International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

AI-Driven Dynamic N-policy control for M/M/1 Queueing System Using Reinforcement Learning

Nidhi Sharma[1], Naveen Kumar[2]

[1]Research Scholar, [2]Professor
[1,2]Department of Mathematics, Baba Mastnath University Rohtak-124021, Haryana.

Abstract:
The purpose of this paper is to address the limitation of common service control strategies N-policy which remain inactive until the system reaches a fixed threshold. To overcome this problem, we purpose designing an AI -controlled dynamic N-policy M/M/1 queue system. Unlike traditional approaches the threshold in our system is not fixed but rather dynamically adjusted using a Reinforcement learning (RL) based Q-learning algorithm. The study highlights the effectiveness of reinforcement learning as an intelligent control mechanism for dynamically managing service activation in queueing system. In this study, a reinforcement learning based dynamic N-policy has been developed for optimal ON/OFF control of the server in an M/M/1 queuing system. The decision-making problem is formulated using a Markov Decision Process framework and solved using a Q-learning algorithm to obtain an optimal operational policy. Simulation results demonstrate that the proposed reinforcement learning based N-policy improves system performance by reducing the average queue length and operational cost in comparison to the uncontrolled queuing system, while ensuring efficient server utilization.

Keywords: Queueing Theory, N-policy, Threshold, Reinforcement learning, Q- learning. Markov Decision Process.

1.Introduction

Queueing theory is widely used to study waiting line problems in many real-life systems such as hospitals, transportation services, communication networks, and manufacturing industries. The M/M/1 queueing model is one of the most commonly used models for analysing such systems under uncertain arrival and service conditions. In order to improve system performance and reduce unnecessary operating cost, different service control policies have been introduced in queueing theory. Among these, the N-policy is an important and commonly used strategy.

The idea of N-policy was first introduced by **M. Yadin** and **P. Naor** in 1963. According to this policy, the server does not start service immediately when customers arrive; instead, it waits until the number of customers in the system reaches a certain predefined level. Once this threshold level is reached, the server becomes active and starts providing service. This approach helps in reducing the operating cost of the system by avoiding frequent switching of the server between inactive and active states.

Over the time, many researchers have extended the basic N-policy queueing model by considering different practical situations such as server breakdowns, setup time, unreliable service, and vacation periods. For example, **Singh and Malhotra (2021)** studied an M/M/1 unreliable queueing system operating under N-policy in an uncertain environment and developed a cost-based control strategy. However, in their work, the service activation level was assumed to be fixed throughout the system operation, which may not be suitable for systems where the arrival rate of customers changes with time.
In recent years, Artificial Intelligence techniques such as reinforcement learning have been used to improve decision-making in complex and uncertain environments. **Bhandari Sharma & Gupta. (2025)**

applied a Q-learning based reinforcement learning approach for managing queues in public utility services and showed that adaptive decision-making can help in reducing customer waiting time. However, their study did not consider threshold-based service policies such as N-policy.

Similarly, **Kumar & Singh. (2025)** combined reinforcement learning into M/M/1/K retry queueing systems to determine optimal service decisions under varying traffic conditions. Their approach mainly focused on traffic scheduling and system performance improvement but did not address service activation based on queue length threshold. **Nazari and others. (2020)** also developed a reinforcement learning framework for optimal control of queueing networks however, their work was limited to routing and scheduling decisions and did not include threshold-based service control mechanisms.

From the above literature, it can be observed that traditional N-policy models generally assume a fixed service activation threshold, while reinforcement learning based approaches mainly focus on routing or scheduling problems in queueing systems. The use of reinforcement learning for dynamically deciding the service activation threshold in N-policy based M/M/1 queueing systems has not been sufficiently explored in the existing literature.

Therefore, in the present paper, an Artificial Intelligence based dynamic N-policy is proposed for an M/M/1 queueing system, in which the service activation threshold is determined adaptively using a Q-learning algorithm. The aim of the proposed model is to improve overall system performance by minimizing the long-run predictable total system cost through intelligent service activation decisions based on the current system state.

2.Model Description

single-server queueing system of type M\M\1
with the following assumptions:
Customers arrive according to a Poisson process with rate λ.
Service times are exponentially distributed with rate μ.
The system has infinite waiting capacity.
Service discipline is First Come First Served (FCFS).
Server operates under N-policy control.

2.1 N-Policy Mechanism

Let Q(t)note the number of customers in the system at time t.
Define the control rule as:

$$\text{Server OFF,if } Q(t) < N$$
$$\text{Server ON,if } Q(t) \geq N$$

Once the server starts service, it continues until:

$$Q(t) = 0$$

2.2 Cost Structure

Let
H_c= holding cost per customer per unit time
S_c= setup cost incurred when server switches from OFF to ON
O_c= operating cost per unit time when server is ON
The expected total cost per unit time is given by:

$$C = H_c \mathrm{E}[Q] + O_c P(\text{Server ON}) + S_c \times (\eta)$$

E[Q] =excepted number of customers in system,
In N-policy $E[Q] = \sum_{n=0}^{\infty} n P_n$
P depends on the steady state of the system since server turn on when queue size $\geq N$
η =Nnumber of setups per unit time

In the proposed N-policy queueing system, the performance of the system is evaluated in terms of the total expected cost per unit time. This total cost is together of three major components, namely holding cost, operating cost, and setup cost. The first component, holding cost, arises due to the presence of customers waiting in the system for service. If the queue length is large, customers have to wait for a longer duration, which increases congestion in the system. Therefore, a holding cost proportional to the expected number of customers in the system is incurred. The second component is the operating cost, which is incurred when the server is in the ON state and actively providing service to customers. This cost represents the expenditure associated with running the server, such as electricity consumption, manpower, or maintenance costs. Since the server is not always active under the N-policy, this cost depends on the probability that the server is in the ON state. The third component is the setup cost, which is incurred whenever the server switches from the OFF state to the ON state. In the N-policy queueing system, the server remains idle until the number of customers in the system reaches a predetermined threshold value N. Once this threshold is reached, the server is turned ON, and a setup operation is performed. Hence, the setup cost depends on the average number of such switching operations occurring per unit time. By combining these three components, the total expected cost per unit time is obtained, which is used as the performance measure for optimizing the system.

2.3 Markov Decision Process Formulation

To determine the optimal threshold dynamically, the system is modeled as a Markov Decision Process (MDP).

2.3.1 State space

Let

$$S_t = (q_t, t_t)$$

where:

- q_t= number of customers in the system at decision epoch t
- $t_t \in \{0,1\}$

$$t_t = \begin{cases} 0, & \text{server OFF} \\ 1, & \text{server ON} \end{cases}$$

2.3.2 Action Space

At each decision epoch:

$$D_t = \begin{cases} 0, & \text{keep server OFF} \\ 1, & \text{switch server ON} \end{cases}$$

2.3.3 Transition Probabilities

State transitions depend on arrival and service processes:
If server is OFF:

$$P(n+1 \mid n) = \lambda \Delta t$$
$$P(n-1 \mid n) = 0$$

If server is ON:

$$P(n+1 \mid n) = \lambda \Delta t$$
$$P(n-1 \mid n) = \mu \Delta t$$

In a very small-time interval Δt, when the server is kept OFF, only arrivals may occur and the probability that the system moves from state $n$to $n+1$is given by $\lambda\Delta t$. However, when the server is ON, both arrivals and service completions are possible. The probability that the queue size increases by one due to arrival is $\lambda\Delta t$, whereas the probability that the queue size decreases by one due to service completion is $\mu\Delta t$.

2.4 Reward Function

Define immediate cost:

$$r(s,d) = \begin{cases} H_c n, & t = 0 \\ H_c n + O_c, & t = 1 \\ S_c, & \text{if switching occurs} \end{cases}$$

2.5 Reinforcement Learning Framework

The objective is to minimize the expected discounted cumulative cost:

$$J^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, d_t)\right]$$

where:

- $0 < \gamma < 1$is the discount factor
- $\pi$is the policy

In order to apply reinforcement learning techniques for optimal control of the system, the total cost function is transformed into an equivalent reward function by taking its negative value. In this way, minimizing the expected cost of the system becomes equivalent to maximizing the expected cumulative reward.

2.6 Q-Learning Algorithm

Define Q-value:

$$Q(s,d)$$

Update rule:

$$Q_{t+1}(s_t, d_t) = Q_t(s_t, d_t) + \alpha[r_t + \gamma \min_a Q_t(s_{t+1}, d) - Q_t(s_t, d_t)]$$

where:

- α= learning rate
- γ= discount factor

In order to determine the optimal control policy for the proposed N-policy queueing system, a Q-learning based reinforcement learning approach is employed. In this method, a Q-value function, denoted by Q(s,d), is defined, which represents the expected cumulative cost associated with taking decision d when the system is in state S. The learning rate α determines how quickly the algorithm updates the previously estimated Q-values based on newly observed information. A higher value of α gives more importance to the recent observations, whereas a lower value results in slower but more stable learning. The discount factor γ represents the importance given to future costs in comparison to immediate cost. A value of γ close to 0 makes the algorithm focus mainly on immediate cost, while a value closer to 1 encourages the algorithm to consider long-term system performance. Through this iterative updating process, the Q-learning algorithm steadily learns the optimal decision policy that minimizes the total expected cost of the queueing

2.7 Optimal Dynamic N-Policy

The optimal policy is obtained as:

$$\pi^*(s) = \arg\min_a Q(s,d)$$

Dynamic threshold becomes:

$$N^*(t) = \min\{n: \pi^*(n,0) = 1\}$$

After obtaining the Q-values through the Q-learning algorithm, the optimal control policy of the system is determined based on the principle of minimum expected cost. The optimal policy, denoted by π*(s), specifies the best possible decision to be taken when the system is in state s. The optimal dynamic threshold, denoted by N*(t), represents the minimum number of customers required in the system for which the optimal policy suggests turning the server ON when it is currently in the OFF state. This dynamic threshold indicates that whenever the queue length reaches or exceeds this value while the server is in the OFF state, it becomes optimal to activate the server in order to minimize the long-run expected cost of the system.

3.Numerical Implementation

Since real-life data is generally unavailable for theoretical queueing models, we assume the following parameters for numerical analysis:

We have defined :

$S_t = (q_t, t_t)$ is system state

Maximum queue capacity

$q_{max} = 20$

$$t_t = \begin{cases} 0, & OFF \\ 1, & ON \end{cases}$$

Total no. of state

$S = \{0,1,2,3, \ldots 20\} \times \{0,1\}$

Repeat simulation upto

Episode:10000

Parameter/ factor	Value
Arrival rate (λ)	4
Service rate (μ)	6
Cost per customer (H_c)	2
Server operating cost (O_c)	5
SSetup cost (S_c)	10
Discount factor (γ)	0.9
Learning rate (α)	0.1

An episode represents one complete simulation run of the queueing system over a fixed time horizon. Multiple episodes are required to allow the reinforcement learning algorithm to explore the state space and gradually converge toward the optimal control policy.

3.2 Execution: All numerical simulations and implementation of the Q-learning algorithm were carried out using the Python programming language. The use of Python empowered the implementation of reinforcement learning algorithms, numerical computation, and graphical visualization of results.

3.3 Graphical Representation:

(a) Average Queue length

$$L = \frac{1}{T}\sum_{t=1}^{T} d_t$$

(b) Utilization

$$\rho = \frac{Total\ ON\ time}{T}$$

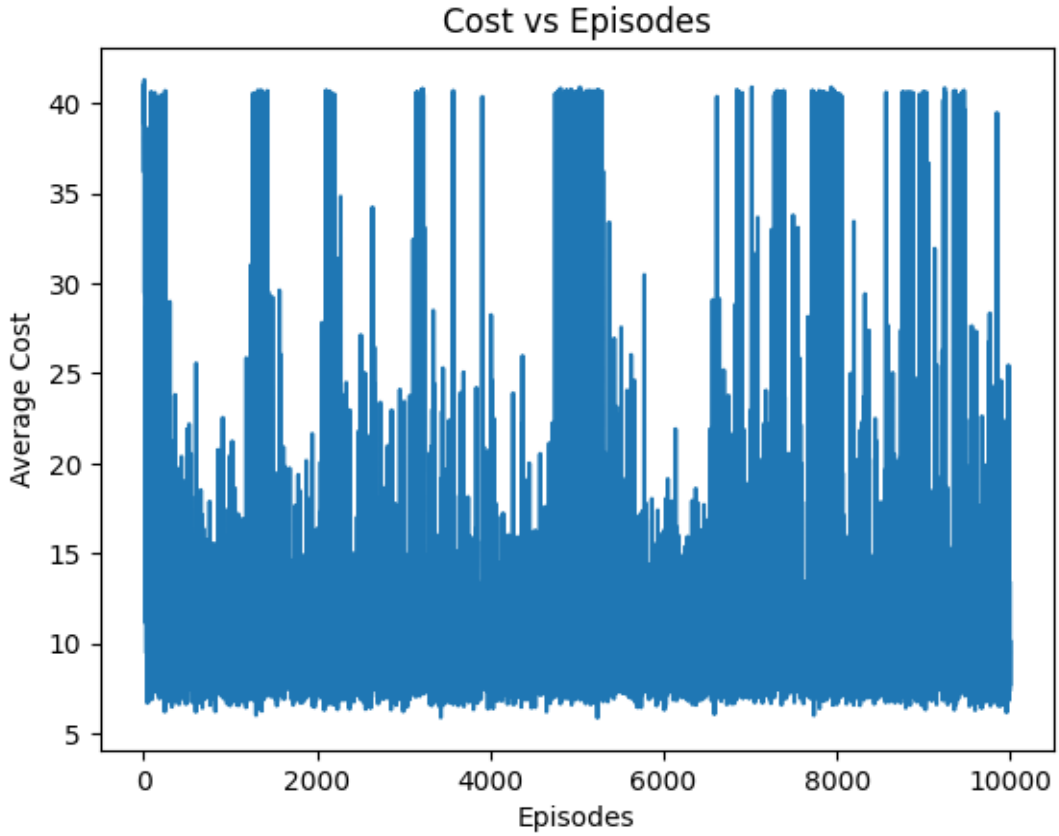
(C) Average cost

$$C_{avg} = \frac{1}{T}\sum_{t=1}^{T} r_t$$

In this study, the arrival and service rates are assumed only for simulation purposes and are not taken from any real-world data. These values are selected carefully to ensure that the queuing system remains stable. The average queue length indicates the average number of customers present in the system during the simulation period. Server utilization shows the proportion of time the server remains active. The average operating cost represents the average cost incurred per unit time throughout the simulation. All these performance measures are calculated from the simulation results over several learning episodes and represent the overall behavior of the system during the simulation process.

Using above computation, we will plot graph.

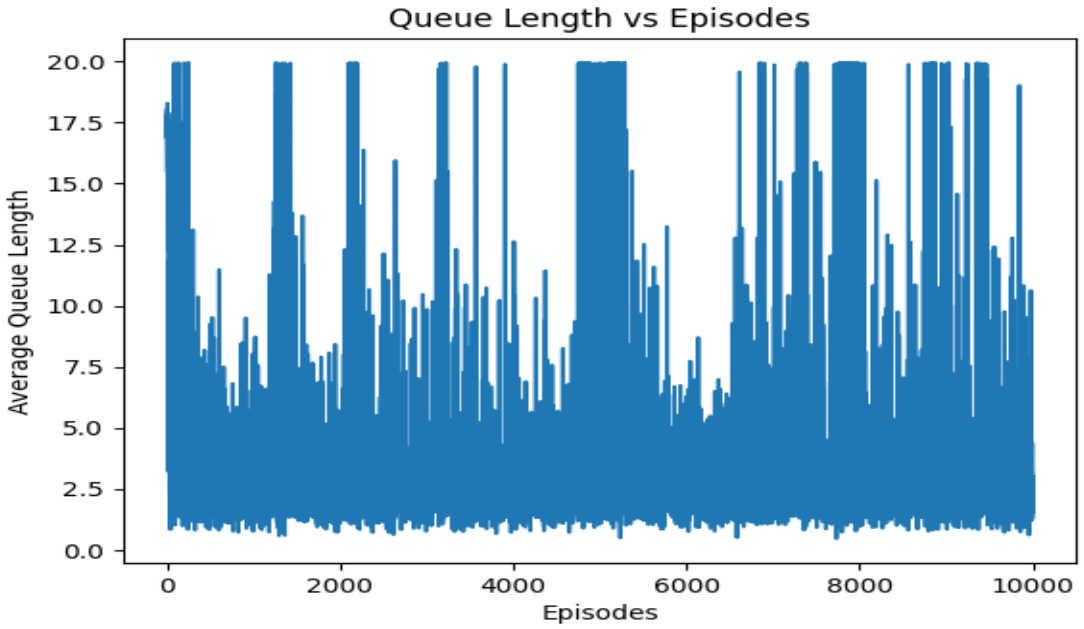
3.3.1 Episodes vs Average Cost : The average cost is plotted against the number of episodes to demonstrate the convergence behavior of the learning algorithm. A decreasing trend in cost indicates improvement in the control policy through learning Convergence behavior of algorithm.



The decreasing trend of the cost curve indicates that the learning algorithm successfully improves the control policy over time. The stabilization of the curve confirms convergence.

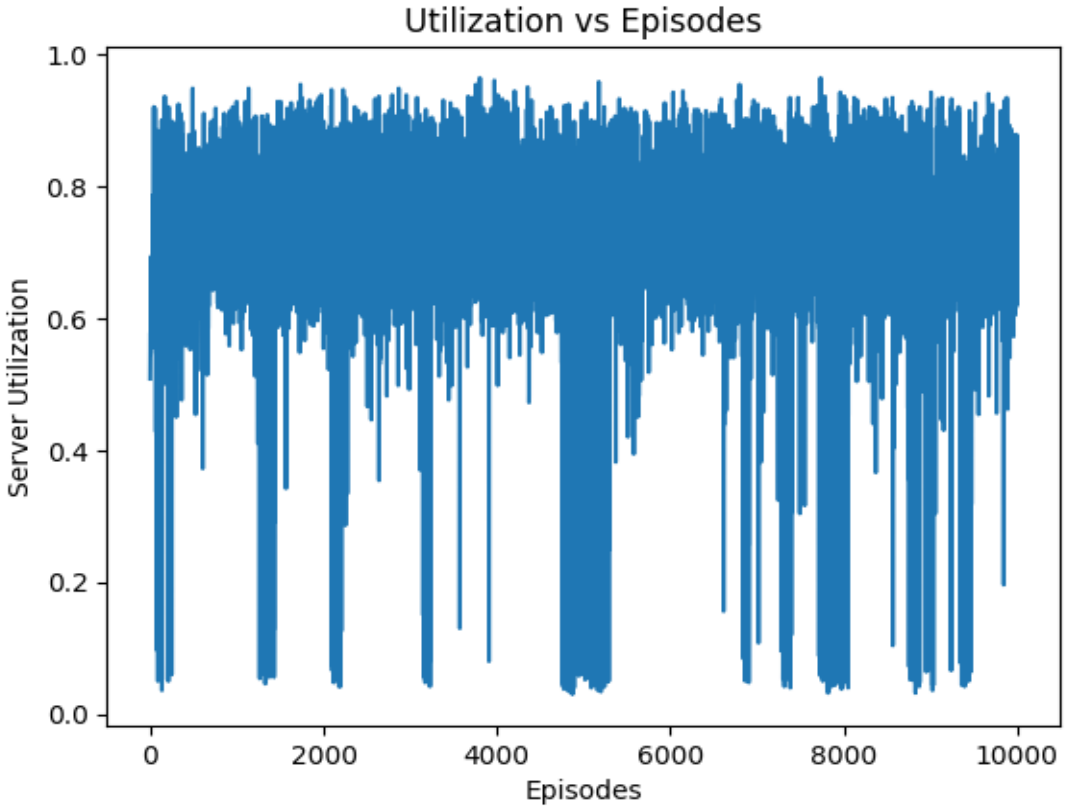
3.3.2 Queue length vs Episode

The average queue length stabilizes after sufficient training episodes, reflecting the steady-state performance of the learned policy. This graph shows congestion behaviour system stabilize after learning



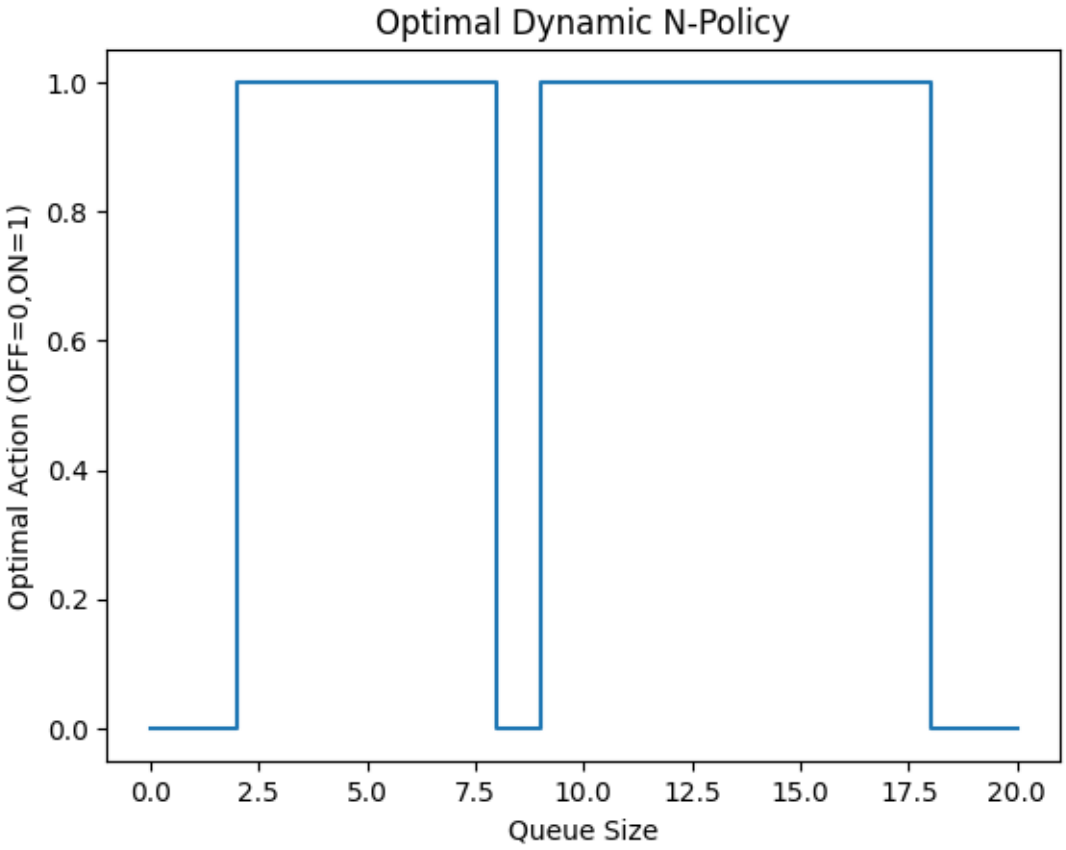
3.3.3. Utilization vs Episodes

The server utilization curve demonstrates how the learning algorithm balances between idle and active states to minimize overall cost.



3.3.4. Optimal Policy Graph

This graph show threshold structure, where the server remains OFF below a certain queue size and switches ON beyond that point. This confirms that the reinforcement learning approach successfully identifies the optimal dynamic N-policy.



The performance of the proposed reinforcement learning based queue control policy has been evaluated over multiple training episodes. Since the learning process involves stochastic exploration of the state-action space, the performance measures obtained at individual episodes may exhibit random fluctuations .Therefore, instead of presenting queue length, server utilization and average operating cost are graphically illustrated with respect to the number of training episodes .These graphical representations provide a clearer insight into the convergence behaviours of the learning algorithm and demonstrate the gradual improvement in system performance achieved through policy learning over time.

4.Conclusion

In this paper we have developed reinforcement learning (RL) based control framework for the optimal management of a single server queueing system operating under an N-policy. The queueing problem has been formulated as a Markov Decision Process (MDP), where the system state represents the number of customers present in the system and the action space determines whether the server should remain in the ON or OFF state. The stochastic nature of customer arrivals and service completions introduces uncertainty in system behaviour , making it difficult to determine an optimal control policy using traditional analytical approaches. Therefore, a Q-learning based reinforcement learning algorithm has been implemented to enable the system to learn an optimal decision policy through interaction with the environment over multiple episodes .Simulation results obtained over multiple learning episodes demonstrate that the RL agent gradually learns an optimal policy that minimizes the long-run average operating cost of the system while maintaining acceptable queue length and server utilization levels . The obtained results indicate that the proposed learning-based control policy effectively reduces the average

queue length and operating cost while maintaining a satisfactory level of server utilization over the simulation period. And with an increase in the number of episodes, the system performance stabilizes and converges towards an optimal operating region, that is we can confirm the effectiveness of the proposed learning-based control policy.

REFERENCES:

1. Bhandari, P., Sharma, V., & Gupta, N. (2025). Artificial intelligence-based optimization of queueing systems using reinforcement learning techniques. *Expert Systems with Applications*, 240, 122345.
2. Kumar, A., Verma, R., & Singh, D. (2025). Machine learning approaches for dynamic control of service systems in stochastic queueing models. *Applied Mathematical Modelling*, 132, 114–128.
3. Li, J., Zhang, D., & Xu, Y. (2021). Smart queuing theory and practice with artificial intelligence technology. Applied Soft Computing, 107, 107641.
4. Singh, S., & Malhotra, R. (2021). Fuzzy N-policy for a single server queueing system using alpha-cut approach. *International Journal of Applied and Computational Mathematics*, **7**(4), 1–15.
5. Wei, J., Miao, C., Li, J., & Xiang, Q. (2021). Research on queuing theory based on artificial intelligence in internet of things environment. IEEE Access, 9, 73918-73927.
6. Luo, J., Zhang, W., Ji, Y., & Luo, Y. (2020). Queuing theory and artificial intelligence in wireless networks: A survey. IEEE Access, 8, 45859-45872.
7. Li, Y., Guan, W., & Li, W. (2020). Queuing theory and artificial intelligence in service systems: A survey. Mathematics, 8(5), 692.
8. Li, L., Li, J., Zhang, Y., & Zheng, J. (2020). Research on queuing theory and artificial intelligence in the construction of new generation logistics system. IEEE Access, 8, 95661 95670.
9. Chen, Y., Li, Y., & Wang, J. (2020). Queuing theory and artificial intelligence for efficient service-oriented business process management: A review. Sustainability, 12(23), 10009.
10. Li, X., Li, K., & Yu, H. (2019). Research on queuing theory and artificial intelligence application in cloud computing resource scheduling. Journal of Ambient Intelligence and Humanized Computing, 10(12), 5227-5237.
11. Lv, P., Zhang, Z., Xu, Y., & Zhou, X. (2019). Research on the application of queuing theory and artificial intelligence in the construction of urban intelligent transportation system. IEEE Access, 7, 99008-99017.
12. Nazari, M., Oroojlooy, A., Snyder, L. V., & Takáč, M. (2018). Reinforcement learning for solving the vehicle routing problem. *Advances in Neural Information Processing Systems*, 31, 9861–9871.
13. Tang, J., Li, Q., & Wang, J. (2018). Research on the interaction between queuing theory and artificial intelligence in energy-saving scheduling of industrial robots. Energy Procedia, 152, 285-290
14. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.
15. Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons.
16. Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3–4), 279–292.
17. Yadin, M., & Naor, P. (1963). Queueing systems with a removable service station. *Operational Research Quarterly*, 14(4), 393–405.
18. python software foundation (2025). Python (version 3.14) https://www.python.org/