Volume: 13 Issue: III Month of publication: March 2025

DOI: https://doi.org/10.22214/ijraset.2025.67300

Intelligent Resource Allocation for NB-IoT Networks Using TD3 Reinforcement Learning

S. Anbazhagan[1], R. K. Mugelan[2], S. K. Thinagaran[3], Krishan Dara[4]
[1, 2, 3, 4]*Department of Communication Engineering, School of Electronics Engineering, Vellore Institute of Technology, Katpadi, Vellore 632014, Tamil Nadu, India*

Abstract: Compared to previous IoT principles, NB-IoT provides end customers with a higher quality of service (QoS) in an era where everything is connected to the internet. The Third-Generation Partnership Project (3GPP) for Low-Power Wide-Area Networks (LPWAN) introduced the narrowband Internet of Things (NB-IoT), a new cellular radio access technology based on Long-Term Evolution (LTE). The main objectives of NB-IoT are to enable low-power, low-cost, and low-data-rate communication and to support massive machine-type communication (mMTC). One of the more difficult tasks in uplink transmission is resource allocation. While numerous suggestions have been made for efficient resource distribution, a comprehensive and successful solution has not yet been provided. In this article, we attempt to suggest a resource allocation technique based on reinforcement learning (RL). Reinforcement learning is a subset of machine learning that essentially employs an agent to act in the environment and gather rewards, both positive and negative. We will be using the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm as it's an improvement to its predecessor algorithm (DDPG). We will be tweaking the parameters in RL for better efficiency: Latency, Throughput, energy efficiency, fairness, and rewards.

Keywords: NB-IoT, LPWAN, Long-Term Evolution (LTE), Reinforcement Learning, TD3.

I. INTRODUCTION

Narrowband Internet of Things (NB-IoT) is a cellular technology designed to provide a cost-effective and low-power consuming connectivity among IoT devices. It was developed by the third-generation partnership program (3GPP). Additionally, it was engineered to offer expanded coverage while operating harmoniously alongside the Long-Term Evolution (LTE) standard. [1]. Various Telecommunications companies and network operators across the world deployed NB-IoT shortly after its release and were later integrated with the 5G network. IoT device manufacturers, chip and module vendors, and technology providers have also contributed to the expansion of NB-IoT. Since its creation, NB-IoT has been known for co-existing with other cellular technologies enabling efficient use of spectrum resources and allowing IoT devices to operate alongside other wireless services without causing interference. NB-IoT is utilized in three operational modes: (1) as a stand-alone system on a dedicated frequency band, (2) within the existing wideband LTE carrier's occupied spectrum, and (3) within the guard band of an established LTE carrier [2].

NB-IoT has a spectral occupation of 180 KHz (both uplink and downlink) with flow rates of at most 250 kbit/s. Even though the flow rates are lower, these speeds are adequate for this protocol’s specific applications. There are certain ways of improving the efficiency of NB-IoT. It is done by tweaking certain parameters like latency, throughput, energy efficiency, and Resource block allocation. NB-IoT’s uplink and downlink have their sub-carriers like the NPDCCH (Narrow-Band Physical Downlink Control Channel) subcarrier, NPUSCH (narrow-band physical uplink shared channel) subcarrier, NPDSCH (Narrowband Physical Downlink Shared Channel), and so on. There are 12 subcarriers in the uplink, and each has a bandwidth of 15 KHz [3]. Resource allocation is usually done with the help of resource blocks or with the help of subcarriers (also known as tones). Resource Block is a concept that involves the allocation of a group of subcarriers in both the frequency and time domains for data transmission. Subcarriers are individual frequency components within an OFDM (Orthogonal Frequency Division Multiplexing) signal and are used for data transmission within each OFDM symbol.

Although NB-IoT is designed for low-rate, time-insensitive applications, improved radio resource management will guarantee the best possible resource utilization for anticipated throughput and coverage improvement if the necessary performance parameters are fulfilled. A novel definition of the paging resource set and matching resource selection procedure are included in the proposed resource allocation strategy [4]. Massive connection paging information transmission overloads the allocated resources, leading to more padding bits and inefficient use of those resources. It is just used for paging traffic unloading.

NB-IoT systems need to be equipped to make better decisions in future wireless networks, and existing algorithms do not meet this need.

This paper [5] offers a broad summary of the NB-IoT innovation's resource allocation solutions and makes additional recommendations and arguments for the intelligence of upcoming resource allocation solutions. Machine learning techniques have been used to investigate alternative solutions; nonetheless, more sophisticated solutions are needed. Reinforcement learning has been used in relatively few publications, such as [6] and [7], for effective uplink transmission and access control in NB-IoT. In this paper, we further the research on using RL-based algorithms for effective resource allocation.

II. ABOUT REINFORCEMENT LEARNING

RL is one of the three main machine learning methods. The three methods are supervised learning, non-supervised learning, and reinforcement learning. Reinforcement learning works as follows: There are two components in this learning system. They are the Agent and the Environment. The Agent observes the environment and makes the best possible move to get the maximum reward, then the environment changes its state and lets the agent appraise the new state and the possible rewards. This learning is a trial-and-error method learning, as the agent itself is prone to make mistakes and gain experience from it. The agent constantly appraises its environment, revises its action strategy to adapt to the environment, and takes action and gets the reward and experience from it [8]. Reinforcement learning is defined by the Markov Decision Process (MDP). This framework outlines an agent's interactions and introduces the concept of the value function to facilitate learning. In reinforcement learning, the utilization of Markov Decision Processes (MDP) and the value function is integral to the construction of the Bellman equation. The Bellman equation encapsulates the recursive relationship between the value of states or state-action pairs, forming the basis for Q-learning—a method employed to solve the Bellman equation and optimize decision-making strategies in sequential decision problems [16].

The goal of Reinforcement Learning is to find a policy that can be used to gain maximum reward. The policy is a strategy or mapping from states to actions that guide the agent's decision-making. The formula for policy is:

$$\pi(\mathrm{a} \mid \mathrm{s}) = \mathrm{P}[\mathrm{At}_{\mathrm{t}} = \mathrm{a} \mid \mathrm{St} = \mathrm{s}] \tag{1}$$

where π = probability of policy, At_{t} = action at time t, St = state at time t, a = action performed in state ' s '.

The Value function is the cumulative reward to be received if the agent follows the policy from its current state. The formula for the value function is:

$$\mathrm{V}\pi(\mathrm{s}) = \mathrm{E}\pi[\mathrm{Rt} + 1 + \gamma \mathrm{V}\pi(\mathrm{St} + 1) \mid \mathrm{St} = \mathrm{s}] \tag{2}$$

Where $\mathrm{V}\pi$ = expected equation, $\mathrm{E}\pi$ = expected value, γ = Discount factor, $\mathrm{Rt} + 1$ = next reward value.

Bellman’s equation represents the relationship between the value functions of the current and the next state. There are two equations:

$$\mathrm{v}_{\pi'}(\mathrm{s}) = \sum_{\mathrm{a}\in \mathrm{A}} \pi(\mathrm{a} \mid \mathrm{s}) \left(\mathrm{R}_{\mathrm{t}+1} + \gamma \sum_{\mathrm{s}'\in \mathrm{S}} P^{\mathrm{a}}_{\mathrm{ss}'} \mathrm{v}_{\pi}(\mathrm{s}')\right) \tag{3}$$

Where $\mathrm{V}\pi'(\mathrm{s})$ = Bellman's expectation equation.

$$\mathrm{v}^{*}(\mathrm{s}) = \max \mathrm{E}\pi[\mathrm{Rt} + 1 + \gamma \mathrm{v}^{*}(\mathrm{St} + 1) \mid \mathrm{St} = \mathrm{s}] \tag{4}$$

Where $\mathrm{V}^{*}(\mathrm{s})$ = Bellman's optimality equation.

Reinforcement learning uses many types of algorithms like SARSA (State-Action-Reward-State-Action), Q-learning, Temporal Difference (TD), and function approximation [8]. We’ll be focusing on Q-learning. Q-learning is an off-policy reinforcement algorithm proposed by Christopher Watkins in 1989. It is an incremental (stochastic approximation) method for estimating the Q-function in a Markov decision process (MDP) [9].

In this paper, we’ll be using Twin Delayed Deep Deterministic Policy Gradient (TD3): An algorithm based on Q-learning. This algorithm is built on the DDPG, and it was an upgrade to its predecessor, with increased stability, performance, and consideration of function approximation error [10]. TD3 also resolves an issue in DDPG, in which the learned Q-function dramatically overestimates Q-values, eventually leading to breaking policies. This is resolved by introducing three tricks [15]: (1) Clipped Double-Q Learning: TD3 trains with a pair of Q-functions, introducing a "twin" approach, and it utilizes the lower of the two Q-values to construct the targets in the Bellman error loss functions. (2) Delayed Policy Updates: TD3 adjusts the policy (along with the target networks) at a lower frequency compared to the Q-function. The suggested practice, as outlined in the paper [15], is to perform one policy update for every two Q-function updates. (3) Target Policy Smoothing: In TD3, noise is introduced to the target action to increase the difficulty for the policy to exploit Q-function errors. This is done to smooth out the Q-value along with changes in action, preventing the policy from taking advantage of inaccuracies in the Q-function [15].

This study focuses on the difficult issue of controlling NPUSCH transmissions, which includes scheduling which devices can transmit, determining their link adaptation parameters, and allocating resources to each NPUSCH transmission.

III. SPECIFICATION

A base station that provides service to several NB-IoT devices is part of the system. The devices need to finish both a connection procedure and a random-access procedure using NPRACH resources. After connecting, they watch for an uplink grant from the base station so they can send data over an NPUSCH. With a 180 KHz bandwidth, the system uses one carrier for each transmission direction. Ten subframes, each lasting one millisecond, make up each frame that makes up the time dimension.

Devices that have transmitted preambles receive a Random-Access Response (RAR) message from the base station, which gives them access to uplink resources so they can send their connection request message. The RAR message contains uplink grants that give the UE (User Equipment) the uplink resources it needs to send its connection request message, as well as a Temporary C-RNTI (Cell Radio Network Temporary Identifier) that is specific to each device. Twelve 15 kHz subcarriers make up the carrier bandwidth that isn't set aside for NPRACH (Narrowband Physical Random-Access Channel). For two types of transmissions, the base station needs to allot time-frequency resources of the uplink carrier: (i) the previously mentioned UE connection requests; and (ii) NPUSCH (Narrowband Physical Uplink Shared Channel) transmissions containing UE data. This distribution needs to prevent any overlap with NPRACH resources.

Devices receive control messages via the downlink carrier's NPDCCH (Narrowband Physical Downlink Control Channel). The NPDCCH is used to transmit specialized control messages known as Downlink Control Information (DCI) Format N0, which contain the uplink data grants (alternate formats serve different purposes). Depending on the destination device's path loss, multiple DCI messages are sent.

Several UEs share the NPUSCH channel, which is used to transmit uplink data. Each UE receives time-frequency resources for transmission from the base station based on a scheduling algorithm that considers priority and channel quality. The scheduling algorithm seeks to balance UE fairness with maximizing the system throughput. The modulation and coding scheme (MCS) is also modified by the base station using link adaptation techniques to optimize the data rate while preserving a certain degree of reliability. The channel quality indicator (CQI), which is a measurement of the signal-to-noise ratio (SNR) and interference level, is provided by the UE and is used to choose the MCS. The Shannon formula can be used to determine the data rate:

$$C = B\log 2(1 + SNR) \qquad (5)$$

Where B is the bandwidth in hertz, SNR is the signal-to-noise ratio, and C is the channel capacity in bits per second. To increase transmission reliability, the base station additionally employs hybrid automatic repeat request, or HARQ, which retransmits packets that are incorrectly received. By combining retransmission and forward error correction (FEC), HARQ can achieve high overhead with minimal reliability.

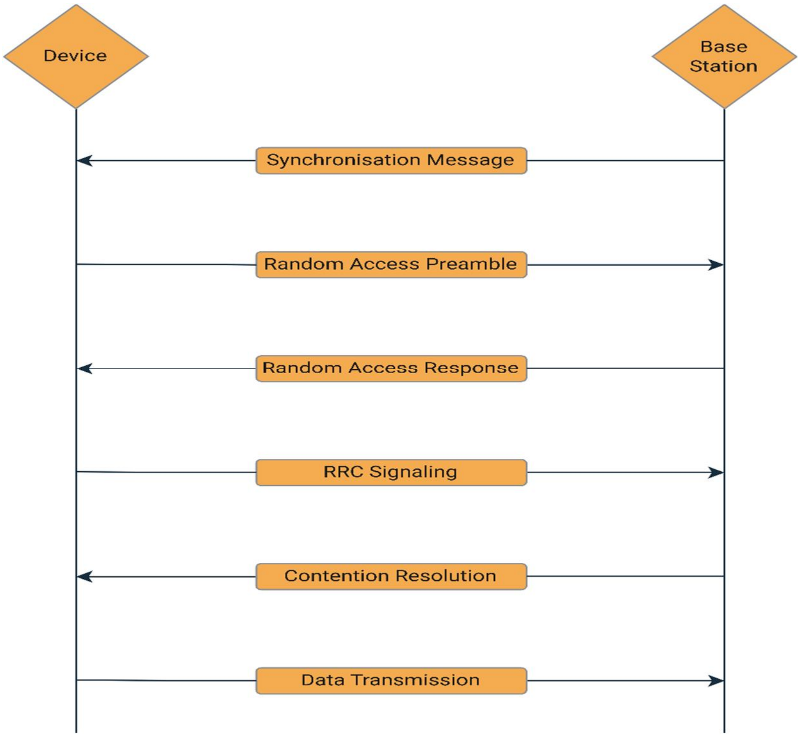


Fig. 1 Random Access Procedure Flowchart

IV. PROBLEM STATEMENT

We suggest using reinforcement learning techniques, which appear to have not been thoroughly investigated in NB-IoT transmission control despite their present popularity. The heuristic suggested in [11] could easily work with our transmission control method, in contrast to attempts to apply RL in the setup of random-access parameters, particularly for the allocation of NPRACH resources of each CE level [12].

As noted in [12], real network conditions can differ dramatically from the simulated environment after deployment, rendering control rules worthless. Therefore, RL algorithms are usually required to be trained offline in a simulator before being deployed. One possible solution is to set up the algorithms to self-tune after deployment, but this raises questions about how long the tuning process would take and how it might affect network performance. Our objective is to suggest a system that, once implemented in the network, can learn on its own (online learning). Our numerical results confirm that current RL algorithms cannot perform this task without worsening the transmission delay while they are still learning. The low sample efficiency of model-free RL algorithms is the cause. They must investigate a variety of policies before settling on an effective one, which necessitates choosing extremely ineffective courses of action over an extended length of time.

Prior research on creating online learning algorithms to manage network functions has been done by [13], which suggests a model-based reinforcement learning approach based on a novel modeling strategy consisting of a self-assessment mechanism and a classifier based on a kernel.[14] employs particular ad hoc procedures derived from sequential likelihood ratio calculations and [6] suggests a novel model-based reinforcement learning algorithm for link adaptation that has a high sample efficiency.

Our goal is to create an algorithm that can learn how to minimize the average transmission delay by controlling uplink data transmissions. The radio access network's algorithm needs to be able to learn online. This suggests that the algorithm needs to make irreversible decisions, track the outcomes of those decisions, and adjust its control strategy as necessary.

V. METHODOLOGY

In this paper, we will be comparing the efficiency of the algorithms DQN (Deep Q-Networks), PPO (Proximal Policy Optimization), and TD3 (Twin Delayed Deep Deterministic Policy Gradient) under different factors. The factors are Throughput, Latency, Fairness, Energy efficiency, and Total Rewards.

DQN is a value-based method that is designed for discrete action spaces but can be used for continuous action spaces. DQN requires more experience to work as intended, and the experience is only generated over time. The behavior of a DQN agent is unpredictable as it is prone to make more random decisions to get familiar with the unknown parts of the environment. This makes it difficult to understand whether the actions done were intentional or random [19].

PPO (Proximal Policy Optimization) is a model-free DDPG (Deep Deterministic Policy Gradient) algorithm that focuses on optimizing policies/strategies so that the expected cumulative reward is maximized [6]. While being a successful algorithm for continuous action spaces, its optimization behavior still runs a risk of performance instability [17].

TD3 is an extension of the DDPG algorithm. It combines value-based and action-based methods. It employs two critic networks to approximately find the value function and an actor-network to find the policy. It is particularly created for continuous action spaces [18].

VI. SIMULATION ENVIRONMENT

The Python-based simulation environment is intended to simulate the resource allocation dynamics of an NB-IoT (Narrowband Internet of Things) network. The environment consists of several components, such as a population of devices that are attempting to connect to the system and send packets of data. The base station is a key element; it's in charge of channel coordination, scheduling transmission opportunities, and access management.

A device goes through several stages once it is active. It must first finish the access process, which entails sending a preamble sequence selected at random over the NPRACH. During the RAR window, contention resolution takes place when the base station transmits a signaling message to the device. When a signaling exchange is successful, the device becomes connected. Before making another access attempt, a backoff period is started in the event of a collision or poor signal quality.

Following a successful connection, a device watches for an uplink grant that a DCI indicates. The device sends its data packet to the designated NPUSCH after obtaining the grant. Depending on the decoding status, the device waits for an acknowledgment (ACK) or another DCI for retransmission. When you receive an ACK, the connection is closed.

VII. PARAMETERS

TABLE 1
NB-IOT PARAMETERS

PARAMETER	VALUE
FREQUENCY BANDS	[800,900,1800] MHZ
BANDWIDTHS	[1.4,3.75] MHZ
DATA RATES	[200,250] KBPS
TRANSMISSION POWER LEVELS	[0,5,10] DBM
NOISE POWER	-120.0 DBM

TABLE 2
STATE AND ACTION SPACES

PARAMETER	VALUE
NUMBER OF USERS	100
NUMBER OF CHANNELS	10
MINIMUM RESOURCE UNITS	0
MAXIMUM RESOURCE UNITS	10

TABLE 3
TIME PARAMETERS AND DEVICE BEHAVIOR

PARAMETER	VALUE
CURRENT TIME STEPS	0
MAXIMUM TIME STEPS	100
INITIAL ENERGY EFFICIENCY FINAL ENERGY EFFICIENCY	1.0 0.2

TABLE 4
PATH LOSS MODEL PARAMETERS

PARAMETER	VALUE
PATH LOSS EXPONENT PATH LOSS REFERENCE DISTANCE	2.0 1.0 METRES
BATCH SIZE	64
MAXIMUM EPISODES	1000

TABLE 5
TD3 AGENT-SPECIFIC PARAMETERS

PARAMETER	VALUE
Critic networks: Learning rate Actor networks: Learning rate Policy noise	0.001 0.001 0.2
Noise clip	0.5
discount	0.99

VIII. ALGORITHM

Input: Initialize the variables and the values given in Tables 1 to 5. Instantiate lists to store entropy and alpha values. Initialize lists to store the values of the following metrics: latency per episode, fairness per episode, throughput per episode, energy efficiency per episode, rewards per episode, and episode data. Initialize lists to store action distributions for each episode. Initialize lists to store action distributions for each episode. Initialize energy efficiency increment.

Output: Graphical representation of Energy efficiency, Throughput, Latency, Fairness, Episode vs Reward, and Resource allocation.

Algorithm 1 Training TD3 agent

```
Initialize TD3 agent and replay buffer
for episode = 1 to max_episode do
    Obtain the initial state after resetting the environment
    total_reward = 0
    Initialize a list to accumulate experiences for batch training
    while True do
        Obtain the value of an action by passing state value through a policy network, adding noise to it, clips result to a specific range, and get a float output
        Obtain the values for next state, reward, and done (Confirmation if maximum episode count is reached)
        Append the values of state, action, reward, next_state, and done to list "episode_experiences"
        state = new_state
        total_reward = total_reward + reward
        if done is True then
            break
        end if
    end while
    Append the list "episode_experiences" to list "replay_buffer"
    Train the TD3 agent with the values of "replay_buffer" and "batch_size"
    Calculate latency for this episode
    Latency = $\frac{\text{Total Communication Delay}}{\text{Number of Data Transmissions}}$
    Extract resource allocations from the replay buffer entries
    Calculate fairness using Jain's fairness index
    Fairness Index = $\frac{(\sum_{i-1}^{N} \text{Throughput }_i)^2}{N\sum_{i-1}^{N} \text{Throughput }_i)^2}$
    Append fairness to the list "fairness_per_episode"
    total_resource_units = sum of all action values
    bit_rate_per_resource = 200
    Append the value of throughput to list "throughput_per_episode"
    Throughput = $\frac{\text{Total Data Successfully Transmitted}}{\text{Total Time}}$
    Convert total energy consumed to milliwatts
    Energy Efficiency = $\frac{\text{Throughput}}{\text{Power Consumption}}$
    Append total_reward to rewards_per_episode
    Append the episode's data to the dictionary "episode_data"
    Append values of resource allocations to the list "episode_action_distributions"
    Print episode+1, total_reward, energy_efficiency, total_latency, achieved_data_throughput, total_latency, fairness_index
    Plot throughput, latency, fairness, episode vs reward, latency, fairness, action distribution evolution
end for
```

IX. SIMULATION RESULTS AND ANALYSIS

A. *Total Reward*

Total reward refers to the cumulative reward obtained by the agent during the course of training or executing episodes. During training, the agent interacts with the environment, taking actions based on its current state, receiving rewards as feedback, and updating its policy accordingly. At each time step within an episode, the agent receives a reward from the environment, and these rewards are accumulated over the entire episode to compute the total reward for that episode.

In the context of NB-IoT (Narrowband Internet of Things) resource allocation, our TD3 (Twin Delayed Deep Deterministic Policy Gradient) algorithm demonstrates superior performance compared to existing DQN (Deep Q-Network) and PPO (Proximal Policy Optimization) methods, as evidenced by the Fig 2. TD3 consistently achieves higher cumulative rewards across training episodes, indicating more effective learning and better performance in optimizing resource allocation for NB-IoT systems. This outperformance suggests that TD3 is better suited for addressing the complexities and challenges of NB-IoT resource allocation tasks, offering improved efficiency and effectiveness in managing resources and enhancing overall system performance.

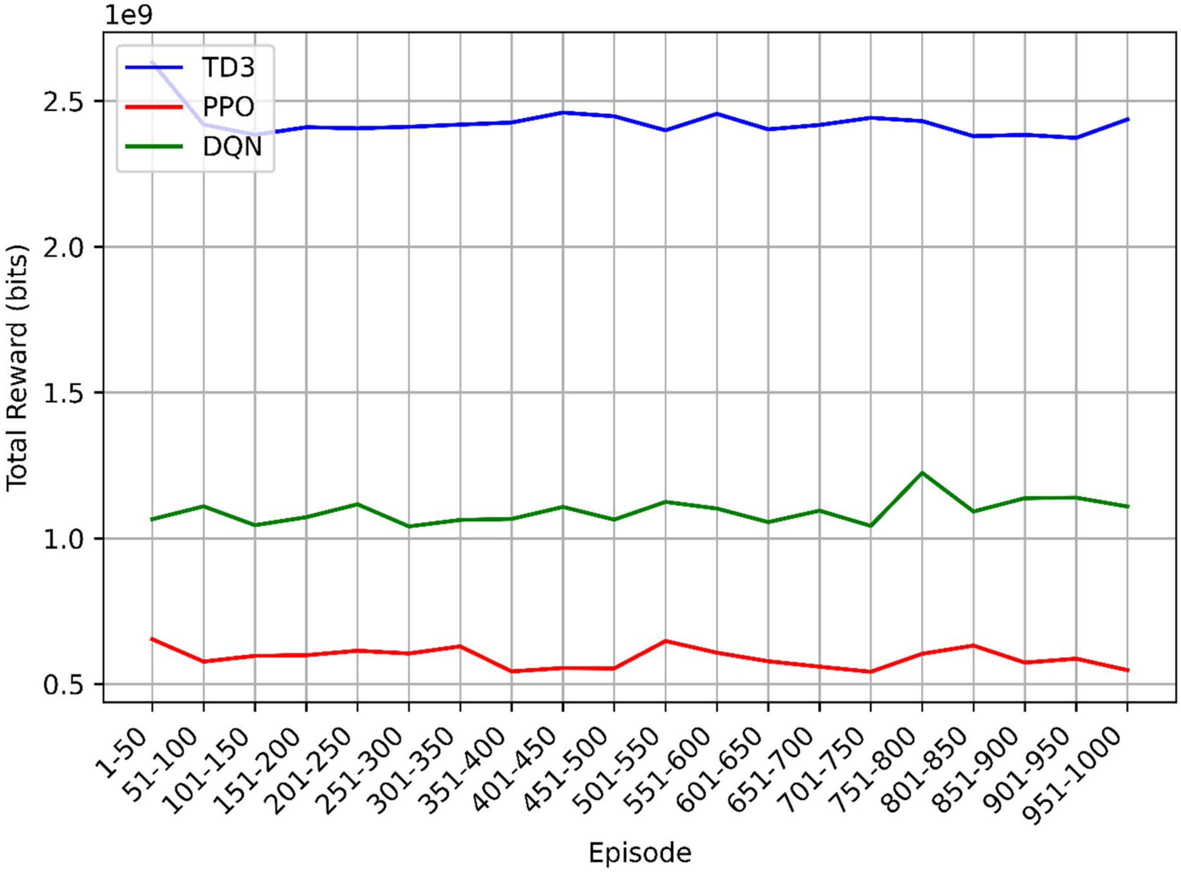


Fig. 2 Total Reward Versus Episode

B. *Energy Efficiency*

The ability of a system or algorithm to accomplish its goals while consuming the least amount of energy is referred to as energy efficiency. It entails maximizing the intended output or performance per unit of energy consumed through resource optimization. In Fig 3, which is after averaging, TD3's energy efficiency remained nearly constant at 7.2 bits/joule for the duration of the simulation, indicating consistently high performance. PPO showed a waveform that was mostly below that threshold, with occasional peaks that matched the TD3 algorithm to indicate periods of comparable efficiency. When compared to TD3 and PPO, DQN consistently displayed the lowest energy efficiency, indicating a higher energy consumption.

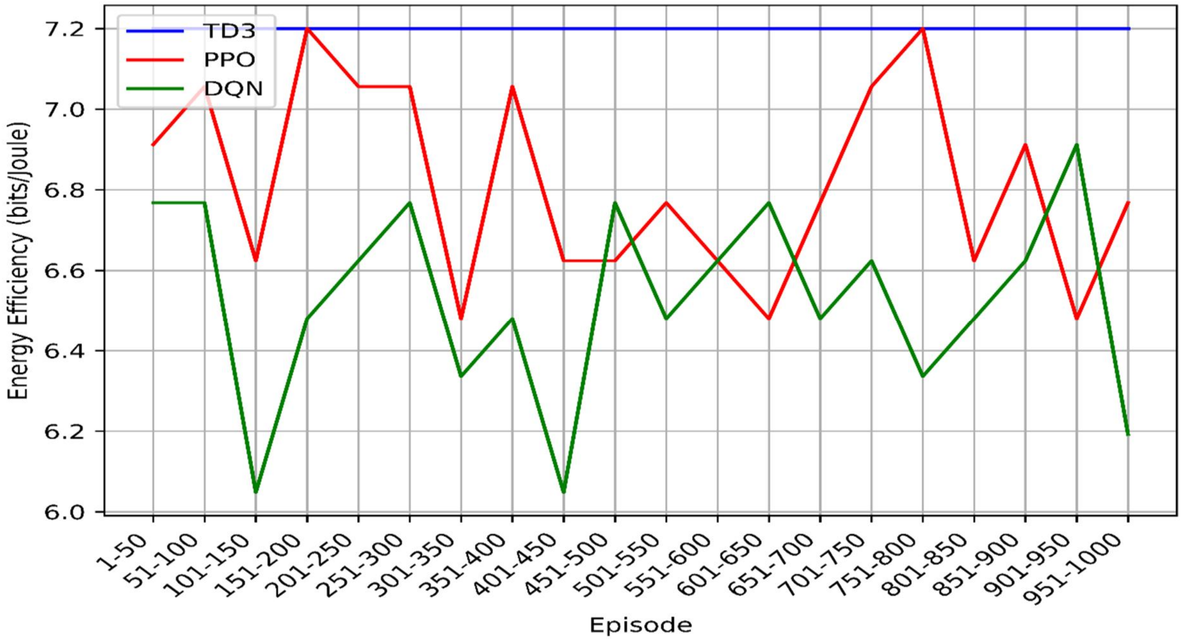


Fig. 3 Energy Efficiency Versus Episode

C. *Throughput*

Throughput refers to the amount of information or data processed per unit of time, and an increased throughput indicates that the algorithm is processing and managing a greater amount of data effectively. In Fig 4, the TD3, DQN, and PPO algorithms were compared over a thousand episodes, and with a waveform peak that was higher than 3000 bits, TD3 demonstrated its superior performance in terms of throughput optimization. while DQN trailed closely behind and showed strong simulation efficiency, the PPO trailed behind and showed relatively lower throughput.

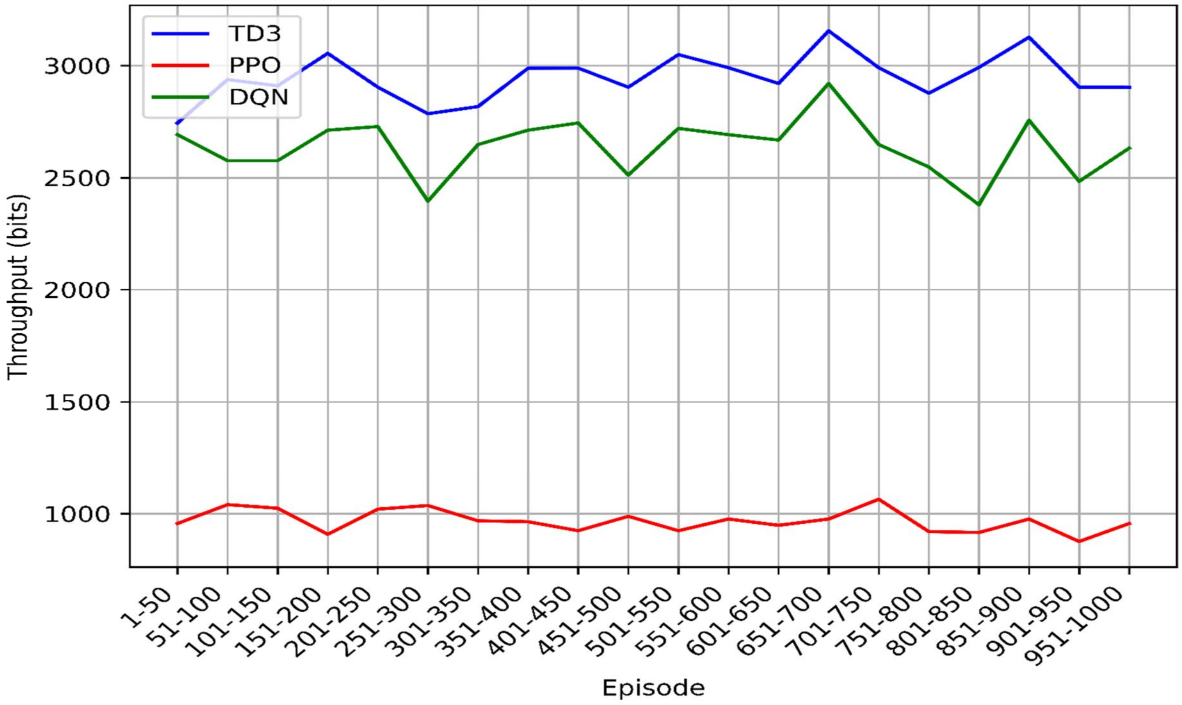


Fig. 4 Throughput Versus Episode

D. Fairness

In general, Fairness refers to the distribution of resources among entities without any bias. Throughput and Fairness are frequently complicated trade-offs that rely on the needs and priorities of the system or application. In Fig 5, in evaluating the Fairness of TD3, PPO, and DQN algorithms over 1000 episodes, TD3 treated all entities equally during the simulation, maintaining a high level of Fairness and a steady waveform at 1.0. Meanwhile, DQN, which showcased better throughput, now showed the lowest Fairness among the three algorithms. PO showed a significant fairness next to TD3.

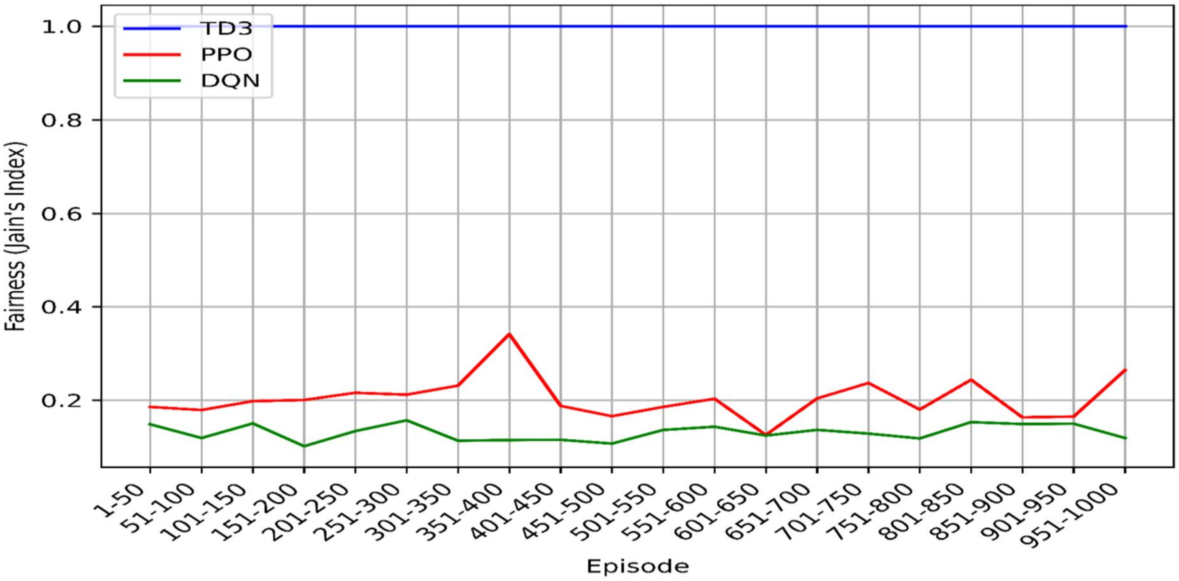


Fig. 5 Fairness Versus Episode

E. Latency

The term "latency" describes the interval of time that elapses between the start of a resource request and the real delivery or accomplishment of the associated task. Low latency is preferred for resource allocation because it guarantees timely resource allocation, which accelerates task completion and enhances user experience. After analyzing Fig 6, with the lowest latency values, TD3 demonstrated its effectiveness in reducing delays between episodes. Conversely, PPO and DQN demonstrated increased latency values, indicating relatively longer response times. According to these results, TD3 minimizes latency better than PPO and DQN, which makes it a good option in situations where quick responses are essential.

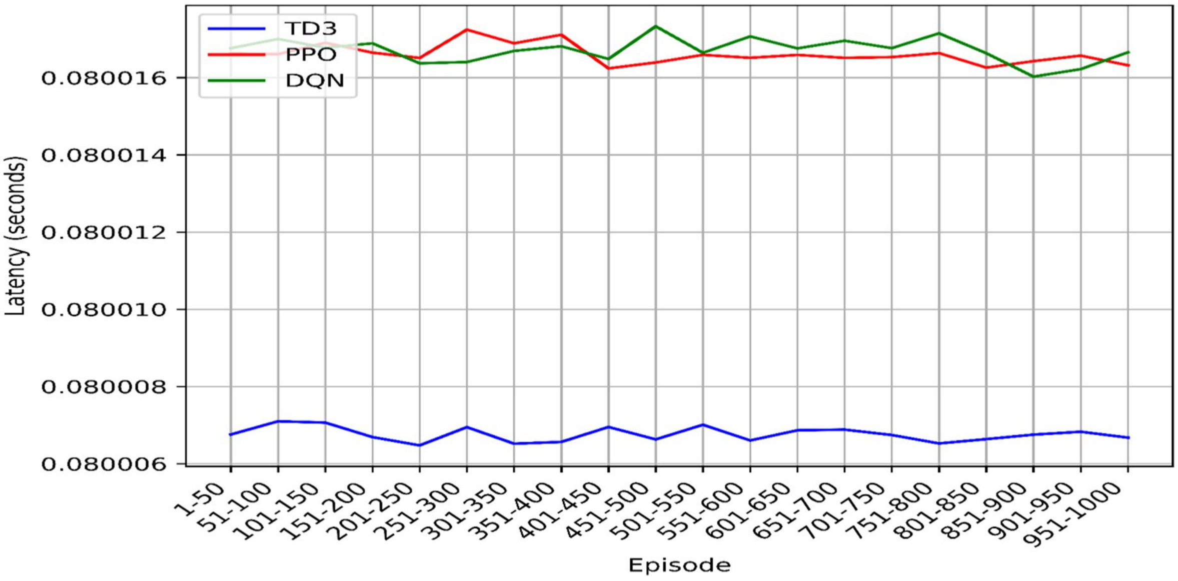


Fig. 6 Latency Versus Episode

F. Recovery Time

Recovery time is crucial for assessing the efficiency of reinforcement learning algorithms such as TD3, PPO, and DQN. It reflects how quickly each algorithm can recover and stabilize performance after training episodes, indicating learning speed and policy optimization. In Fig 7, TD3 excels with the lowest recovery times, averaging around 2.99 seconds, showcasing its quick adaptation abilities. PPO follows, starting at 3.67 seconds and showing consistent performance with minor fluctuations. In contrast, DQN has the highest recovery times, beginning at approximately 3.97 seconds and remaining above 3.95 seconds, indicating challenges in learning efficiency. Overall, TD3's superior recovery time makes it ideal for rapid decision-making applications, while DQN may need improvements to compete effectively.

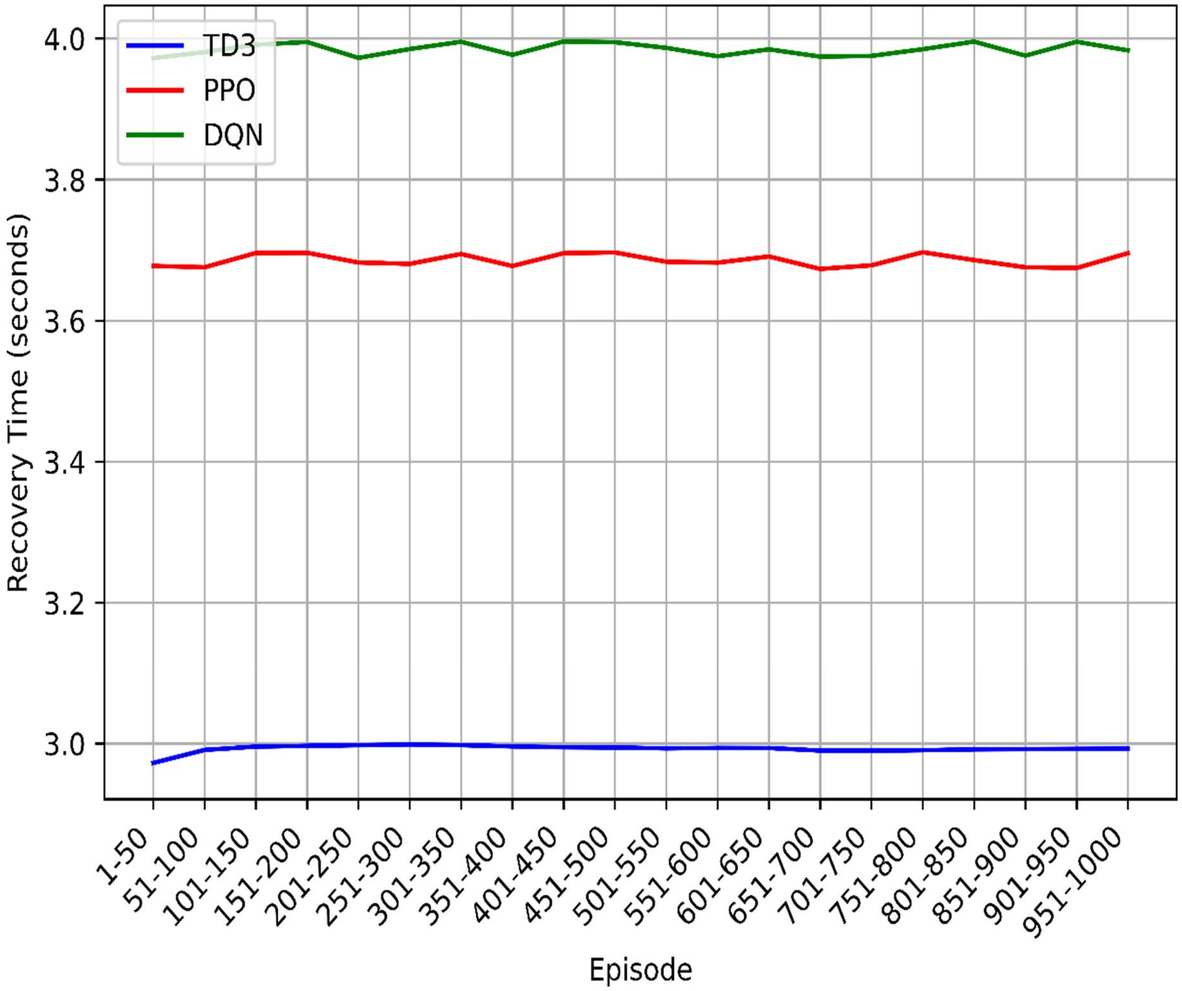


Fig. 6 Recovery Time Versus Episode

G. Resource Allocation (Channels) Over Episodes

In our NB-IoT resource allocation framework, we conducted experiments to analyze the resource allocation patterns across the 10 available channels. Through our training process using the TD3 algorithm, we recorded the resource allocation values for each channel at the end of every episode. Subsequently, we visualized this data by plotting the resource allocation against the training episodes for all 10 channels. Fig 8 offers insights into how the resource allocation strategies evolve throughout training and provides a comprehensive overview of the allocation dynamics across different channels. Such analysis allows for a deeper understanding of the algorithm's behavior and its ability to effectively distribute resources among multiple channels in NB-IoT environments.

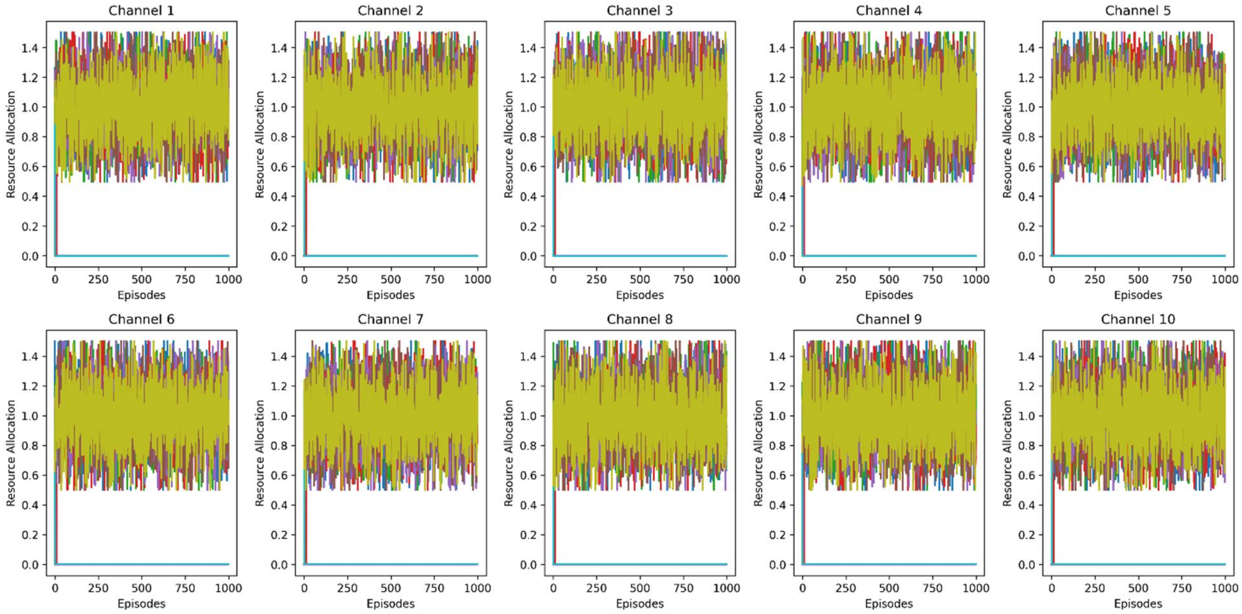


Fig. 8 Resource Allocation (Channels) Over Episodes

X. CONCLUSION AND FUTURE WORK

In conclusion, this research article explores the application of reinforcement learning, specifically the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm, for resource allocation in Narrowband Internet of Things (NB-IoT) systems. Through a comprehensive analysis and comparison with existing algorithms such as DQN and PPO, TD3 demonstrates superior performance across various metrics including total reward, energy efficiency, throughput, fairness, and latency. The results highlight TD3's effectiveness in optimizing resource allocation, managing data transmission, and minimizing delays in NB-IoT environments. Additionally, the visualization of resource allocation patterns across multiple channels provides valuable insights into the algorithm's behavior and its ability to adapt to different scenarios. Overall, the findings suggest that TD3 offers a promising solution for enhancing the efficiency and effectiveness of resource allocation in NB-IoT systems, paving the way for improved performance and reliability in IoT applications. Further research and experimentation could delve deeper into fine-tuning parameters and exploring additional optimization strategies to further enhance the capabilities of TD3 in real-world deployment scenarios.

REFERENCES

[1] Wang Y-P Eric, Lin Xingqin, Adhikary Ansuman, et al. A primer on 3GPP narrowband Internet of Things. IEEE Communications Magazine. 2017; 55(3): 117-123.

[2] Ratasuk Rapeepat, Vejlgaard Benny, Mangalvedhe Nitin, et al. NB-IoT system for M2M communication. 2016 IEEE Wireless Communications and Networking Conference. IEEE; 2016: 1-5.

[3] Yu Ya-Ju, Wang Jhih-Kai. Uplink resource allocation for narrowband Internet of Things (NB-IoT) cellular networks. 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). IEEE; 2018: 466-471.

[4] Liu Jiahui, Mu Qin, Liu Liu, et al. Investigation about the paging resource allocation in NB-IoT. 2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC). IEEE; 2017: 320-324.

[5] Muteba KF, Djouani K, Olwal TO. Opportunistic resource allocation for narrowband Internet of Things: A literature review. 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE). IEEE; 2020: 1-6.

[6] Alcaraz Juan J, Losilla Fernando, Gonzalez-Castaño Francisco-Javier. Transmission Control in NB-IoT with Model-Based Reinforcement Learning. IEEE Access. 2023.

[7] Hadjadj-Aoul Yassine, Ait-Chellouche Soraya. Access control in nb-iot networks: A deep reinforcement learning strategy. Information. 2020; 11(11): 541.

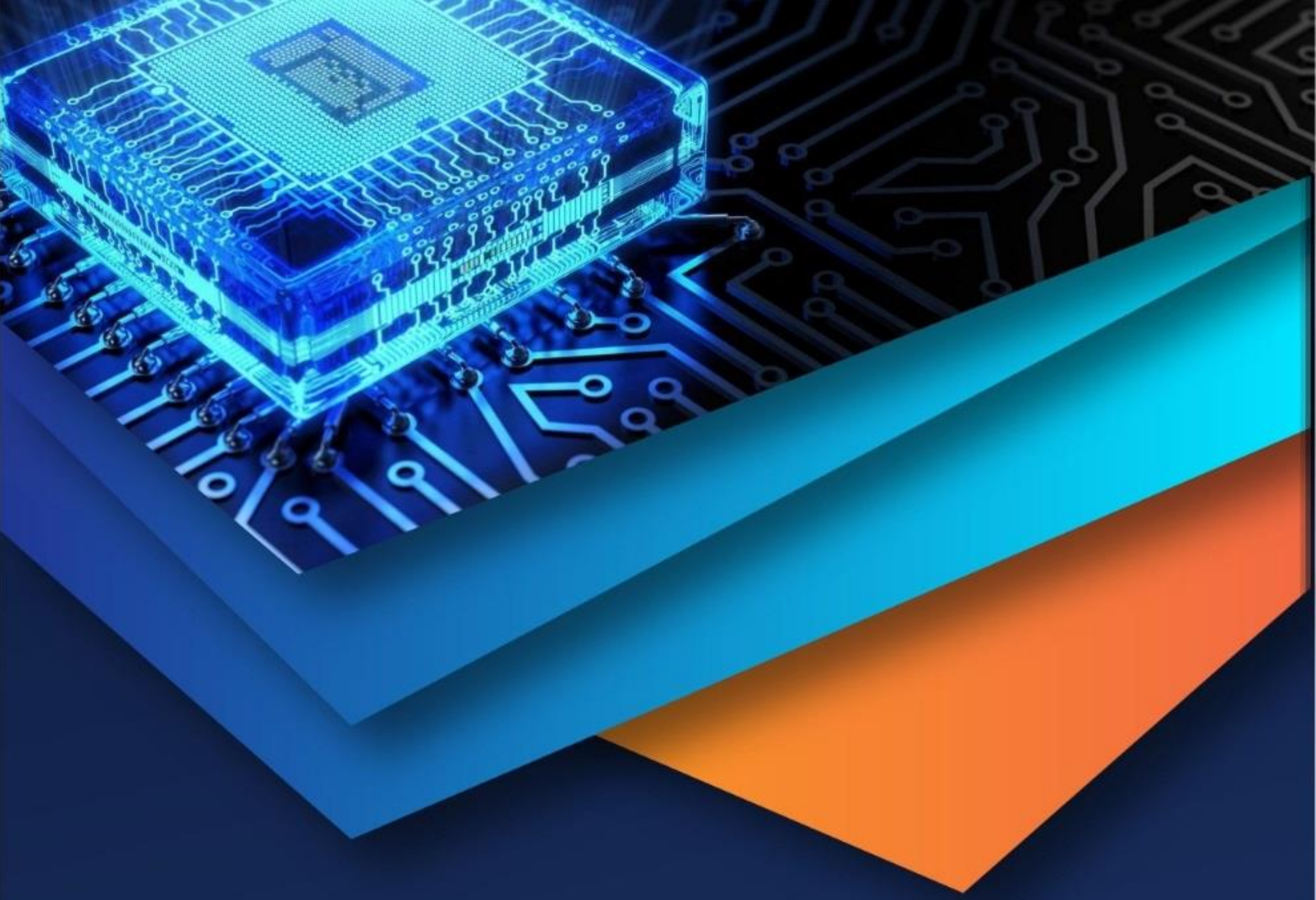
[8] Qiang Wang, Zhongli Zhan. Reinforcement learning model, algorithms and its application. 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC). IEEE; 2011: 1143-1146.

[9] Clifton Jesse, Laber Eric. Q-learning: Theory and applications. Annual Review of Statistics and Its Application. 2020; 7: 279-301.

[10] Dankwa Stephen, Zheng Wenfeng. Twin-delayed ddpg: A deep reinforcement learning technique to model the continuous movement of an intelligent robot agent. Proceedings of the 3rd international conference on vision, image, and signal processing. 2019: 1-5.

[11] Harwahyu Ruki, Cheng Ray-Guang, Liu Da-Hao, et al. Fair configuration scheme for random access in NB-IoT with multiple coverage enhancement levels. IEEE Transactions on Mobile Computing. 2019; 20(4): 1408-1419.

[12] Jiang Nan, Deng Yansha, Nallanathan Arumugam, et al. Reinforcement learning for real-time optimization in NB-IoT networks. IEEE Journal on Selected Areas in Communications. 2019; 37(6): 1424-1440.
[13] Alcaraz Juan J, Losilla Fernando, Zanella Andrea, et al. Model-based reinforcement learning with kernels for resource allocation in RAN slices. IEEE Transactions on Wireless Communications. 2022; 22(1): 486-501.
[14] Alcaraz Juan J, Ayala-Romero Jose A, Vales-Alonso Javier, et al. Online reinforcement learning for adaptive interference coordination. Transactions on Emerging Telecommunications Technologies. 2020; 31(10): e4087.
[15] Fujimoto Scott, Hoof Herke, Meger David. Addressing function approximation error in actor-critic methods. International conference on machine learning. PMLR; 2018: 1587-1596.
[16] Jang Beakcheol, Kim Myeonghwi, Harerimana Gaspard, et al. Q-learning algorithms: A comprehensive classification and applications. IEEE Access. 2019; 7: 133653-133667.
[17] Wang Junpeng, Gou Liang, Shen Han-Wei, et al. Dqnviz: A visual analytics approach to understanding deep q-networks. IEEE Transactions on Visualization and Computer Graphics. 2018; 25(1): 288-298.
[18] Wang Yuhui, He Hao, Tan Xiaoyang. Truly proximal policy optimization. Uncertainty in Artificial Intelligence. PMLR; 2020: 113-122.
[19] Wu Jiaolv, Wu QM Jonathan, Chen Shuyue, et al. A-TD3: An Adaptive Asynchronous Twin Delayed Deep Deterministic for Continuous Action Spaces. IEEE Access. 2022; 10: 128077-128089.



10.22214/IJRASET

INDEX COPERNICUS

45.98

IMPACT FACTOR:
7.129

TOGETHER WE REACH THE GOAL

IMPACT FACTOR:
7.429