords—Queuing systems, Reinforcement Learning, Q-Learning, reneging, cost of operation.***

## I. INTRODUCTION

Researchers have tried to understand the phenomenon of reneging and how to optimize their service rates to control reneging. To this effect, they have used many tools, and many studies have been performed. Kumar & Sharma study a finite waiting space single server Markovian queuing model where they develop an expression for the rate of abandonment and also develop a model for the retention of reneging customers [1]. Sharma et al. use an adaptive queuing model to control reneging [2]. Wang and Zhang studied a multi-server queuing system with a finite calling population where the strategic behaviour of the server is used in order to control balking and reneging, but they don't involve the use of any machine learning algorithm [3]. Queuing models, however, have been studied using machine learning before. Kyritsis and Deriaz used machine learning to predict waiting times [4]. Yao et al. use machine learning to propose routing schemes for load balancing [5]. Garbi et al. use a compact system of ordinary differential equations; they further encode these equations into a recurrent neural network [6]. Zhang et al. perform a forecasting of EV load using deep learning [7]. Benevento et al. design a new queue-based variable that captures the state the emergency departments are in in order to improve the prediction accuracy of waiting time [8]. Petrovic et al. used some recurrent neural network architectures for predicting the expected intensity of arrivals [9]. Thus, we see that machine learning has been effectively used to study queuing systems and queuing networks. In this work, Q-learning, a type of Reinforcement Learning, is used to deal with the stochastic nature

of the number of reneging customers, and a model is proposed that will help the server decide the optimal rate of service. We set up a Q-learning model by discretizing the service rates, these service rates are taken as states that the Agent (server) can be in. The actions that the agent can take in each state are to either increase the service rate or to decrease the service rate, and that will incur him some reward, which we take as the negative total cost function. This way the agent can decide at each state, from the Q-table, whether he should increase the service rate or decrease the service rate. Hence, from the Q-table, we can decide which state (service rate) is optimal. The paper further goes on to give an example to illustrate the operation and effectiveness of the model.

## II. MODEL FORMULTION

### *A. Notations and Symbols*

$\mu =$ service rate
$r_\mu =$ number of reneging customers when the service rate is $\mu$
$r_\sigma =$ standard deviation of the reneging customers.
$c_\mu =$ the cost of operation when the service rate is $\mu$
$p =$ the profit from each customer
$\omega_1 =$ the weight assigned to the total cost due to reneging
$\omega_2 =$ the weight assigned to the total cost due to cost of operation
$TC_\mu =$ total cost when the service rate is $\mu$
$\alpha =$ the learning rate
$\gamma =$ the future discount rate
$\epsilon =$ the exploration rate
$ep =$ the number of episodes
$S =$ the state space
$S' =$ the terminal state space
$A =$ the action space
$R =$ reward
$Q(s, a) =$ the Q-value corresponding to state $s \in S \; and \; a \in A$
$M_s =$ the maximum number of steps in each episode

### *B. Assumptions*

In this model,

- We assume a general single server queue with any arrival and service process.
- We assume that the reneging and cost of operation are both dependent on the service rate.
- We assume that the number of reneging customers in each state follows a normal distribution.

### *C. Model Description*

We first introduce the total cost function that we wish to minimize, we take a weighted sum of the costs that the system has to incur:

1. the profit loss from reneging that reduces as the service rate increases and

2. the cost of operation which increases as the service rate increases.

We write this as:

$$TC_\mu = \omega_1 . p . r_\mu + \omega_2 . c_\mu \quad (1)$$

We wish to find the optimal $\mu$ that minimizes this function. Due to the stochastic nature of $r_\mu$ the total cost $TC_\mu$ also becomes stochastic. To properly capture this phenomenon and optimize the function, we take the help of Q-learning.

Q-learning is an off-policy Reinforcement Learning method that uses data over long periods to give us an optimal policy. In this case, we will use it to identify what the optimal service rate is for the server that reduces his total cost.

We start by defining our state space S (the discretized service rates) and take the smallest and largest service rates to be the terminal state space $S'$, the action space A (increasing and decreasing the service rate), and our reward (the negative total cost function). We then initialize a $|S|\text{x}|A|$ matrix 'Q', called the Q-table, with all zero entries.

The agent then starts to interact with the environment over a number of episodes, the path that an agent takes in each episode is called a trajectory. The action '$a$'that the agent takes in state '$\mu$' is based on an epsilon-greedy policy where, in state '$\mu$', the agent explores i.e. he chooses a random action according to the exploration rate '$\epsilon$'; otherwise he exploits i.e. he goes with the action that gives him the greatest reward. The action that gives the greatest reward is the action corresponding to the highest Q-value corresponding to that state. After taking the action, the agent transitions to a new state '$\mu'$', where he gets a reward $R = -TC_\mu$.

The Q-value for the state action pair $(\mu, a)$ i.e. $Q(\mu, a)$ is updated using relation (2).

$$Q(\mu, a) \leftarrow Q(\mu, a) + \alpha\{R + \gamma \max_{a'} Q(\mu', a') - Q(\mu, a)\} \quad (2)$$

The Q-learning flowchart for this particular model is given in Figure 1.

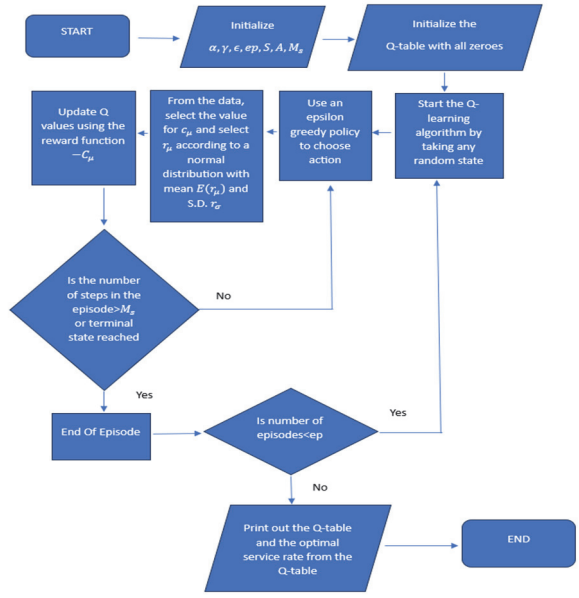


*Figure 1. Q-Learning Flowchart for the proposed model*

Using this we can find the optimal service rate $\mu$ that minimizes our total cost.

## III. ILLUSTRATIVE EXAMPLE

Let us say, the maximum service rate($\mu$) of the queuing system is 20. Also, we take the following hypothetical data:

| Service rate $\mu$ | Expected number of reneging customers $E(r_\mu)$ | Cost of Operation $c_\mu$ | Expected Total Cost $E(TC_\mu)$ |
|---|---|---|---|
| 0 | 42.294849 | 0.000000 | 317.2113675 |
| 1 | 39.668156 | 10.000000 | 302.51117 |
| 2 | 37.075809 | 28.284271 | 292.210703 |
| 3 | 34.519234 | 51.961525 | 284.8750175 |
| 4 | 32.000000 | 80.000000 | 280 |
| 5 | 29.519846 | 111.803398 | 277.300544 |
| 6 | 27.080709 | 146.969391 | 276.590013 |
| 7 | 24.684776 | 185.202591 | 277.7371155 |
| 8 | 22.334517 | 226.274170 | 280.6459625 |
| 9 | 20.032763 | 270.000000 | 285.2457225 |
| 10 | 17.782795 | 316.227753 | 291.484839 |
| 11 | 15.588457 | 364.828735 | 299.327795 |

ISBN : 978-81-978522-2-0

| | | | |
|---|---|---|---|
| 12 | 13.454343 | 415.692200 | 308.7536725 |
| 13 | 11.386036 | 468.721680 | 319.75611 |
| 14 | 9.390508 | 523.832031 | 332.3448255 |
| 15 | 7.476744 | 580.947510 | 346.549335 |
| 16 | 5.656854 | 640.000000 | 362.426405 |
| 17 | 3.948222 | 700.927979 | 380.0756545 |
| 18 | 2.378414 | 763.675354 | 399.675782 |
| 19 | 1 | 828.190796 | 421.595398 |
| 20 | 0 | 894.427185 | 447.2135925 |

***Table 1. The data for E ($r_\mu$), $c_\mu$ and E ($TC_\mu$) corresponding to μ.***

Here, we take $\alpha = 0.01, \gamma = 0.9, \epsilon = 0.1, S = \{1, \dots, 19\}, S' = \{0,20\}, A = \{-1, +1\}, p = 15,$ $ep = 2000, M_s = 500, \omega_1 = 0.5, \omega_2 = 0.5, r_\sigma = 2.$

***A.*** *Q-tables*

The Q table shows what is the action an agent should take given that he is in a certain state.

The states in this scenario are the service rates, ranging from 1 to 19 and the actions are increasing or decreasing it by a step size of 1.

We use the Q- Learning algorithm to obtain the following Q-tables at different number of episodes:

| | |
|---|---|
| -2778.6349893979504 | -2778.457255180012 |
| -2776.133442478905 | -2773.5918919503133 |
| -2779.927621644395 | -2770.6672638176124 |
| -2774.380233192465 | -2767.9801067887493 |
| -2770.0241836718383 | -2773.332147701332 |
| -2772.936419131013 | -2773.8117749885637 |
| -2772.387325654487 | -2781.4060575811295 |
| -2774.4984504330373 | -2780.1208426217318 |
| -2782.2992414201735 | -2787.500187961884 |
| -2795.763184292583 | -2799.518791497188 |
| -2813.296704282304 | -2814.354801091023 |
| -2837.5027834163207 | -2838.3482422127026 |
| -2862.1082301830747 | -2863.2062873232003 |
| -2894.339660368871 | -2895.672184349154 |
| -2926.2387290450756 | -2926.5327181416055 |
| -2957.735118116294 | -2958.4419345563683 |
| -2977.4930706298937 | -2978.301670472658 |
| -2992.2565792484193 | -2992.596104311169 |
| -2962.025224735303 | -2960.884519289209 |

*Table 2a*

| | |
|---|---|
| -2783.1959543248654 | -2782.128507101723 |
| -2780.6191549452205 | -2774.5552588290184 |
| -2783.419685198795 | -2769.880250735418 |
| -2776.636313741894 | -2769.3907931979656 |
| -2770.8850186651202 | -2771.4970596900994 |
| -2769.8430857065528 | -2777.2999647333454 |
| -2771.7166221580887 | -2782.6723426528843 |
| -2775.6796001514767 | -2787.3804592243737 |
| -2783.2773012170196 | -2793.619819567722 |
| -2797.401743745973 | -2806.6564668443984 |
| -2818.1939735927444 | -2829.401728761093 |
| -2846.5508953756394 | -2853.6537204371602 |
| -2881.8912992691817 | -2888.1191673353505 |
| -2925.5209923750326 | -2929.666928673992 |
| -2975.394142168671 | -2978.297457011993 |
| -3020.4507294437485 | -3020.6356102783584 |
| -3062.1490102112775 | -3061.711745893855 |
| -3086.6614565533323 | -3085.958492868214 |
| -3059.327876521262 | -3058.718680727118 |

*Table 3b*

***Table 4. Table 2a consists of the Q-table for state-action pair at 1000 episodes and at 2000***

*episodes in Table 2b*

The right column is the action of increasing $\mu$ by 1 and the left is the action of decreasing $\mu$ by 1. And the rows are the states($\mu$) from 1 till 19.

***B.*** *Efficiency of the Algorithm*

The graph below tells us how the algorithm gets us closer to the optimal value as the number of episodes increase:

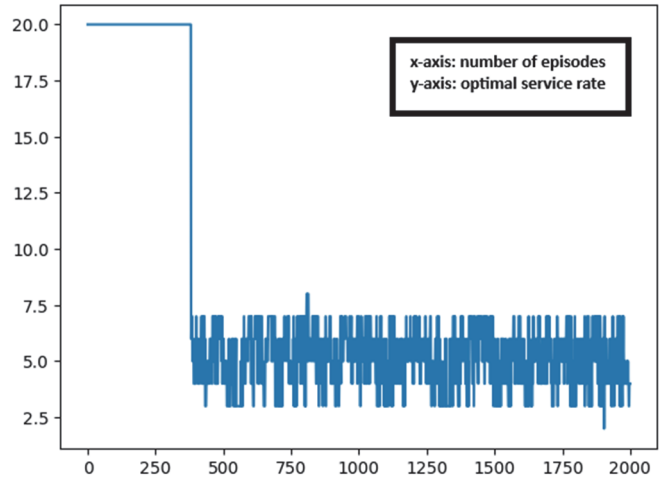


***Figure 2. The optimal service rate against the number of episodes***

After 2000 episodes the algorithm give us the optimal service rate as 4. From Table 1. One can also find out the expected minimum total cost which is 276.59 at service rate 6.

## CONCLUSION

In this paper we solved an optimization problem and we found the optimal service rate that the server should maintain if he wishes to reduce his total cost. In this paper we have seen that Q-learning can help us map the problem and efficiently solve it; however, from Figure (2) we can see that it doesn't always give us an accurate solution but only gives us a close estimate. From Table 1. we also saw that the expected optimal service rate is 6. However, the expected value doesn't take into account the full diversity of the reneging phenomenon which the machine learning algorithm does. The Q-learning approach used in this paper can have many industry applications and will especially be useful in situations where obtaining analytical solutions is challenging. Reducing the learning rate and increasing the number of episodes would give us a more accurate result; however, that might require quite a bit of computing power. The requirement of a lot of data to get a more accurate result is a limitation of this method and might be impractical for situations where the data set is too high.

## REFERENCES

[1] Kumar, Rakesh. "A single-server Markovian queuing system with discouraged arrivals and

retention of reneged customers." *Yugoslav journal of operations research* 24, no. 1 (2016).

[2] Sharma, Sapana, Rakesh Kumar, Godlove Suila Kuaban, Bhavneet Singh Soodan, and Pradeep Singh. "Performance and cost evaluation of an adaptive queuing system with customer reneging and retention: steady-state and transient analysis." *International Journal of Services, Economics and Management* 15, no. 3 (2024): 254-272.I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Wang, Qiangqiang, and Bin Zhang. "Analysis of a busy period queuing system with balking, reneging and motivating." *Applied Mathematical Modelling* 64 (2018): 480-488.

[4] Kyritsis, Athanasios I., and Michel Deriaz. "A machine learning approach to waiting time prediction in queueing scenarios." In *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*, pp. 17-21. IEEE, 2019.D. P. Kingma and M. Welling, “ Auto-encoding variational Bayes,” 2013, arXiv:1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114.

[5] Yao, Haipeng, Xin Yuan, Peiying Zhang, Jingjing Wang, Chunxiao Jiang, and Mohsen Guizani. "Machine learning aided load balance routing scheme considering queue utilization." *IEEE Transactions on Vehicular Technology* 68, no. 8 (2019): 7987-7999.

[6] Garbi, Giulio, Emilio Incerto, and Mirco Tribastone. "Learning queuing networks by recurrent neural networks." In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pp. 56-66. 2020.

[7] Zhang, Xian, Ka Wing Chan, Hairong Li, Huaizhi Wang, Jing Qiu, and Guibin Wang. "Deep-learning-based probabilistic forecasting of electric vehicle charging load with a novel queuing model." *IEEE transactions on cybernetics* 51, no. 6 (2020): 3157-3170.

[8] Benevento, Elisabetta, Davide Aloini, and Nunzia Squicciarini. "Towards a real-time prediction of waiting times in emergency departments: A comparative analysis of machine learning techniques." *International Journal of Forecasting* 39, no. 1 (2023): 192-208.

[9] Petrović, Andrija, Mladen Nikolić, Uglješa Bugarić, Boris Delibašić, and Pietro Lio. "Controlling highway toll stations using deep learning, queuing theory, and differential evolution." *Engineering Applications of Artificial Intelligence* 119 (2023): 105683.