ORIGINAL RESEARCH PAPER
Pages: 272-286

ARL-RA: Efficient Resource Allocation in 5G Edge Networks: A Novel Intelligent Solution UsingApproximate Reinforcement Learning Algorithm

M. Khani[1], S. Jamali[2*], and M.K. Sohrabi[1]
1.Department of Computer Engineering, Semnan Branch, Islamic Azad University, Semnan, Iran
2.Department of Computer Engineering, University of Mohaghegh Ardabili, Ardabil, Iran
mohsenkhani622@gmail.com, jamali@uma.ac.ir, amir_sohraby@aut.ac.ir
Corresponding author: jamali@uma.ac.ir

DOI: 10.22070/JCE.2023.17967.1251

Abstract- The rapid proliferation of fifth-generation (5G) technology has resulted in a wide range of applications, posing challenges in managing network resources effectively and efficiently. To address these challenges, network slicing (NS) and Fog-Radio Access Networks (F-RAN) have emerged as key technologies, enabling the creation of isolated virtual networks on shared physical infrastructure to support high-bandwidth and low-latency communication. However, allocating network resources to latency-sensitive applications like self-driving cars and remote surgery, while ensuring a quality experience, is complex due to stringent latency requirements and limited availability. In this paper, we propose a novel approach, leveraging the Q-learning algorithm, specifically the Approximate Reinforcement Learning for dynamic resource allocation (RA-ARL), in the context of 5G environments. Our modified algorithm takes into account crucial network attributes, introduces innovations such as service type classification based on latency sensitivity, and considers time-varying resource conditions and service demands. We propose an RL model to optimize network utility, focusing on the F-RAN model. Our experimental results demonstrate the effectiveness of ARL-RA in terms of convergence, resource utilization, and the ability to handle user request rejections. This work contributes to the advancement of efficient and effective resource allocation in dynamic 5G networks, particularly for latency-sensitive applications with stringent quality requirements.

Index Terms- Resource allocation, 5G, Approximate reinforcement learning, F-RAN, Network slicing.

Manuscript received 15-March-2021 and revised 11-May-2021,
Accepted on 13- June-2021
P-ISSN: 2322-4088
E-ISSN:2322-3936

I. INTRODUCTION

The emergence of 5G technology has led to an increase in the number and diversity of applications that can be supported by wireless networks [1]. However, managing network resources in a way that ensures optimal performance and efficient use poses challenges. Network slicing (NS) is a key technology that addresses these challenges by creating isolated virtual networks on top of shared physical infrastructure [2].

Fog-Radio Access Network (F-RAN) is a network architecture that has emerged as a solution to address the challenges posed by the emergence of 5G technology. F-RAN integrates the principles of fog computing and radio access networks to create a distributed computing environment that can provide low-latency and high-bandwidth communication to end users [3, 4].

One of the significant advantages of NS is its ability to provide low-latency services, which are essential for emerging applications such as self-driving cars, remote surgery, and Internet of Things (IoT) services. However, allocating of network resources. Ensuring the quality of experience (QoE) for all services while maintaining resource efficiency is a complex problem [5].

Addressing these challenges, machine learning (ML) methods have been proposed for dynamic resource management, including Slice Admission Control (SAC), Network Slicing (NS), and Resource Allocation (RA). However, most of these methods rely on deep learning (DL), deep reinforcement learning (DRL), and reinforcement learning (RL), which may not be suitable for the dynamic nature of 5G networks with varying parameters [6, 7]. DL-based approaches require large labeled datasets to train the models. RL-based methods may not be suitable for large and continuous environments, and DRL-based techniques require complex designs for freezing data and transferring branches of data for training the machine [8].

As we know, the core of many machine learning algorithms is the Q-learning algorithm. This algorithm is widely used due to its structure and adaptability to many desecrate environments. However, it is necessary to modify the algorithm's functions to use this algorithm in more advanced environments such as the 5G environment [9].

In this paper, we propose using the Q-learning algorithm named Approximate Reinforcement Learning for dynamic resource allocation (RA-ARL) in the 5G environment. To do so, we modify the algorithm's functions by extracting critical network features and weighing them accordingly so that the algorithm only learns the weight of each feature by performing actions in the environment and updates them by removing extra and unnecessary calculations. Additionally, we introduce several significant innovations for RA in F-RAN, including a comprehensive network model that takes into account nodes located across multiple layers, classification of service types based on latency sensitivity, and consideration of time-varying resource situations and diverse service requirements. To optimize network utility, we propose an RL model that identifies the optimal strategy and apply our proposed method to the F-RAN model.

The rest of this work is organized as follows. We investigate the literature review in the next Section. The system model is described in Section 3. In Section 4, we present RL then present how to overcome the weaknesses of RL through ARL and explain how to apply the proposed method to our model. The simulation results are summarized in Section 5. Finally, sec 6 concludes

II. LITERATURE REVIEW

Resource allocation is a crucial aspect of 5G network management that has been extensively researched in recent years [10]. Traditional approaches, such as QoS-based, game theory, and optimization methods, have been proposed for resource allocation in these networks, but they face challenges in highly dynamic and heterogeneous environments. To address these challenges, various intelligent techniques, including reinforcement learning, machine learning, and hybrid approaches, have been proposed.

Several articles have focused on using DRL algorithms to solve the RA problem in Fog-RAN. Al-Abiad et al. [11] used RL and cross-layer network coding for efficiently pre-fetching requested contents to the local caches and delivering these contents to requesting users in a downlink F-RAN with device-to-device (D2D) communications. In [12], the authors formulated the RA problem in the Markov decision problem (MDP) model and employed RL methods to solve it by consecutively assigning the Fog-Node's (FN) limited resources to IoT use cases of heterogeneous latency requirements. Khumalo et al. [13] investigated RM in F-RAN and presented the RL algorithm as a dynamic and autonomous RA method, proposing an algorithm based on Q-learning.

Xiang et al. [14] Proposed a DRL algorithm for content caching and mode selection to maximize reward performance under the dynamical channel and cache status. Authors [15] presented a DRL-based method for smart decisions on user equipment communication modes and processors' on-off states, optimized subsequently, and targeting to minimize long-term system power consumption under the dynamics of edge cache states. The authors of [16] addressed the slicing problem by allocating limited F-RAN resources to vehicular and smart city users with heterogeneous latency and computing demands in dynamic environments through an NS model based on a cluster of FNs coordinated with an edge controller to efficiently use the limited resources at the network edge.

Zhou et al. [17] proposed a DRL-based approach for optimizing cache resources in F-RAN by intelligently allocating the limited cache spaces of F-APs to different coded files based on the historical requests of the user. Article [18] proposed a DRL-based joint cache and power allocation in F-RANs to learn the user's requirements and make a smart decision for caching suitable content and allocating a significant amount of power resources. In [19], the authors used the RL method and proposed a Double Deep Q-Learning (DDQL)-based scheduling algorithm using the target network and experience replay techniques to minimize long-term service latency and computation cost under resource and deadline constraints.

The paper[20] focuses on minimizing energy consumption in fog computing by efficient task scheduling. It proposes the Binary KHA-AHA (BAHA-KHA) approach, combining the Krill Herd Algorithm (KHA) and the Artificial Hummingbird Algorithm (AHA). The goal is to minimize resource usage, task communication, and overall energy consumption. The study evaluates the BAHA-KHA model on different workflows and compares it with other algorithms. Results show that BAHA-KHA outperforms other algorithms, reducing makespan by 18% and energy consumption by 24% compared to GA.

In this article[21], the focus is on tackling the difficulties encountered in the field of IoT, specifically related to limited processing power, high latency, traffic, and energy consumption. To address these issues, a new architecture is proposed, integrating Fog Computing with the Moth-Flame Optimization algorithm and Opposition-based Learning for efficient resource allocation and job offloading within the IoT. The architecture comprises sensors, controllers, and fog servers, while the second layer adopts the subtask pool method for offloading tasks and utilizes the OBLMFO combination for resource distribution. Moreover, the second layer leverages blockchain technology to ensure the accuracy of transaction data.

In the article [21], various challenges are discussed related to IoT, including the diverse nature of IoT infrastructures, limitations in communication, transmitting data through multiple hops, and the limited energy resources in IoT devices. To address these challenges, a novel protocol is proposed for efficient data routing in IoT by taking into account energy considerations. The protocol combines the power of ML with the innovative Heat Transfer Optimizer algorithm.

In the article [21], various challenges are discussed related to IoT, including the diverse nature of IoT infrastructures, limitations in communication, transmitting data through multiple hops, and the limited energy resources in IoT devices. To address these challenges, a novel protocol is proposed for efficient data routing in IoT by taking into account energy considerations. The protocol combines the power of ML with the innovative Heat Transfer Optimizer algorithm.

Despite the proposed method being a practical and flexible approach for various applications, this article shares similarities with our previous work[22] in terms of structure and architecture. However, there are several technical and methodological differences between these two articles, which are highlighted below:

1. Our focus in this article is on resource allocation, whereas the previous work was centered aroundslice acceptance control and preventing income degradation.
2. In this article, we study five types of service types, whereas in the previous article, services were categorized into delay-sensitive and tolerance-based, and the acceptance was based on those categories.
3. The considered algorithm weight learning features differ between the two articles.

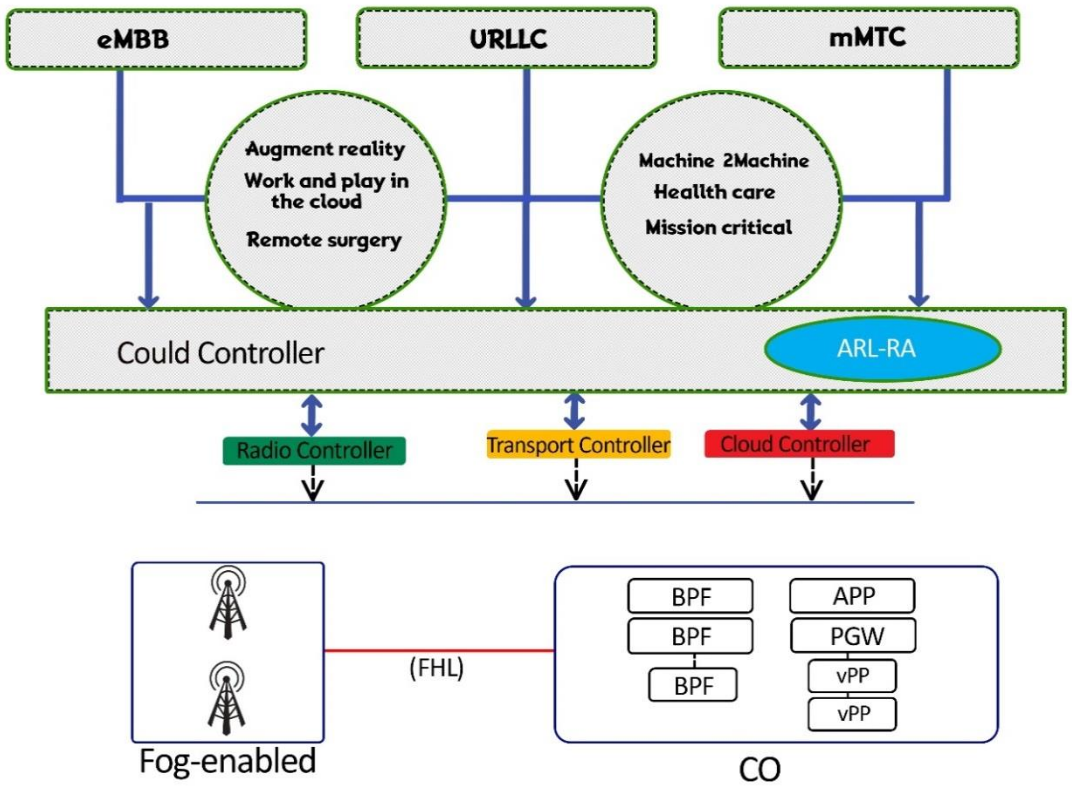

Fig. 1 System Model.

4. The simulation parameters and extracted graphs exhibit significant differences between the two articles.

III. SYSTEM MODEL AND PROBLEM STATEMENT

As shown in Fig. 1, we propose a system model that considers the limited resources of F-RAN for delivering network functionalities at the edge. The system model includes F-RAN connected to the central office (CO) via a fronthaul link [23].

The baseband processing functions (BPF) are separated from the base stations (BS) and run on special purpose processors (SPP) at CO. Additionally, the virtualized packet processor (vPP) function and packet

gateway (PGW) are virtual network functions (VNFs) running on general-purpose processors (GPPs). Assuming two types of single-antenna user equipment UEs: the eMBB UEs Z_0 and the URLLC and mMTC UEs〖 Z〗_1. The Z_0demand transfers high data rates while the UEs Z_1need delay guaranteed data transmission.

To serve these services, ARL trained in the cloud server will provide the corresponding RA. As shown in Fig. 1, service items are situated between two groups of types (e.g., AR, WPC, RS, etc.), and their required infrastructures must be provided in either exclusively F-RAN or jointly F-RAN and CO. Table 2 demonstrates the sensitivity of cases to delay and serving level. Users submit a request to the controller to

```
Algorithm 1, Q-learning for RA:
# Define the environment
n_states = # Number of possible states
n_actions = # Number of possible actions
p = # Transition probability matrix (shape: n_states x n_actions x n_states)
r = # Reward matrix (shape: n_states x n_actions)
# Initialize Q-values
Q = np.zeros((n_states, n_actions))
# Hyperparameters
gamma = # Discount factor
alpha = # Learning rate
n_episodes = # Number of episodes to run
# Q-learning algorithm
for episode in range(n_episodes):
  s = # Initial state
  done = False
whilenot done:
a = # Choose action using epsilon-greedy policy
    s_prime = # Sample the next state from the transition probability matrix
    r_t = r[s, a]
if s_prime == # Terminal state:
      Q[s, a] += alpha * (r_t - Q[s, a])
      done = True
else:
      max_Q_prime = np.max(Q[s_prime, :])
      Q[s, a] += alpha * (r_t + gamma * max_Q_prime - Q[s, a])
      s = s_prime
```

create a slice, and the controller decides whether to serve the user locally at the edge using its computing and processing resources or refer it to the CO. A sequence of requests is defined as$R_t = \{R_{0.0}.R_{0.1}.\ldots.R_{a.b}.\ldots\}$ where $R_{a.b}$ signifies the jth request of user a and is specified as $R_{a.b} = \{\nu_{a.b}, K_{a.b}.\vartheta_{a.b}\}$, which represents the data size, the required resources, and the maximum tolerable time, respectively. There exists a set of users $U = \{u_1.u_2.u_3.\ldots.u_m\}$at any given time slot t, where M is the number of users.

The total service time for a slice $s_{i.j}$ includes the remaining time $s_d^{r}$for the slice running in the selected node, $s_{i.j}^{T}$ the transmission time to offload the slice to the cloud node, s_d^{Q} the waiting time in the queue before service, and $s_{i.j}^{K}$ the computing time. The amount of resource usage of each device and the available links are shown by$U_T = \{u_{d1.t}.u_{d2.t}.\ldots.u_{D_n.t}.u_{Ln.t}\}$, where $u_{D_n.t}$indicates the amount of use of the Nth server from device D, and $u_{Ln.t}$ indicates the used capacity of the fronhaul links.

The objective of the system is twofold: 1) to maximize the number of served slices, and 2) to maximize resource utilization.The optimization problem can be expressed as equations 7 and 8 in the article [22].

IV. PROPOSED METHODE

In this section, we present a model-free approach known as Q-learning, which is used to find an optimal policy for the Markov decision process (MDP) model. We further discuss how this method can beimproved by approximate reinforcement learning for RA.

A. Q-learning algorithm

The Q-learning algorithm is an efficient machine-learning technique that follows the MDP model and uses dynamic learning via the train-validation method to find the optimum policy〖 π〗^*. The MDP model defines sample paths in which state-action pairs are represented by a tuple (S,A,p,r), consisting of possible states S, possible actions A, transition probability p from state S to s', and the reward r obtained after executing an action. The reward is calculated as the sum of bandwidth G.BW, resource utility j.RE, and quality of experience e.QoE:

In [20], the matrix $\mathbf{F}_e^H$ has been used to transform element space to beamspace. The UCA steering vector of (4) is mapped to the VULA array through this beamformer, which is defined by

$$R = G, BW + j, \mathrm{RE} + e, QoE \tag{1}$$

For this work, an agent is designed in the cloud controller that observes the environment and trains it to make decisions. However, this method can achieve the optimal strategy when environment dimensions are small and actions are discrete. The process of executing the Q algorithm follows Algorithm 1.To gain a more comprehensive understanding of the agent's functioning in the Q-learning algorithm, we suggest referring to the article[24].

ARL for RA

The wireless network environment is vast and full of random events. Using conventional Q-learning can cause the Q-value table to explode and lack convergence in the exploration and exploitation phases. To address this issue, we have developed a Q-learning algorithm and introduced the ARL algorithm for RA. The core idea of our proposed method is to obtain Q(s,a) from a linear combination of features:

$$\mathrm{Q}\ (\mathrm{s}.a; \mathrm{w}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{f_i}(\mathrm{s}.a)\mathrm{w_i} \tag{2}$$

We extract and define significant environment features as f, and w is the variable that updates the feature's weight with each action. The agent must learn the weights for the features extracted from the model states.

We define a feature $f(s.a)$over the state and action pairs that yield a vector $(f_0(s.a), f_1(s.a), \dots, f_i(s.a) \dots, \ \ f_n(s.a))$ of feature values. The weights are updated according to the following rules:

$Varied\ case$: $(s.a.s'.r)$

Algorithm 2, ARL for RA:

```
    1. Define the significant environmental features and their weights:
      - The feature function f(s, a) outputs a vector of feature values
(f_1, f_2... f_n) for a given state-action pair.
      - The weight vector w updates with each action taken by the agent.
    2. Initialize the Q-value table arbitrarily.
    3. Set the learning rate $\alpha$ and discounting factor $\gamma$ in the range [0, 1].
    4. For each episode, repeat:
      - Choose an action a with probability $\varepsilon$.
      - Observe the new state s' and reward r.
      - Compute the difference between the expected and actual
rewards:
        difference = r + $\gamma$ * max(Q(s', a')) - Q(s, a)
      - Update the weight vector for each feature:
        w_i ← w_i + $\alpha$ * difference * f_i(s, a)
      - Update the Q-value for the current state-action pair:
    Q(s, a) ← Q(s, a) + $\alpha$ * difference
    5. Update the policy based on the new Q-value table.
```

The ARL algorithm for RA addresses the exploding Q-value table and lack of convergence issues associated with conventional Q-learning methods in wireless networks. By using a linear combination of significant features extracted from the environment, the proposed method reduces the dimensionality of the problem and enhances the agent's learning efficiency.

$$\text{disagreement } = \left[\mathrm{r} + \gamma \ \max_{\mathrm{a}'} \mathrm{Q}\,(\mathrm{s}'.\mathrm{a}') =\right] - \mathrm{Q}(\mathrm{s.a})$$

$$\mathrm{w_i} \leftarrow \mathrm{w_i} \ + \alpha\,, [\text{disagreement}] \ \times \ \mathrm{f_i}(\mathrm{s.a})$$

$$\mathrm{w_i} \leftarrow \mathrm{w_i} \ + \ \alpha\,,\left(\mathrm{r} + \gamma \ \max_{\mathrm{a}'} \mathrm{Q}\,(\mathrm{s}'.\mathrm{a}') - \ \mathrm{Q}(\mathrm{s.a})\right) \times \ \mathrm{f_i}(\mathrm{s.a}) \tag{3}$$

Algorithm 2 outlines the operation of our proposed method. To use ARL for RA, we must first identify the associated features and weights. For this purpose, we measure the capacities of computing resourcesand FHL transfer to containers and number their values according to standardization. Additionally, one counter-deployment into each container indicates the amount of used and remaining resources.Based on these measures, we define four suitable features for our work:

- (f_1) Computational resource capacity: This feature represents the total capacity of the computational system, and if it is higher, it provides more processing power for the system.
- (f_2) Data transfer rate: The data transfer rate between cloud computing and edge computing systems can also be extracted as an environmental feature.

- (f_3) Satisfaction: This feature indicates how satisfying the system is for accepting user requests and needs. It can be measured by different satisfaction metrics like Quality of Service (QoS).

Table 2: Classification of use cases

Applications	*Bandwidth*	*Latency*	*Executor nodes*
Service type 1	≤ 5 Mbps	≤ 5 MS	Fog nodes
Service type 2	3-7 Mbps	≤ 10 MS	Fog nodes
Service type 3	≤ 10 Mbps	50 ms-1s	Fog nodes & CO
Service type 4	15~ Mbps	≤ 5 s	Fog nodes & CO
Service type 5	≤ 100 Mbps	≤ 5 s	Fog nodes & CO

- (f_4) User demand level: The level of user demand for using the system and processing their data can be considered as a significant environmental feature. It can be measured by the number of
- requests or amount of data that users need to process. Therefore, Initial weights are given for the 4 features that are:

$$Q(s.a) = w_1 f_1(s.a) + w_2 f_2(s.a) + w_3 f_3(s.a) + w_4 f_4(s.a) \quad (4)$$

V. SIMULATIONANDNUMERICALANALYSES

This section discusses the evaluation of an ARL-RA-based approach using PyTorch, an open-source ML framework that accelerates the transition from research prototyping to production deployment. Libraries such as TensorFlow and NumPy were also utilized. The convergence rates of ARL and DRL were compared using two F-RANs (with 2.60 GHz power computations) and one CO (with 3.6 GHz power computations), each with five different types of slices with a bandwidth of 40-MegaHertz. Table 2 provides the classification of the use cases based on service type, bandwidth, latency, and executor nodes.In Fig. 2, the reward variation relative to the epoch index is illustrated, showing that regardless of the value of α, the ARL converged after 200-400 epochs while the DRL had not yet shown signs of convergence even at 1000 epochs. Therefore, it can be concluded that the ARL converges more quickly than the DRL. Additionally, the performance of the equal-allocation policy was assessed and compared to the resource allocation based on the number of slices, demonstrating that ARL is executed more efficiently than the alike-allocation policy.In Fig 2, an illustration of the reward variation concerning the epoch index is presented. This function is defined by equation (14). Notably, regardless of the value assigned to α, the ARL may converge after 200-400 epochs. In contrast, the DRL displays no sign of convergence until 1000 epochs have passed. Hence, it can be safely inferred that the ARL converges more rapidly than the DRL.Furthermore, Fig. 2 presents the performance of the equal allocation policy, where resources are allocated based on the number of slices. The results indicate that the ARL outperforms the equal allocation policy in terms of efficiency.In our scenario, the usage of CO resources depends on the accepted slice in the fog resources, given the nature of the services listed in Table 2. Specifically, if the controller only accepts slices of type 1 and 2 that need to be processed at the edge, it may result in cloud and communication

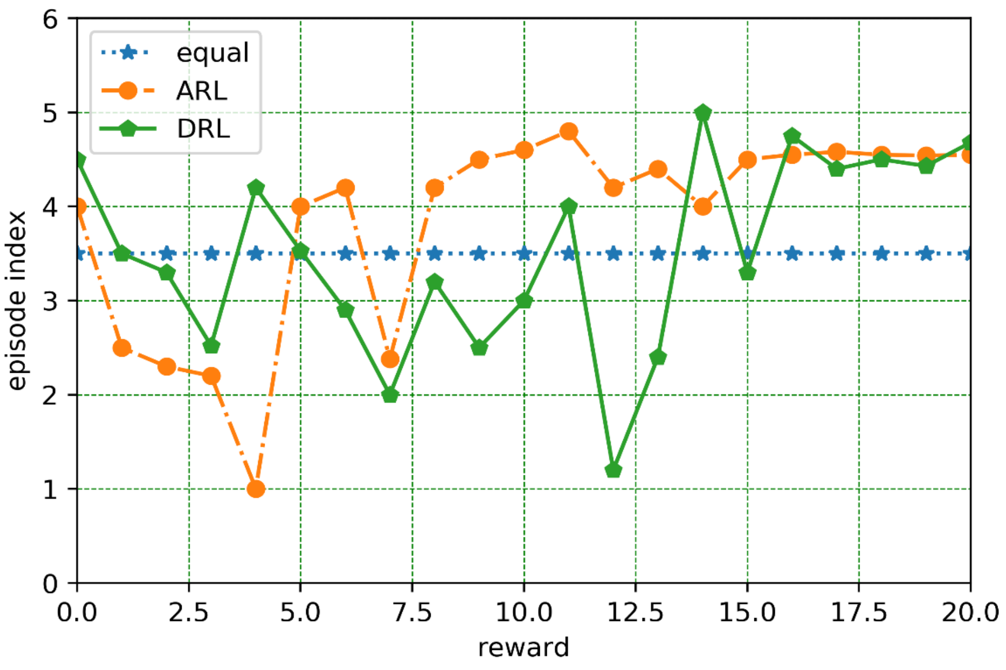

Fig. 2. The convergence rate of the ARL-RA and DRL

resources being wasted and remaining idle. Additionally, the maximum number of users who can be served in the best possible case is limited to 25-30, depending on the number of available resources. As mentioned before, CPU processing is performed at the edge nodes, while the fronthaul link is used for communication among F-RAN and CO. It is essential to ensure that the number of resources used is proportional to different resource types. Fig.3 depicts the usage rate of edge resources. If resource management is random and without a strategy, the number of resources used will increase exponentially with an increase in the number of users. For instance, accepting 16 users results in 100 of their resources being consumed in this field. However, the proposed ARL-RA method for serving 20 users has used approximately 83% of the resources, indicating that our method accounts for the overall network's resource consumption. The ARL-RA method not only ensures efficient resource utilization but also increases user satisfaction by optimizing the allocation of resources based on their demand. The proposed approach achieves this by using ARL algorithms that dynamically update the weights of feathers to adapt to future demands, ensuring that the available resources are utilized optimally while meeting the users' requirements. This dynamic optimization is particularly important in scenarios where the user population varies over time, making it challenging to allocate resources adequately.Moreover, the ARL-RA method takes into account the entire network's resource consumption, including both edge and central office resources, and ensures that the usage rate of these resources is proportional to their availability. By doing so, the proposed approach reduces waste and idle time, leading to better resource utilization and a more efficient network.

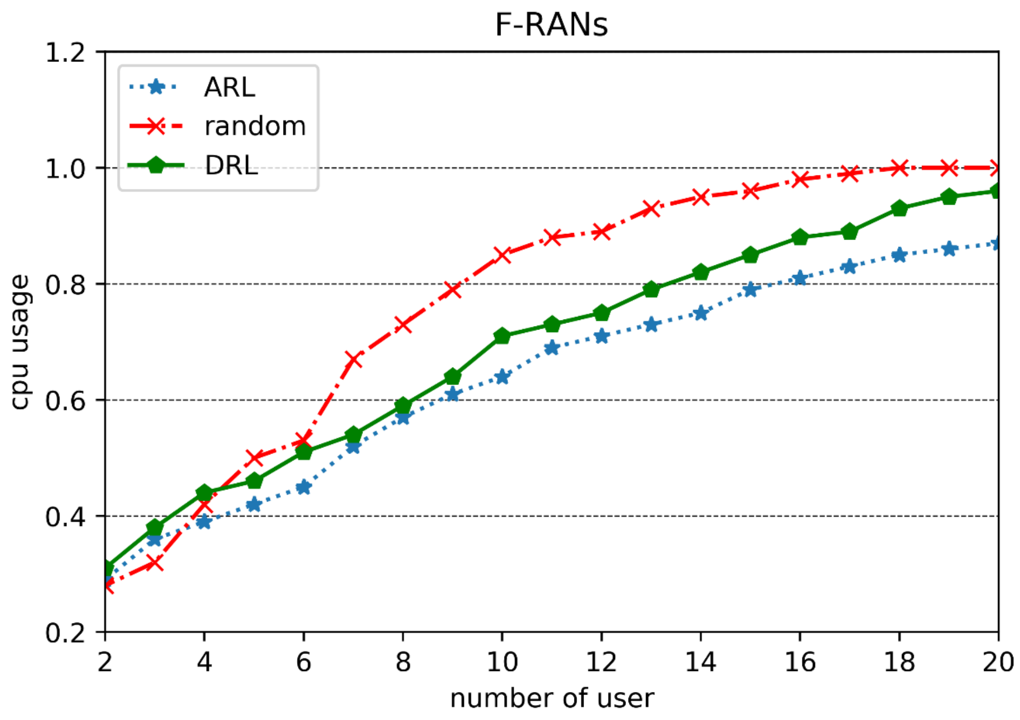

Fig. 3. Resource utilization.

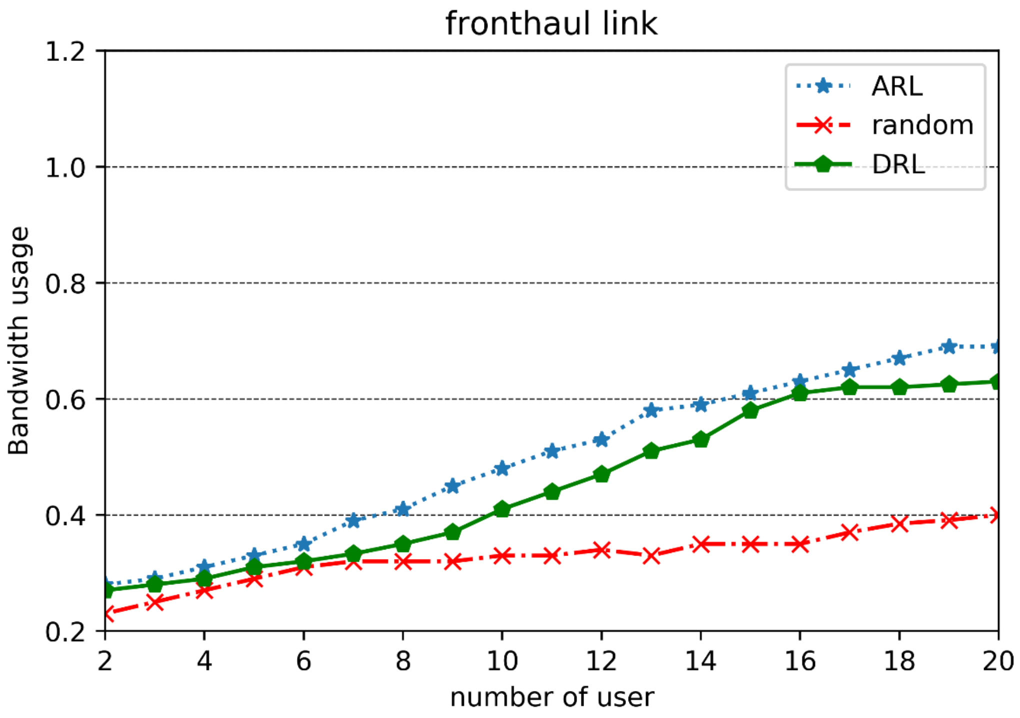

Fig. 4. Resource utilization.

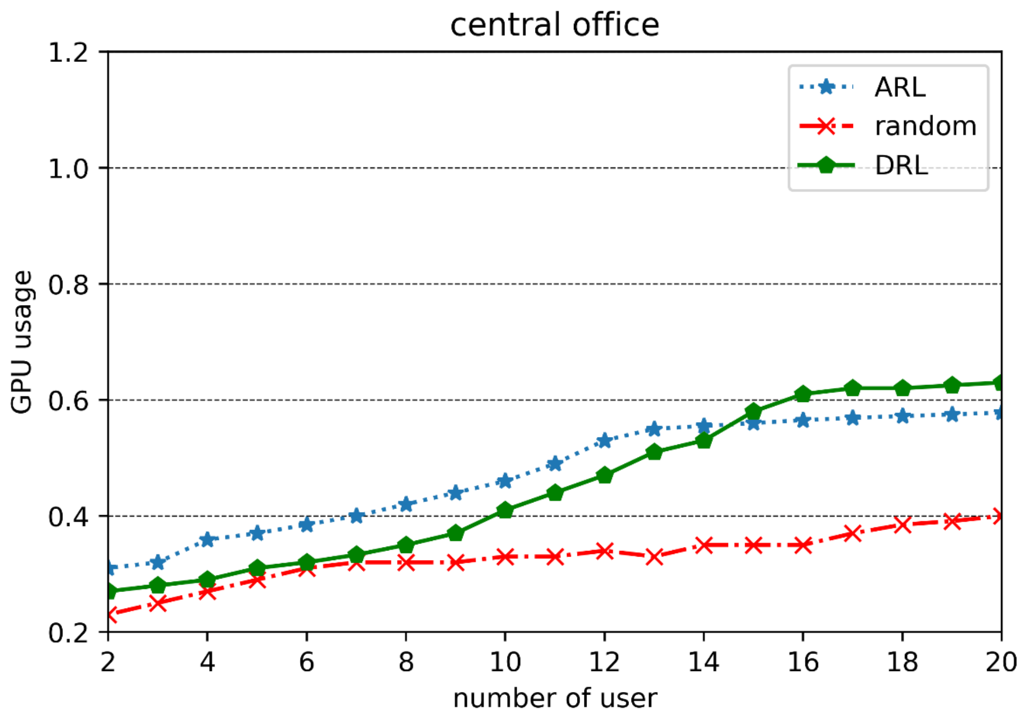

Fig. 5. Resource utilization.

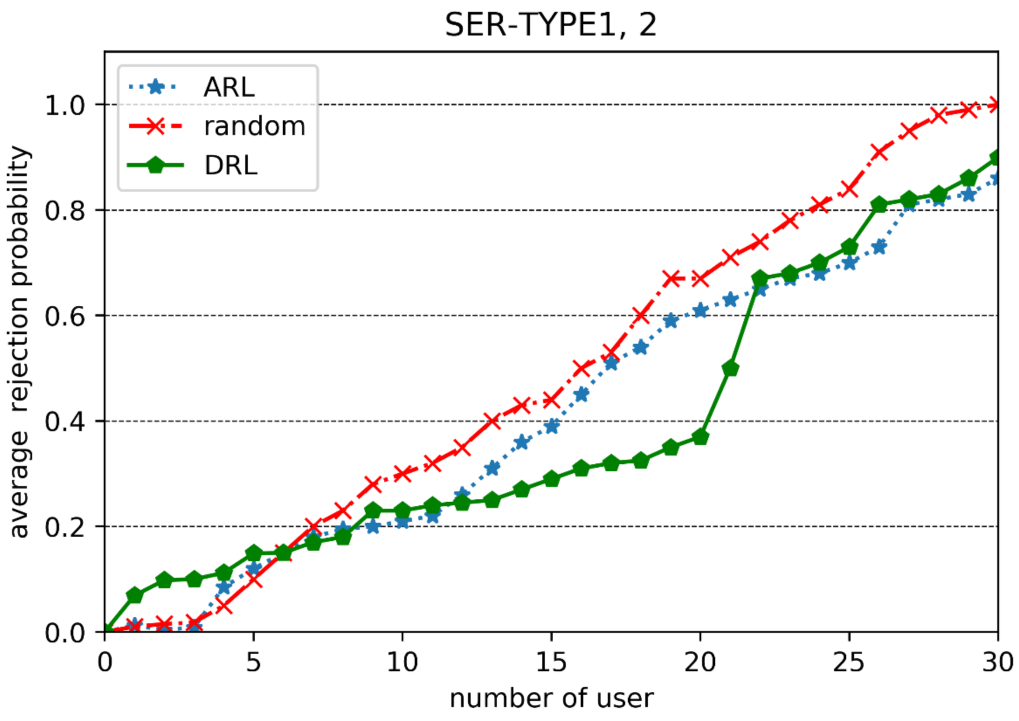

Fig. 6. The possibility of rejecting users' requests.

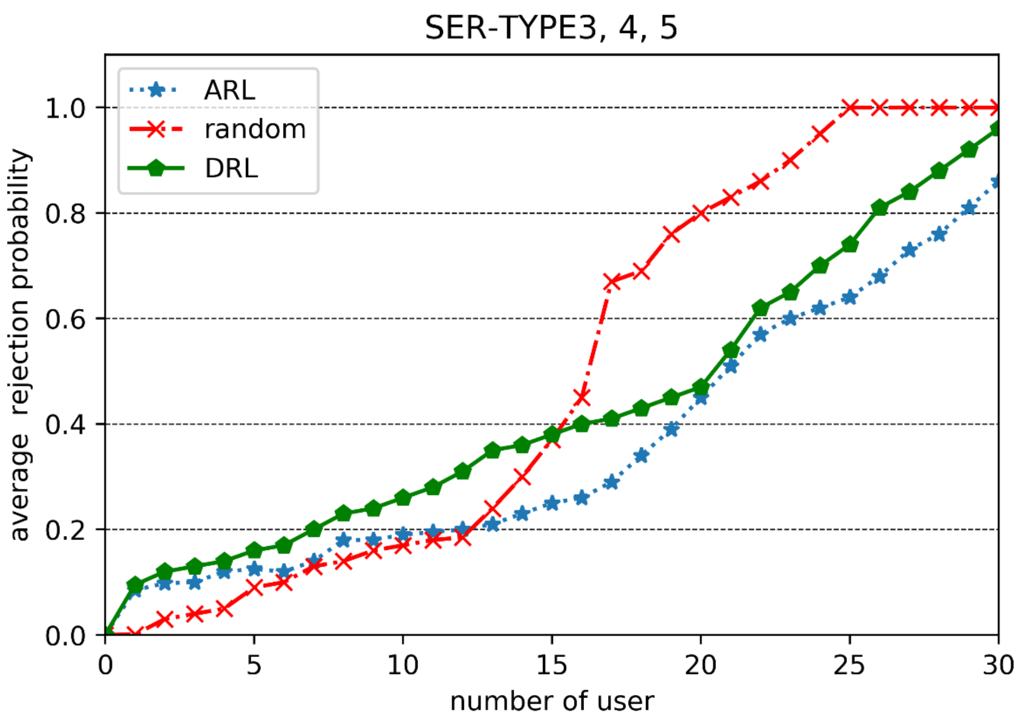

Fig. 7. The possibility of rejecting users' requests.

Overall, the ARL-RA method outperforms random resource management in terms of resource utilization, user satisfaction, and network efficiency, making it a promising approach for managing fog and cloud resources in a dynamic network environment. Our assertion is supported by the data presented in Figs. 4 and 5, which indicate that the proposed method results in significantly lower traffic volume directed toward the CO compared to both the random and DRL methods. We have conducted an assessment of the likelihood of user rejection based on the type of service they receive. Our findings, as demonstrated in Figs. 6 and 7, indicate that the acceptance probability of users using the random method is contingent on their access to resources. Each user who accesses these resources creates a slice without considering the distribution of resource consumption in other areas, resulting in an increased possibility of rejecting delay-sensitive services with an increasing number of applicants. In contrast, intelligent methods exhibit more intelligent behavior than the random approach.

VI. CONCLUTION

This study presents a novel approach for intelligent resource allocation in 5G mobile networks using Network Slicing (NS) technology and Fog-radio access networks (F-RAN). The proposed method, named Approximate Reinforcement Learning (ARL), leverages reinforcement learning techniques to extract and evaluate critical network features, outperforming conventional Q-learning algorithms. ARL-RA exhibits promising performance in terms of convergence, resource utilization, and the ability to handle users' request fields efficiently.

Furthermore, this research contributes to the field of academics and practices by offering insights into intelligent resource allocation techniques in 5G networks. The utilization of Network Slicing and Fog-RAN optimizes resource allocation, improves network efficiency, and enhances user experience. The findings of this study can serve as a foundation for future research in this area.

Moving forward, it is recommended to explore the integration of multiple objectives in the resource allocation process to cater to diverse user demands. This can involve considering factors such as latency, energy efficiency, and quality of service requirements. Additionally, the scalability of the proposed approach should be investigated to ensure its effectiveness in large-scale networks with a significant number of users. Moreover, assessing the impact of various network conditions on the performance of the proposed approach would further enhance its applicability in different scenarios.

REFERENCES

[1] R. Atat, L. Liu, H. Chen, J. Wu, H. Li, and Y. Yi, "Enabling cyber-physical communication in 5G cellular networks: challenges, spatial spectrum sensing, and cyber-security," *IET Cyber-Physical Systems: Theory & Applications,* vol. 2, no. 1, pp. 49-54, Apr. 2017.

[2] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Computer Networks,* vol. 167, no. 7, p. 106984, Feb. 2020.

[3] Y. D. Ahmad Sarlak "An Approach to Improve the Quality of Service in DTN and Non-DTN based VANET," *Journal of Information Systems and Telecommunication,* no. Issue 4, pp. 240 - 248, Jan. 2020.

[4] H. Xiang, W. Zhou, M. Daneshmand, and M. Peng, "Network slicing in fog radio access networks: Issues and challenges," *IEEE Communications Magazine,* vol. 55, no. 12, pp. 110-116, Dec. 2017.

[5] M. Khazaei, "Dynamic Tree- Based Routing: Applied in Wireless Sensor Network and IOT," *Journal of Information Systems and Telecommunication,* vol. Vol.10, No.3, pp. 191-200, Aug. 2022.

[6] J. Kaur, M. A. Khan, M. Iftikhar, M. Imran, and Q. E. U. Haq, "Machine learning techniques for 5G and beyond," *IEEE Access,* vol. 9, no. 3, pp. 23472-23488, Jan. 2021.

[7] M. E. Morocho-Cayamcela, H. Lee, and W. Lim, "Machine learning for 5G/B5G mobile and wireless communications: Potential, limitations, and future directions," *IEEE access,* vol. 7, no. 7. pp. 137184-137206, Sep. 2019.

[8] H. F. Sajad Mohammadzadeh , Zohreh Dorrani, "Edge Detection and Identification using Deep Learning to Identify Vehicles," *Journal of Information Systems and Telecommunication (JIST),* vol. Vol.10, No.3, pp. 201-210, Aug. 2022.

[9] J. Clifton and E. Laber, "Q-learning: Theory and applications," *Annual Review of Statistics and Its Application,* vol. 7, no. 3, pp. 279-301, Mar. 2020.

[10] B. Han and H. D. Schotten, "Machine learning for network slicing resource management: A comprehensive survey," *arXiv preprint arXiv:2001.07974*, Jan .2020.

[11] M. S. Al-Abiad, M. Z. Hassan, and M. J. Hossain, "A joint reinforcement-learning enabled caching and cross-layer network code in F-RAN with D2D communications," *IEEE Transactions on Communications,* vol. 70, no. 7, pp. 4400-4416, Apr. 2022.

[12] A. Nassar and Y. Yilmaz, "Reinforcement learning for adaptive resource allocation in fog RAN for IoT with heterogeneous latency requirements," *IEEE Access,* vol. 7, no. 2, pp. 128014-128025, Sep. 2019.

[13] N. N. Khumalo, O. O. Oyerinde, and L. Mfupe, "Reinforcement learning-based resource management model for fog radio access network architectures in 5G," *IEEE Access,* vol. 9,no. 4, pp. 12706-12716, Jan. 2021.

[14] R. Aghazadeh, A. Shahidinejad, and M. Ghobaei-Arani, "Proactive content caching in edge computing environment: A review," *Software: Practice and Experience,* vol. 53, no. 3, pp. 811-855, Mar. 2023.

[15] Z. Cheng, M. Min, M. Liwang, L. Huang, and Z. Gao, "Multiagent DDPG-based joint task partitioning and power control in Fog computing networks," *IEEE Internet of Things Journal,* vol. 9, no. 1, pp. 104-116, June 2021.

[16] A. Nassar and Y. Yilmaz, "Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities," *IEEE Internet of Things Journal,* vol. 9, no. 1, pp. 222-235, 2021.

[17] Y. Zhou, M. Peng, S. Yan, and Y. Sun, "Deep reinforcement learning based coded caching scheme in fog radio access networks," in *2018 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, Aug. 2018: IEEE, pp. 309-313.

[18] G. S. Rahman, M. Peng, S. Yan, and T. Dang, "Learning based joint cache and power allocation in fog radio access networks," *IEEE Transactions on Vehicular Technology,* vol. 69, no. 4, pp. 4401-4411, Feb. 2020.

[19] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," *Future Generation Computer Systems,* vol. 110, no. 8, pp. 1098-1115, Sep. 2020.

[20] A. O. Abdalrahman, D. Pilevarzadeh, S. Ghafouri, and A. Ghaffari, "The Application of Hybrid Krill Herd Artificial Hummingbird Algorithm for Scientific Workflow Scheduling in Fog Computing," *Journal of Bionic Engineering,* vol. 20, no. 5, pp. 2443-2464, May 2023.

[21] M. Nematollahi, A. Ghaffari, and A. Mirzaei, "Task and resource allocation in the internet of things based on an improved version of the moth-flame optimization algorithm," *Cluster Computing,* Vol. 10, no. 5, pp. 1-23, June 2023.

[22] M. Khani, S. Jamali, and M. K. Sohrabi, "Approximate Q-learning-based (AQL) network slicing in mobile edge-cloud for delay-sensitive services," *The Journal of Supercomputing,* 2023/09/09 2023, doi: 10.1007/s11227-023-05614-4.

[23] C. E. Nelson *et al.*, "In vivo genome editing improves muscle function in a mouse model of Duchenne muscular dystrophy," *Science,* vol. 351, no. 6271, pp. 403-407, Jan. 2016.

[24] M. Khani, S. Jamali, and M. K. Sohrabi, "An Enhanced Deep Reinforcement Learning-based Slice Acceptance Control System (EDRL-SACS) for Cloud-Radio Access Network," *Physical Communication,* p. 102188, Sep. 2023.