IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 8, 2019, accepted August 26, 2019, date of publication September 5, 2019, date of current version September 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939735

Reinforcement Learning for Adaptive Resource Allocation in Fog RAN for IoT With Heterogeneous Latency Requirements

ALMUTHANNA NASSAR AND YASIN YILMAZ, (Member, IEEE)
Electrical Engineering Department, University of South Florida, Tampa, FL 33620, USA
Corresponding author: Almuthanna Nassar (atnassar@mail.usf.edu)

This work was supported in part by NSF award number 1737598.

ABSTRACT In light of the quick proliferation of Internet of things (IoT) devices and applications, fog radio access network (Fog-RAN) has been recently proposed for fifth generation (5G) wireless communications to assure the requirements of ultra-reliable low-latency communication (URLLC) for the IoT applications which cannot accommodate large delays. To this end, fog nodes (FNs) are equipped with computing, signal processing and storage capabilities to extend the inherent operations and services of the cloud to the edge. We consider the problem of sequentially allocating the FN's limited resources to IoT applications of heterogeneous latency requirements. For each access request from an IoT user, the FN needs to decide whether to serve it locally at the edge utilizing its own resources or to refer it to the cloud to conserve its valuable resources for future users of potentially higher utility to the system (i.e., lower latency requirement). We formulate the Fog-RAN resource allocation problem in the form of a Markov decision process (MDP), and employ several reinforcement learning (RL) methods, namely Q-learning, SARSA, Expected SARSA, and Monte Carlo, for solving the MDP problem by learning the optimum decision-making policies. We verify the performance and adaptivity of the RL methods and compare it with the performance of the network slicing approach with various slicing thresholds. Extensive simulation results considering 19 IoT environments of heterogeneous latency requirements corroborate that RL methods always achieve the best possible performance regardless of the IoT environment.

INDEX TERMS Resource allocation, fog RAN, 5G cellular networks, low-latency communications, IoT, Markov decision process, reinforcement learning.

I. INTRODUCTION

There is an ever-growing demand for wireless communication technologies due to several reasons such as the increasing popularity of Internet of Things (IoT) devices, the widespread use of social networking platforms, the proliferation of mobile applications, and the current lifestyle that has become highly dependent on technology in all aspects. It is expected that the number of connected devices worldwide will reach three times the global population in 2022 with 3.6 devices per capita. However, in some regions, such as North America, the number of connected devices is projected to reach about 13.4 devices per capita by 2022, which makes the massive IoT a very common concept. This trend of massive IoT will generate an annual global IP traffic of 4.8 zettabytes by 2022, which corresponds to 4-times the traffic in 2016 and 184-times the traffic in 2005, in which wireless and mobile devices will account for 71% of this forecast [1]. This unprecedented demand for mobile data services makes it unbearable for service providers with the current third generation (3G) and fourth generation (4G) networks to keep pace with it [2]. The design criteria for fifth generation (5G) wireless communication systems will include providing ultra-low latency, wider coverage, reduced energy usage, increased spectral efficiency, more connected devices, improved availability, and very high data rates of multi giga-bit-per-second (Gbps) everywhere in the network including cell edges [3]. Several radio frequency (RF) coverage and capacity solutions are proposed to fulfill the goals of 5G including, beamforming, carrier aggregation, higher order modulation, and dense deployment of small cells [4]. Millimeter-wave (mm-wave) frequency range is likely to be

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh.

utilized in 5G because of the spacious bandwidths available in these frequencies for cellular services [5]. Massive multi-input-multi-output (MIMO) is potentially involved for excellent spectral efficiency and superior energy efficiency [6].

To cope with the growing number of IoT devices and the increasing amount of traffic for better user satisfaction, cloud radio access network (C-RAN) architecture is suggested for 5G, in which a powerful cloud controller (CC) with pool of baseband units (BBU) and storage pool supports large number of distributed remote radio units (RRU) through high capacity fronthaul links [7], [8]. The C-RAN is characterized by being clean as it reduces energy consumption and improves the spectral efficiency due to the centralized processing and collaborative radio [9]. However, in light of the massive IoT applications and the corresponding generated traffic, C-RAN structure places a huge burden on the centralized CC and its fronthaul, which causes more delay due to limited fronthaul capacity and busy cloud servers in addition to the large transmission delays [10], [11].

A. F-RAN AND HETEROGENEOUS IoT

The latency issue in C-RAN becomes critical for IoT applications that cannot tolerate such delays. And that is the reason fog radio access network (F-RAN) is introduced for 5G, where fog nodes (FN) are not only limited to perform RF functionalities but also empowered with caching, signal processing and computing resources [12], [13]. This makes FNs capable of independently delivering network functionalities to end users at the edge without referring them to the cloud to tackle the low-latency needs.

IoT applications have various latency requirements. Some applications are more delay-sensitive than others, while some can tolerate larger delays [14]–[16]. Hence, especially in a heterogeneous IoT environment with various latency needs, FN must allocate its limited and valuable resources in a smart way. In this work, we present a novel framework for resource allocation in F-RAN for 5G by employing reinforcement learning methods to guarantee the efficient utilization of limited FN resources while satisfying the low-latency requirements of IoT applications.

B. LITERATURE REVIEW

For the last several years, 5G and IoT related topics have been of great interest to many researchers in the wireless communications field. Recently, a good number of works in the literature focused on achieving low latency for IoT applications in 5G F-RAN. For instance, resource allocation based on cooperative edge computing has been studied in [17]–[21] for achieving ultra-low latency in F-RAN. The work in [17] proposed a mesh paradigm for edge computing, where the decision-making tasks are distributed among edge devices instead of utilizing the cloud server. The authors in [18], [21] considered heterogeneous F-RAN structures including, small cells and macro base stations, and provided an algorithm for selecting the F-RAN nodes to serve with proper heterogeneous resource allocation. The number of F-RAN nodes and their locations have been investigated by [22]. Content fetching is used in [7], [19] to maximize the delivery rate when the requested content is available in the cache of fog access points. In [23], cloud predicts users' mobility patterns and determines the required resources for the requested contents by users, which are stored at cloud and small cells. The work in [20] addressed the issue of load balancing in fog computing and used fog clustering to improve user's quality of experience. The congestion problem, when resource allocation is done based on the best signal quality received by the end user, is highlighted in [24], [25]. The work in [24] provided a solution to balance the resource allocation among remote radio heads by achieving an optimal downlink sum-rate, while [25] offered an optimal solution based on reinforcement learning to balance the load among evolved nodes for the arrival of machine-type communication devices. To reduce latency, soft resource reservation mechanism is proposed in [26] for uplink scheduling. The authors of [27] presented an algorithm that works with the smooth handover scheme and suggested scheduling policies to ease the user mobility challenge and reduce the application response time. Radio resource allocation strategies to optimize spectral efficiency and energy efficiency while maintaining a low latency in F-RAN are proposed in [28]. With regard to learning for IoT, [29] provided a comprehensive study about the advantages, limitations, applications, and key results relating to machine learning, sequential learning, and reinforcement learning. Multi-agent reinforcement learning was exploited in [30] to maximize network resource utilization in heterogeneous networks by selecting the radio access technology and allocating resources for individual users. The model-free reinforcement learning approach is used in [31] to learn the optimal policy for user scheduling in heterogeneous networks to maximize the network energy efficiency. Resource allocation in non-orthogonal-multiple-access based F-RAN architecture with selective interference cancellation is investigated in [32] to maximize the spectral efficiency while considering the co-channel interference. With the help of task scheduler, resource selector, and history analyzer, [33] introduced an FN resource selection algorithm in which the selection and allocation of the best FN to execute an IoT task depends on the predicted run-time, where stored execution logs for historical performance data of FNs provide realistic estimation of it.

A comprehensive study of network slicing in 5G system is considered in [34], [35]. Issues and challenges of network slicing in Fog RAN is investigated in [34], where authors presented key techniques and solutions in regards to radio and cache resource management as well as social-aware slicing. [35] provides a comprehensive survey on network slicing which embraces the key principles, enabling technologies, challenges, standardization, and solutions including slicing solutions for 5G system, and illustrates the requirements and diverse use cases of network slicing considering RAN sharing, end-to-end orchestration and management involving the radio access, transport and core networks. Radio resource

allocation for different network slices is exploited in [36]–[38] to support various quality-of-service (QoS) requirements and minimize the queuing delay for low latency requests, in which network is logically partitioned into a high-transmission-rate slice for mobile broadband (MBB) applications, and a low-latency slice which supports ultra-reliable low-latency communication (URLLC) applications. The authors in [38] proposed a hierarchical radio resource allocation architecture for network slicing in which a global radio resource manager (GRRM) allocates subchannels to local radio resource managers (LRRMs) in slices, which then assign resources to their end users. However, the resource allocation problem considered in the network slicing literature focuses on the dynamics of resource allocation among various network slices and layers, i.e., decides on allocation of resources between FNs for URLLC applications and RRUs for MBB applications while it is infeasible for mobile operators and service providers to keep changing the distribution of resources in the network due to the huge accompanying operational expenditure (OPEX) to cover all required hardware, software and license swaps as well as the implementation of any necessary frequency reuse plans and fronthaul links capacity upgrade, and the impact of outages and prolonged testing, optimizing and fine tuning the network performance follow every single change. Hence, deciding on resource allocation among network slices should be so thoughtful and deliberate, and only after assuring that each slice utilizes its allocated resources efficiently. In this work, we zoom in to the allocated limited resources to FNs to optimize and guarantee their efficient utilization. We compare the performance and adaptivity of the RL methods to the performance of the utility filtering-based network slicing approach with various slicing thresholds.

C. CONTRIBUTIONS

With the motivation of satisfying the low-latency requirements of heterogeneous IoT applications through F-RAN, we provide a novel framework for allocating limited resources to users that guarantees efficient utilization of the FN's limited resources. In this work, we develop Markov Decision Process (MDP) formulation for the considered resource allocation problem and employ diverse reinforcement learning (RL) methods for learning optimum decision-making policies adaptive to the IoT environment. Specifically, in this paper we propose an MDP formulation for the considered F-RAN resource allocation problem, and investigate the use of various RL methods, Q-learning (QL), SARSA, Expected SARSA (E-SARSA), and Monte Carlo (MC), for learning the optimal fine-grained decision making policies of the MDP problem to improve efficiency. We also provide extensive simulation results in various IoT environments of heterogeneous latency requirements to evaluate the performance and adaptivity of the four RL methods and compare it to the performance of the network slicing approach with various slicing thresholds.

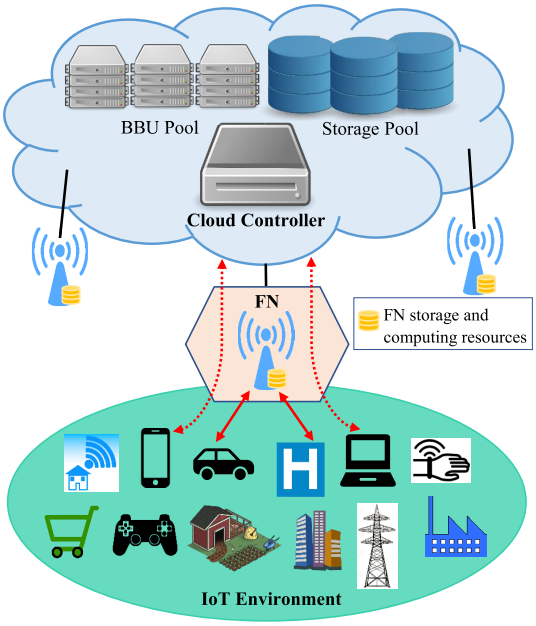

FIGURE 1. Fog-RAN system model. The FN serves heterogeneous latency needs in the IoT environment, and is connected to the cloud through the fronthaul links represented by solid lines. Solid red arrows represent local service by FN to satisfy low-latency requirements, and dashed arrows represent referral to the cloud to save FN's limited resources.

The remainder of the paper is organized as follows. Section II introduces the system model. The proposed MDP formulation for the resource allocation problem is given in Section III. Optimal policies and the related RL algorithms are discussed in Section IV. Simulation results are presented in Section V. Finally, we conclude the paper in Section VI. A list of notation and abbreviations used throughout the paper is provided in Table 1.

II. SYSTEM MODEL

We consider the F-RAN structure shown in Fig. 1, in which FNs are connected through the fronthaul to the cloud controller (CC), where a massive computing capability, centralized baseband units (BBUs) and cloud storage pooling are available. To ease the burden on the fronthaul and the cloud, and to overcome the challenge of the increasing number of IoT devices and low-latency applications, FNs are empowered with capability to deliver network functionalities at the edge. Hence, they are equipped with caching capacity, computing and signal processing capabilities. However, these resources are limited, and therefore need to be utilized efficiently. An end user attempts to access the network by sending a request to the nearest FN. The FN takes a decision whether to serve the user locally at the edge using its own computing and processing resources or refer it to the cloud. We consider the FN's computing and processing capacity to be limited to N resource blocks (RBs). User requests arrive sequentially and decisions are taken quickly, so no queuing occurs.

TABLE 1. Summary of notations and abbreviations.

Notation	Description
IoT	Internet of things
F-RAN	fog radio access network
C-RAN	cloud radio access network
5G	fifth generation
URLLC	ultra-reliable low-latency communication
FN	fog node
SNR	signal-to-noise ratio
MDP	Markov decision process
RL	reinforcement learning
ML	machine learning
QL	Q-learning
E-SARSA	Expected SARSA
MC	Monte Carlo
RF	radio frequency
MIMO	multi input multi output
CC	cloud controller
RRU	remote radio unit
BBU	baseband unit
QoS	quality of service
MBB	mobile broadband
RB	resource block
Thld	threshold
$\mathcal{S}$	set of states
ζ, β, κ	parameters for utility computation
$\mathcal{A}$	set of actions
θ	penalty for staying idle
M	number of served IoT requests in an episode
C	channel capacity in bps
B	frequency bandwidth in Hz
l	latency in milliseconds
ω	throughput requirement in bps
u	user utility
u_h	threshold for defining "high utility"
μ	achievable throughput ratio
ρ	probability of $u > 5$ in Iot environment
s_t	state at time t
s'	successor state
a_t	action taken at time t
a'	action from successor state
a^*	optimal action
r_t	reward received at time t
r_s	reward for serving
r_{sh}	reward for serving high-utility user
r_{sl}	reward for serving low-utility user
r_r	reward for rejecting
r_{rh}	reward for rejecting high-utility user
r_{rl}	reward for rejecting low-utility user
π	policy for taking actions
π^*	optimal policy
G_t	return from time t onward
$V(s)$	state-value function of s
$V_\pi(s)$	state-value function following policy π
$V^*(s)$	optimal state-value function
$Q(s,a)$	action-value function
$Q^*(s,a)$	optimal action-value function
$Q_\pi(s,a)$	action-value function following policy π
T	termination time
$P^a_{ss'}$	transition probability to s' given s, a
$R^a_{ss'}$	reward received for taking a from s given s'
N	total number of FN's resource blocks
$\mathbb{E}_u$	Expectation with respect to u
b_t	number of occupied RBs at time t
γ	discount factor
α	learning rate
ϵ	probability of random action
n	batch size
$\mathbb{Q}$	an array for updated $Q(s,a)$, for all (s,a)
τ	time whose Q estimate is being updated
$\mathcal{E}$	IoT environment
$\bar{u}$	mean value of utilities in Iot environment
σ	standard deviation
R	objective performance metric
u_m	utility of served IoT request

The QoS requirements of a wireless user are typically given by the latency requirement and throughput requirement. IoT applications have various levels of latency requirement, hence it is sensible for the FN to give higher priority for serving the low-latency applications. To differentiate between similar latency requirements we also consider the risk of failing to satisfy the throughput requirement. This risk is related to the ratio of the achievable throughput to the throughput requirement. The achievable throughput is characterized by the signal-to-noise ratio (SNR) through Shannon channel capacity. Shannon's fundamental limit on the capacity of a communications channel gives an upper bound for the achievable throughput, as a function of available bandwidth (B) in Hz and SNR in dB, $C = B\log_2(1 + \text{SNR})$. Hence, we define the utility of an IoT user request to be a function of latency requirement, l (in milliseconds), throughput requirement, ω (in bits per second), and channel capacity, C (in bits per second), i.e., $u = f(l, \omega, C)$. Since the utility should be inversely proportional to the latency requirement, and directly proportional to the achievable throughput ratio, $\mu = C/\omega$, we define utility as

$$u = \kappa(\mu^\zeta / l^\beta), \tag{1}$$

where $\kappa, \zeta, \beta > 0$ are mapping parameters. This provides a flexible model for utility. By selecting the parameters κ, ζ, β a desired range of u and importance levels for latency and throughput requirements can be obtained. Since F-RAN is intended for satisfying low-latency requirements, typically, more weight should be given to latency by choosing larger β values.

FNs should be smart to learn how to decide (serve/refer to the cloud) for each IoT request (i.e., how to allocate its limited resources), so as to achieve the conflicting objectives of maximizing the average total utility of served users over time and minimizing its idle (no-service) time. The system objective can be stated as a constrained optimization problem,

$$\max_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T} \mathbb{1}_{\{a_t=serve\}} u_t \text{ and } \min_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T} \mathbb{1}_{\{a_t=reject\}}$$
$$\text{subject to } \sum_{t=0}^{T} \mathbb{1}_{\{a_t=serve\}} = N, \tag{2}$$

where a_t denotes the action taken at time t (either serves the request locally or rejects it and refers to cloud), T denotes the termination time when all RBs are filled, N denotes the number of RBs, and $\mathbb{1}_{\{\cdot\}}$ is the indicator function taking value 1 if its argument is true and 0 if false. The goal is to find the optimum decision policy $\{a_0, a_1, \ldots, a_{T-1}\}$ for an IoT environment which randomly generates $\{u_t\}$. Note that the final decision is always $a_T = serve$ by definition, hence omitted in the policy representation.

One straightforward approach to deal with this resource allocation problem is to apply network slicing [34], [35] based on the user utility, in which the network is logically partitioned into two slices [36]–[38], a fog slice handles high-utility IoT requests of low-latency demand, and cloud slice considers low-utility users. Hence, a filtering standard is required for the FN to direct users' requests to their

corresponding network slices. For instance, we can define a threshold rule, such as "serve locally if $u > 5$", if we classify all applications in an IoT environment into ten different utilities $u \in \{1, 2, \ldots, 10\}$, 10 being the highest utility. However, such a policy is sub-optimum since the FN will be waiting for a user to satisfy the threshold, which will increase the FN's idle time and make the CC busier. The main drawback of this policy is that it cannot adapt to the dynamic IoT environment to achieve the objective. For instance, when the user utilities are almost uniformly distributed, a very selective policy with a high threshold will stay idle most of the time, whereas an impatient policy with a low threshold will in general obtain a low average served utility. A mild policy with threshold 5 may in general perform better than the extreme policies, yet it will not be able adapt to different IoT environments. A better solution for the F-RAN resource allocation problem is to use RL techniques which can continuously learn the environment and adapt the decision rule accordingly.

III. MDP PROBLEM FORMULATION

RL can be thought as the third paradigm of machine learning in addition to the other two paradigms, supervised learning and unsupervised learning. The key point in the proposed RL approach is that FN learns about the IoT environment by interaction and then adapts to it. FN gains rewards from the environment for every action it takes, and once the optimum policy of actions is learned, FN will be able to maximize its expected cumulative rewards, adapt to the IoT environment, and achieve the objective.

For an access request from a user with utility u_t, at time t, if the FN decides to take the action $a_t = serve$, which means to serve the user at the edge, then it will gain an immediate reward r_t and one of the RBs will be occupied. Otherwise, for the action $a_t = reject$, which means to reject serving the user at the edge and refer it to the cloud, the FN will maintain its available RBs and get a reward r_t. The value of r_t depends on a_t and u_t. For tractability, we consider quantized utility values, $u_t \in \{1, 2, \ldots, U\}$.

We define the state s_t of the FN at any time t as

$$s_t = 10\, b_t + u_t, \tag{3}$$

where $b_t \in \{0, 1, 2, \ldots, N\}$ is the number of occupied RBs at time t. Note that the successor state s_{t+1} depends only on the current state s_t, the utility u_{t+1} of the next service request, and the action taken (*serve* or *reject*), satisfying the Markov property $P(s_{t+1}|s_0, \ldots, s_{t-2}, s_{t-1}, s_t, a_t) = P(s_{t+1}|s_t, a_t)$, i.e., Markov state. Hence, we formulate the Fog-RAN resource allocation problem in the form of a Markov decision process (MDP), which is defined by the tuple $(\mathcal{S}, \mathcal{A}, P^a_{ss'}, R^a_{ss'})$, where $\mathcal{S}$ is the set of all possible states, i.e., $s_t \in \mathcal{S}$, $\mathcal{A}$ is the set of actions, i.e., $a_t \in \mathcal{A} = \{serve, reject\}$, $P^a_{ss'}$ is the transition probability from state s to s' when the action a is taken, i.e., $P^a_{ss'} = P(s'|s, a)$, where s' is a shorthand notation for the successor state, and $R^a_{ss'}$ is the immediate reward received when the action a is taken at state s which ends up in state s', e.g., $r_t = R^{a_t}_{s_t s_{t+1}} \in \mathcal{R}$. The return G_t is defined as the cumulative discounted rewards received from time t onward and given by

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{j=0}^{\infty} \gamma^j r_{t+j}, \tag{4}$$

where $\gamma \in [0, 1]$ is the discount factor. γ represents the weight of future rewards with respect to the immediate reward, $\gamma = 0$ ignores future rewards, whereas $\gamma = 1$ means that future rewards are of the same importance as the immediate rewards. The objective of the MDP problem is to maximize the expected initial return $\mathbb{E}[G_0]$.

In the presented MDP, for an FN that has a computing and processing capacity of N RBs, there are $U(N + 1)$ states, $s_t \in \mathcal{S} = \{1, 2, 3, \ldots, U(N + 1)\}$, where U is the greatest discrete utility level. At the initiation time $t = 0$, all RBs are available, i.e., $b = 0$, hence from (3), there are U possible initial states $s_0 \in \{1, 2, \ldots, U\}$ dependent on u_0. The MDP terminates at time T when all RBs are occupied, i.e., $b_T = N$, hence similarly there are U terminal states $s_T \in \{UN + 1, UN + 2, \ldots, U(N + 1)\}$. Note that a policy treating the MDP problem can continue operating after T as in-use RBs become available in time by taking actions similarly to its operation before T.

The reward mechanism $R^a_{ss'}$ is typically chosen by the system designer according to the objective. We propose a reward mechanism based on the received utility and the action taken for it. Specifically, at time t, based on u_t and a_t, the FN receives an immediate reward $r_t \in \mathcal{R} = \{r_{sh}, r_{sl}, r_{rh}, r_{rl}\}$, and moves to the successor state s_{t+1}, where r_{sh} is the reward for serving a high-utility request, r_{sl} is the reward for serving a low-utility request, r_{rh} is the reward for rejecting a high-utility request, and r_{rl} is the reward for rejecting a low-utility request. A request is determined as high-utility or low-utility relative to the environment based on a threshold u_h, which is a design parameter dependent on the utility distribution in IoT environment. For instance, u_h can be selected as a certain percentile, such as the 50$^{\text{th}}$ percentile, i.e., median, of the utilities in the environment. Hence, the proposed reward function is given by

$$r_t = \begin{cases} r_{sh}, & \text{if } a_t = serve, \quad u_t \geq u_h \\ r_{rh}, & \text{if } a_t = reject, \quad u_t \geq u_h \\ r_{sl}, & \text{if } a_t = serve, \quad u_t < u_h \\ r_{rl}, & \text{if } a_t = reject, \quad u_t < u_h. \end{cases} \tag{5}$$

Remark 1: Note that the threshold u_h does not have a definitive meaning with respect to the system requirements, i.e., there is no requirement saying that requests with utility lower/greater than u_h must be rejected/served. The goal here is to introduce an internal reward mechanism for the RL approach to facilitate learning the expected future gains, as will be clear later in this section and the following section. For an effective learning performance, the reward mechanism should be simple enough to guide the RL algorithm towards the system objective (see (2)) [39]. That is, its role is not to imitate the system objective closely to make the algorithm

TABLE 2. State transitions of 5-RB FN for a sample of IoT requests and random actions with $U = 10$, $u_h = 6$.

t	u_t	b_t	s_t	a_t	r_t	s_{t+1}
0	5	0	5	$reject$	r_{rl}	9
1	9	0	9	$serve$	r_{sh}	13
2	3	1	13	$reject$	r_{rl}	13
3	3	1	13	$serve$	r_{sl}	28
4	8	2	28	$serve$	r_{sh}	36
5	6	3	36	$reject$	r_{rh}	31
6	1	3	31	$reject$	r_{rl}	40
7	10	3	40	$serve$	r_{sh}	47
8	7	4	47	$reject$	r_{rh}	49
9	9	4	49	$serve$	r_{sh}	54
10	4	5	54	·	·	·

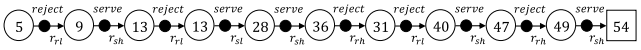

FIGURE 2. State transition graph for the MDP episode given in Table 2 for an FN with $N = 5$, $U = 10$, $u_h = 6$. Non-terminal states and terminal state are represented by circles and squares, respectively, and labeled by the states names. Filled circles represent actions, and arrows show the transitions with corresponding rewards.

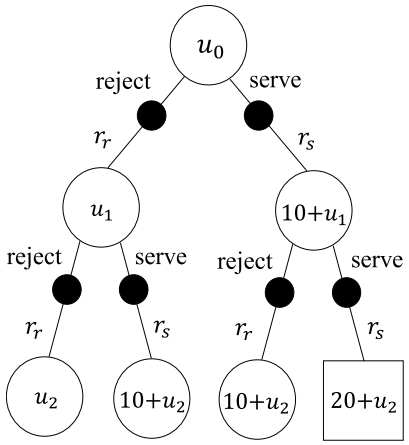

FIGURE 3. The first 3 levels of the backup diagram for the MDP with 2-RB FN ($N = 2$). Non-terminal states and terminal states are represented by open circles and squares, respectively, and labeled by the states according to (3). $r_s \in \{r_{sh}, r_{sl}\}$ and $r_r \in \{r_{rh}, r_{rl}\}$ are the rewards for serving and rejecting, respectively, and depend on u_h.

achieve it at once, but to resemble it in a simple manner to let the algorithm iteratively achieve a high performance.

Remark 2: Although a threshold u_h is utilized in the proposed reward mechanism, its use is fundamentally different than the utility filtering-based policy in network slicing approach which always accepts/rejects requests with utility greater/lower than a threshold. While the utility filtering-based policy considers only the immediate gain from the current utility, the algorithms tackling the MDP problem, such as the RL algorithms, consider the expected return $\mathbb{E}[G_0]$ which includes the immediate reward and expected future rewards. Hence, the threshold u_h does not necessarily cause the algorithm to accept/reject requests with utility greater/lower than u_h; it only plays an internal role in learning the expected future rewards.

State transitions for an FN with 5 RBs ($N = 5$), 10 utility levels ($U = 10$), and $u_h = 6$, a sample of IoT requests with utilities u_t, and random actions a_t are shown in Table 2. At time t, being at state s_t, and taking the action a_t will result in getting an immediate reward r_t and moving to the successor state s_{t+1}. The state transitions in Table 2 represent an episode of the MDP, it starts at $t = 0$ and terminates at $T = 10$ with the states $5 \rightarrow 9 \rightarrow 13 \rightarrow 13 \rightarrow 28 \rightarrow 36 \rightarrow 31 \rightarrow 40 \rightarrow 47 \rightarrow 49 \rightarrow 54$. The dynamics of this episode is shown through a state transition graph in Fig. 2, in which non-terminal states and terminal state are represented by circles and squares, respectively, and labeled by the states names, filled circles represent actions, and arrows show the transitions with corresponding rewards.

IV. OPTIMAL POLICIES

The state-value function $V(s)$, shown in (6), represents the long-term value of being in state s in terms of the expected return which can be collected starting from this state onward till termination. Hence, the terminal state has zero value since no reward can be collected from that state, and the value of initial state is equal to the objective function $\mathbb{E}[G_0]$. The state value can be viewed also in two parts: the immediate reward from the action taken and the discounted value of the successor state where we move to. Similarly, the action-value function $Q(s, a)$ is the expected return that can be achieved after taking the action a at state s, as shown in (7). The action value function tells how good it is to take a particular action at a given state. The expressions in (6) and (7) are known as the Bellman expectation equations for state value and action value, respectively [39],

$$V(s) = \mathbb{E}[G_t|s_t = s] = \mathbb{E}[r_t + \gamma V(s')|s], \quad (6)$$

$$Q(s, a) = \mathbb{E}[G_t|s, a] = \mathbb{E}[r_t + \gamma Q(s', a')|s, a], \quad (7)$$

where a' denotes the successor action at the successor state s'. Since (6) shows the relationship between the value of a state and its successor states, similarly for the value of an action in (7), it is useful to show the dynamics of the MDP in a backup diagram, as shown in Fig. 3 for a 2-RB FN ($N = 2$). The backup diagram provides an overview for the possible episodes of the considered MDP, where the minimum termination time required to reach a terminal state, at which all RBs are occupied ($b = N$), is $T = 2$ through the episode $u_0 \rightarrow 10 + u_1 \rightarrow 20 + u_2$, i.e., *serve* all the time. Note that early termination does not necessarily maximize the return.

The objective of the FN in the presented MDP is to utilize the N resource blocks for high-utility IoT applications in a timely manner. This can be done through maximizing the value of initial state, which is equal to the MDP objective $\mathbb{E}[G_0]$. To this end, an optimal decision policy is required, which is discussed next.

A policy π is a way of selecting actions. It can be defined as the set of probabilities of taking a particular action given the state, i.e., $\pi = \{P(a|s)\}$ for all possible state-action pairs. The policy π is said to be optimal if it maximizes

the value of all states, i.e., $\pi^* = \arg\max_{\pi} V_\pi(s), \forall s$. Hence, to solve the considered MDP problem, the FN needs to find the optimal policy through finding the optimal state-value function $V^*(s) = \max_{\pi} V_\pi(s)$, which is similar to finding the optimal action-value function $Q^*(s, a) = \max_{\pi} Q_\pi(s, a)$ for all state-action pairs. From (6) and (7), we can write the Bellman optimality equations for $V^*(s)$ and $Q^*(s, a)$ as,

$$V^*(s) = \max_{a\in\mathcal{A}} Q^*(s, a) = \max_{a\in\mathcal{A}} \mathbb{E}[r_t + \gamma V^*(s')|s, a], \quad (8)$$

$$Q^*(s, a) = \mathbb{E}[r_t + \gamma \max_{a'\in\mathcal{A}} Q^*(s', a')|s, a]. \quad (9)$$

The notion of optimal state-value function $V^*(s)$ greatly simplifies the search for optimal policy. Since the goal of maximizing the expected future rewards is already taken care of the optimal value of the successor state, $V^*(s')$ can be taken out of the expectation in (8). Hence, the optimal policy is given by the best local actions at each state. Dealing with $Q^*(s, a)$ to choose optimal actions is even easier, because with $Q^*(s, a)$ there is no need for the FN to do the one-step-ahead search and instead it picks the best action that maximizes $Q^*(s, a)$ at each state. Optimal actions are defined as follows,

$$a^* = \arg\max_{a\in\mathcal{A}} Q^*(s, a) = \arg\max_{a\in\mathcal{A}} \mathbb{E}[r_t|s, a] + \gamma V^*(s'|s, a). \quad (10)$$

After discretizing the utility into U levels, the state space becomes tractable with cardinality $|\mathcal{S}| = U(N + 1)$, hence in this case the optimal policy can be learned by estimating the optimal value functions (either (8) or (9)) using tabular methods such as model-free RL methods (e.g., Monte Carlo, SARSA, Expected SARSA, and Q-learning), which are also called approximate dynamic programming methods [39]. Since the expectations involved in value functions are not tractable to find in closed form, we resort to model-free RL methods in this work instead of exact dynamic programming. Continuous utility values (see (1)) would yield infinite dimensional state space, and thus require function approximation methods, such as deep Q-learning known as DQN [40], for predicting the value function at different states, which we leave to a future work.

In our MDP problem, firstly FN receives a request from an IoT application of utility u, then it makes a decision to *serve* or *reject*, meaning that the reward for serving $r_s \in \{r_{sh}, r_{sl}\}$ and the reward for rejecting $r_r \in \{r_{rh}, r_{rl}\}$ are known at the time of decision making. Thus, from (6) and (10), the optimal action at state s is given by

$$a^* = \begin{cases} serve, & \text{if } r_s + \gamma\mathbb{E}_u[V^*(s'_{serve} = 10(b+1) + u_{t+1})] \\ & \quad > r_r + \gamma\mathbb{E}_u[V^*(s'_{reject} = 10b + u_{t+1})], \\ reject, & \text{otherwise}, \end{cases} \quad (11)$$

where s'_{serve} is the successor state when $a = serve$, s'_{reject} is the successor state when $a = reject$, and $\mathbb{E}_u$ is the expectation with respect to the utilities u in the IoT environment.

Algorithm 1 Learning Optimum Policy Using Monte Carlo

1: Select: $\gamma \in [0, 1]$, $\{u_h, r_{sh}, r_{sl}, r_{rh}, r_{rl}\} \in \mathbb{R}$;
2: Input: N (number of RBs);
3: Initialize: $V(s) \leftarrow 0, \forall s$; *Returns(s)* (an array to save states' returns in all iterations);
4: **for** *iteration* $= 0, 1, 2, \ldots$ **do**
5: Initialize: $b \leftarrow 0$;
6: Generate an episode: Take actions using (11) until termination;
7: $G(s) \leftarrow$ sum of discounted rewards from s till terminal state for all states appearing in the episode;
8: Append $G(s)$ to *Returns(s)*;
9: $V(s) \leftarrow$ average(Returns(s));
10: **if** $V(s)$ converges for all s **then**
11: **break**
12: $V^*(s) \leftarrow V(s), \forall s$;
13: **end if**
14: **end for**
15: Use the estimated $V^*(s)$ to find optimal actions using (11).

A popular way to compute the optimal state values, required by the optimal policy as shown in (11), is through value iteration by Monte Carlo computations. The procedure to learn the optimal policy from the IoT environment using Monte Carlo is given in Algorithm 1. Given the parameters N, γ, $\{u_h, r_{sh}, r_{sl}, r_{rh}, r_{rl}\}$, and the data of IoT users $\{u_t\}$, Algorithm 1 shows how to learn the optimal policy for the considered MDP problem. Note that $\{u_t\}$ can be real data from the IoT environment, as well as from simulations if the probability distribution is known. The *Returns* array at line 2 represents a matrix to save the return of each state at every episode, which corresponds to an iteration. At line 3, we initialize all state values with zeros. Starting from the initial state in each iteration $b = 0$, the current state values, which constitutes the current policy, are used to take actions until the terminal state is reached. To promote exploring different states randomized actions can be taken sometimes at line 6 [39]. $G(s)$ in lines 7 and 8 represents a vector of returns of all states appearing in the episode. Inserting these values into the *Returns* array, the state values are updated by taking the average as shown in line 9. The algorithm stops when all state values converge, the converged values are then used to determine actions as in (11).

Similar to (11), we can write the optimal action at state s in terms of $Q^*(s, a)$ as follows,

$$a^* = \begin{cases} serve, & \text{if } Q^*(s, serve) > Q^*(s, reject), \\ reject, & \text{otherwise}. \end{cases} \quad (12)$$

The optimal action-value functions, required by the optimal policy as shown in (12), can be also computed through the value iteration technique using different RL algorithms. The procedure to learn the optimal policy from the IoT

environment using the model-free SARSA, E-SARSA, and Q-learning methods is given in Algorithm 2.

Algorithm 2 Learning Optimum Policy Using QL, E-SARSA, and SARSA

1: Select: $\{\gamma, \epsilon\} \in [0, 1]$, $\alpha \in (0, 1]$, $n \in \{1, 2, \ldots\}$;
2: Input: N (number of RBs);
3: Initialize: $Q(s, a)$ arbitrarily in $\mathbb{Q}$, $\forall(s, a)$;
4: Initialize: $b \leftarrow 0$;
5: **for** $t = 0, 1, 2, \ldots$ **do**
6: Take action a_t according to π (e.g., ϵ-greedy), and store r_t and s_{t+1};
7: **if** $t \geq n - 1$ **then**
8: $\tau \leftarrow t + 1 - n$;
9: QL: $G \leftarrow \sum_{j=\tau}^{t+1} \gamma^{(j-\tau)} r_j + \gamma^n \max_a Q(s_{t+1}, a)$;
10: E-SARSA: $G \leftarrow \sum_{j=\tau}^{t+1} \gamma^{(j-\tau)} r_j + \gamma^n \mathbb{E}_a[Q(s_{t+1}, a)]$;
11: SARSA: $G \leftarrow \sum_{j=\tau}^{t+1} \gamma^{(j-\tau)} r_j + \gamma^n Q(s_{t+1}, a_{t+1})$;
12: $Q(s_\tau, a_\tau) \leftarrow Q(s_\tau, a_\tau) + \alpha[G - Q(s_\tau, a_\tau)]$;
13: Update $\mathbb{Q}$ with $Q(s_\tau, a_\tau)$;
14: **end if**
15: **if** $Q(s, a)$ converges for all (s, a) **then**
16: $Q^*(s, a) \leftarrow Q(s, a)$;
17: **break**
18: **end if**
19: **end for**
20: Use $Q^*(s, a)$ estimated in $\mathbb{Q}$ for π^* using (12)

Algorithm 2 shows how FN learns the optimal policy for the MDP by estimating $Q^*(s, a)$ using QL, E-SARSA, and SARSA methods. The step size parameter α represents the weight we give to the change in our experience, i.e., the learning rate, ϵ is the probability of making a random action for exploration, and the batch size n represents the number of time steps after which we update the $Q(s, a)$ values. The $\mathbb{Q}$ array at line 3 represents a matrix to save the updated values of the action-value functions of all states and actions in each iteration. In each iteration, we take an action, observe and store the collected reward and the successor state. Actions are taken according to a policy π such as the ϵ-greedy policy in line 6, in which a random action with probability ϵ is taken to explore new rewards, and an optimal action (see (12)) is taken with probability $(1 - \epsilon)$ to maximize the rewards; with $\epsilon = 0$, the policy becomes greedy. The condition at line 7 represents the time, in terms of the batch size, at which we start updating the Q values of the actions taken in the previously visited states. The way target G is computed for QL, E-SARSA and SARSA is shown at lines 9-11. G represents the return collected starting from time $(t+1-n)$ to n time-steps ahead, and it contains two parts, the discounted collected rewards and a function of the action-value for future rewards. The latter part changes for QL, E-SARSA and SARSA. For QL, the maximum action-value is used considering all possible actions which can be taken from the state at $t + 1$. Whereas, E-SARSA uses the expected value of $Q(s_{t+1}, a)$ over possible actions at state s_{t+1}, and SARSA uses $Q(s_{t+1}, a_{t+1})$ considering the action that will be taken at time $t + 1$ according to the current policy. The way to update the action-value is shown at line 12, where τ is the time whose Q estimate is being updated. At line 13, the matrix $\mathbb{Q}$ is updated with the new Q value and used to make future decisions. The algorithm stops when all Q values converge. The converged values represent the optimal action values Q^* which are then used to determine optimal actions as in (12).

Recall that the FN objective is to maximize the expected total served utility and minimize the expected termination time, as shown in (2). Hence, to compare the performance of FN when using QL, SARSA, E-SARSA and MC provided in Algorithms 1 and 2 with the performance of a fixed-threshold algorithm in the utility filtering-based network slicing, which does not learn from the interactions with environment, we define an objective performance metric R as

$$R = \mathbb{E}\left[\sum_{m=1}^{M} u_m - \theta(T - M)\right], \tag{13}$$

where a served utility is denoted with u_m, the number of served IoT requests in an episode is denoted with M, $(T - M)$ represents the total idle time for RBs, and θ is a penalty for being idle.

V. SIMULATIONS

We next provide simulation results to evaluate the performance of FN when implementing the RL methods, Q-learning, SARSA, Expected-SARSA, and Monte Carlo, given in Algorithms 1 and 2. We also compare the RL-based FN performance with the FN performance when utility filtering-based network slicing is employed with a fixed thresholding algorithm. We evaluate the performances in various IoT environments with different compositions of IoT latency requirements. For brevity, we do not consider the effect of ratio of the achievable throughput to the throughput requirement in assessing the utility of a service request. Specifically, we consider 10 utility classes with different latency requirements to exemplify the variety of IoT applications in an F-RAN setting. That is, we consider $\zeta = 0$, $\beta = 1, \kappa = 1$ in (1), and discretize the latency-based utility to 10 classes ($U = 10$). The utility values $1, 2, \ldots, 10$ may represent the following IoT applications, respectively: smart farming, smart retail, smart home, wearables, entertainment, smart grid, smart city, industrial Internet, autonomous vehicles, and connected health. By changing the composition of utility classes, we generate 19 scenarios of IoT environments, 6 of which are summarized in Table 3. Higher density of high-utility users makes the IoT environment richer in terms of low-latency IoT applications.

Denoting an IoT environment of a particular utility distribution with $\mathcal{E}$, we show in Table 3 the statistics of $\mathcal{E}_1$, $\mathcal{E}_4$, $\mathcal{E}_7$, $\mathcal{E}_{10}$, $\mathcal{E}_{15}$, and $\mathcal{E}_{19}$. The first 10 rows in the table provide detailed information about the proportion of each utility class in an IoT environment corresponding to a

TABLE 3. Utility distributions for various IoT environments with heterogeneous latency requirements.

	$\mathcal{E}_1$	$\mathcal{E}_4$	$\mathcal{E}_7$	$\mathcal{E}_{10}$	$\mathcal{E}_{15}$	$\mathcal{E}_{19}$
$P(u=1)$	0.015	0.012	0.01	0.008	0.004	0.001
$P(u=2)$	0.073	0.062	0.05	0.038	0.019	0.004
$P(u=3)$	0.365	0.308	0.25	0.192	0.096	0.019
$P(u=4)$	0.292	0.246	0.2	0.154	0.077	0.015
$P(u=5)$	0.205	0.172	0.14	0.108	0.054	0.011
$P(u=6)$	0.014	0.057	0.1	0.142	0.214	0.271
$P(u=7)$	0.013	0.051	0.09	0.129	0.193	0.244
$P(u=8)$	0.011	0.046	0.08	0.114	0.171	0.217
$P(u=9)$	0.009	0.034	0.06	0.086	0.129	0.163
$P(u=10)$	0.003	0.012	0.02	0.029	0.043	0.055
$\rho = P(u>5)$	5%	20%	35%	50%	75%	95%
$\bar{u}$	3.82	4.4	4.97	5.55	6.5	7.27

TABLE 4. Summary of simulation parameters and their values.

Parameter	Description	Value
γ	discount factor	0.7
α	learning rate	0.01
ϵ	probability of random action	0
θ	penalty of idle time	1
n	batch/step size	1
N	total number of resource blocks of FN	15
r_{sh}	reward for serving high-utility user	2
r_{sl}	reward for serving low-utility user	-1
r_{rh}	reward for rejecting high-utility user	-2
r_{rl}	reward for rejecting low-utility user	1
u_h	the threshold for "high-utility"	mean

latency requirement. The last two rows illustrate the quality or richness of IoT environments, where ρ is the probability of a utility being greater than 5, and $\bar{u}$ is the mean value of utilities in the environment. In the considered 19 scenarios, ρ increases in steps of 0.05 from 5% to 95% for $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_{19}$ respectively. The remaining 13 scenarios have statistics proportional to their ρ values. We started with a general scenario given by $\mathcal{E}_7$, and changed ρ to obtain the other scenarios.

The simulation parameters shown in Table 4 are used for the presented results in this section. The rewards $\mathcal{R} = \{r_{sh}, r_{sl}, r_{rh}, r_{rl}\}$ are chosen to facilitate learning the optimal policy. We consider that the FN is equipped with computing, signal processing and storage resources of 15 resource blocks (RBs), i.e., $N = 15$. In a particular environment $\mathcal{E}$, the threshold that defines "high utility" is set to the mean of all utilities, i.e., $u_h = \bar{u}$. We applied the greedy policy in our simulations, hence $\epsilon = 0$.

We firstly consider the MDP formulation for the IoT environment given by scenario $\mathcal{E}_7$ shown in Table 3. By interaction with the environment, the FN updates the state value functions which converge to the optimum policy. Fig. 4, shows how the FN learns the optimal policy using the Monte Carlo (MC) method given in Algorithm 1 to estimate the optimal state values. With 15 RBs, there are 160 states, the last 10 of which are terminal states with $b = 15$ for which $V(s) = 0$. The state-value functions of 16 states are given in Fig. 4. The remaining states have values within a standard deviation $\sigma = 0.5$ of the selected 16 states. It is seen that for most of the states the state values converges to the optimal

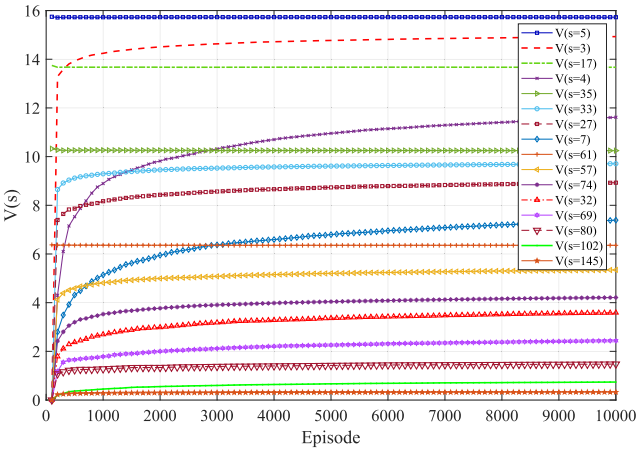

FIGURE 4. Learning optimum policy of the MDP by applying the Monte Carlo method given by Algorithm 1 to obtain the optimal state values required in (11). The IoT environment $\mathcal{E}_7$ is considered, and the FN is equipped with 15 RBs. The 16 state values shown in the figure are a sample of the 150 non-terminal state values.

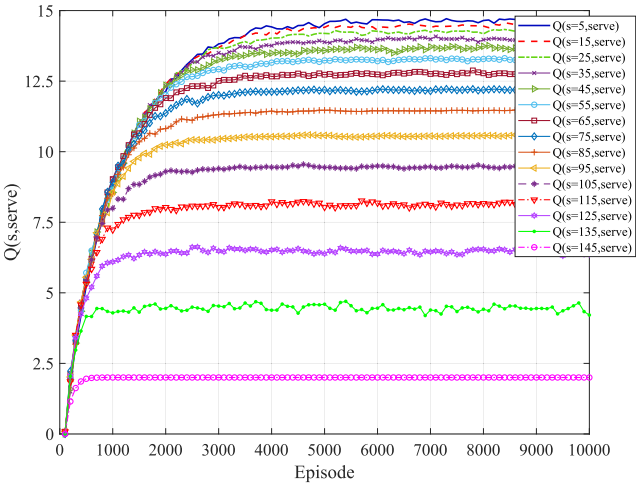

FIGURE 5. Learning the optimal action-value function $Q^*(s, serve)$ required in (12) using the Q-learning method given by Algorithm 2. Q-values converge to the optimal values after around 4000 episodes. The IoT environment $\mathcal{E}_7$ is considered, and the FN is equipped with 15 RBs.

value $V^*(s)$ after about 5000 iterations. This number can be easily exceeded by the number of requests received by FN during a busy hour from a variety of IoT applications [1].

We next apply SARSA, Expected SARSA and QL in the IoT environment $\mathcal{E}_7$, for learning the optimal policy in (12) using the estimated $Q^*(s, a)$ in Algorithm 2. The convergence of $Q(s, serve)$ and $Q(s, reject)$ when using QL is shown in Figs. 5 and 6, respectively. In our MDP problem, QL converges slightly faster than E-SARSA, SARSA and MC since it implements a greedy approach by selecting the maximum $Q(s', a')$ when updating the return G_t as shown in Algorithm 2. However, this is not a general rule as it depends on the nature of each problem. There are many factors affecting the convergence rate, e.g., large values of the learning rate α make the Q-values bounce around a mean value, whereas small values causes it to converge slowly.

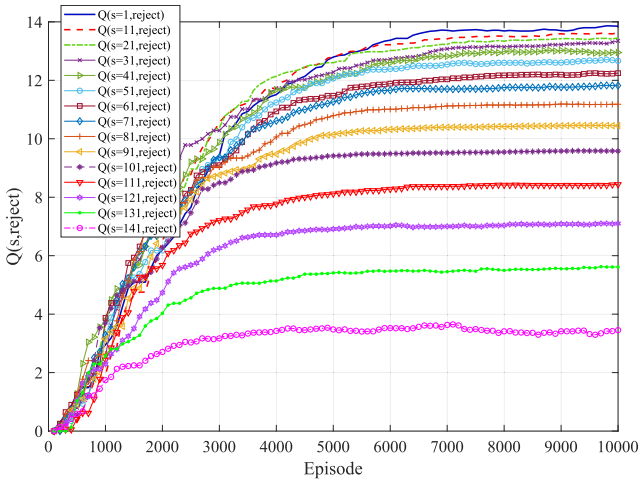

FIGURE 6. Learning the optimal action-value function $Q^*(s, reject)$ required in (12) using the Q-learning method given by Algorithm 2. Q-values converge to the optimal values after around 5000 episodes. The IoT environment $\mathcal{E}_7$ is considered, and the FN is equipped with 15 RBs.

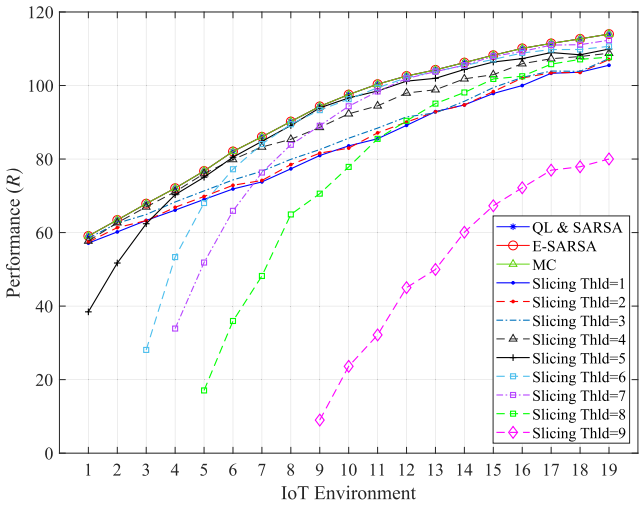

FIGURE 7. The performance in terms of R for the FN with $N = 15$, in various IoT environments when applying the RL methods (QL, SARSA, E-SARSA and MC) given in Algorithms 1 and 2, and the utility filtering algorithm in network slicing with different slicing thresholds. RL methods' performances are indistinguishable here, and better than the conventional-filtering based network slicing in all environments thanks to their learning and adaptation capability.

Unnecessary exploration makes the convergence slower, controlled by the ϵ value in the ϵ-greedy policy. The step size n after which we update the state values or Q-values affects also the convergence dependent on the problem. For instance, MC updates the state values at the end of an episode regardless of how long it is, which makes it slower to exploit the updated state values in making better actions, whereas QL, SARSA and E-SARSA using $n = 1$ update the Q-value every time step. Unlike MC, the FN needs to keep updating two Q-values for each state instead of updating one state value. Hence, we have 300 Q-values to update in order to learn the optimal policy.

We compare the performance of the RL methods, in terms of R, as shown in (13), with that of the utility filtering-based

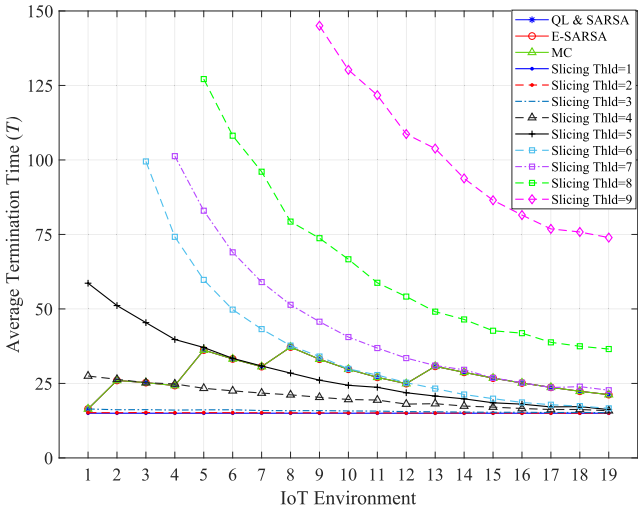

FIGURE 8. The average termination time T in time-steps for FN with $N = 15$ in various IoT environments when applying the RL methods (QL, SARSA, E-SARSA and MC) given by Algorithms 1 and 2, and the utility filtering algorithm in network slicing with different slicing thresholds. RL methods manage to have a steady termination time in all environments.

network slicing with various slicing thresholds in the 19 IoT environments. The utility filtering algorithm uses the same threshold for network slicing regardless of the environment. For the RL methods, we consider the simulation setup shown in Table 4, and for the utility filtering-based network slicing we consider all possible slicing thresholds $1, 2, \ldots, 10$. As shown in Figs. 7 and 8, the RL methods exhibit the best performance as they learn how to balance early termination with higher total served utilities. It never terminates too early or too late ($T \approx 27$ for all environments as seen in Fig. 8), as opposed to the utility filtering-based network slicing which is not adaptive to the environment. As seen in Fig. 7, the performance of the utility filtering algorithm with slicing thresholds $1, 2, 3, 8, 9$ are steadily below that of the RL algorithms. The average termination time for slicing thresholds $1, 2$, and 3 is about 15 which is the minimum termination time, though they could not achieve good performance. Slicing threshold 4 has a comparable performance to RL for the environments $\mathcal{E}_2 - \mathcal{E}_5$, after which its performance starts to decline. Although slicing thresholds $5, 6, 7$ have good performances close to RL for environments with medium to high ρ, they perform far from RL for IoT environments with small ρ. The performance of slicing threshold 10 is much worse than threshold 9 for all environments due to the long termination time which exceeds 280 time-steps, thus it does not appear in Figs. 7 and 8.

The performance of the RL methods is very close to each other, hence it is not easy to distinguish them in Figs. 7 and 8. For a clearer view, Fig. 9 compares the performance of the four RL methods in terms of the performance ratio with respect to performance of slicing threshold 4, i.e., (R_{RL}/R_{Thld4}). QL has the best performance with an average performance ratio of 104% in all IoT environments with a peak of 106% in $\mathcal{E}_9$, followed by E-SARSA and MC.

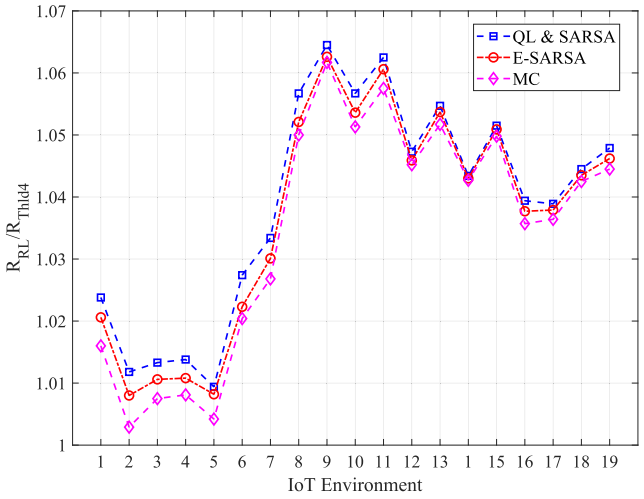

FIGURE 9. Comparison between the performance of RL methods in terms of relative performance with respect to the utility filtering algorithm in network slicing with slicing threshold 4. QL and SARSA coincide due to the greedy policy is used in the simulations.

SARSA has the same performance as QL because greedy policy, i.e., $\epsilon = 0$, was used.

VI. CONCLUSION

We proposed a Markov Decision Process (MDP) formulation for the resource allocation problem in Fog RAN for IoT services with heterogeneous latency requirements. Several reinforcement learning (RL) methods, namely Q-learning, SARSA, Expected SARSA, and Monte Carlo, were discussed for learning the optimum decision-making policy adaptive to the IoT environment. Their superior performance over utility filtering-based network slicing methods, and adaptivity to the IoT environment were verified through extensive simulations. The RL methods strike a right balance between the two conflicting objectives, maximize the average total served utility vs. minimize the fog node's idle time, which helps utilize fog node's limited resource blocks efficiently. As future work we consider expanding the presented resource allocation framework to more challenging scenarios such as dynamic resource allocation with heterogeneous service times and number of resource blocks needed, and collaborative resource allocation with multiple fog nodes.

REFERENCES

[1] Cisco Syst., "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," Cisco Syst., Corporate Headquarters, San Jose, CA, USA, White Paper, 2018. Accessed: Sep. 10, 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html and https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf

[2] A. T. Nassar, A. I. Sulyman, and A. Alsanie, "Achievable RF coverage and system capacity using millimeter wave cellular technologies in 5G networks," in *Proc. IEEE 27th Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2014, pp. 1–6.

[3] A. I. Sulyman, A. T. Nassar, M. K. Samimi, G. R. MacCartney, Jr., T. S. Rappaport, and A. Alsanie, "Radio propagation path loss models for 5G cellular networks in the 28 GHz and 38 GHz millimeter-wave bands," *IEEE Commun. Mag.*, vol. 52, no. 9, pp. 78–86, Sep. 2014.

[4] B. Yang, Z. Yu, J. Lan, R. Zhang, J. Zhou, and W. Hong, "Digital beamforming-based massive MIMO transceiver for 5G millimeter-wave communications," *IEEE Trans. Microw. Theory Techn.*, vol. 66, no. 7, pp. 3403–3418, Jul. 2018.

[5] S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter-wave cellular wireless networks: Potentials and challenges," *Proc. IEEE*, vol. 102, no. 3, pp. 366–385, Mar. 2014.

[6] J. Zhang, Z. Zheng, Y. Zhang, J. Xi, X. Zhao, and G. Gui, "3D MIMO for 5G NR: Several observations from 32 to massive 256 antennas based on channel measurement," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 62–70, Mar. 2018.

[7] S.-H. Park, O. Simeone, and S. Shamai (Shitz), "Joint optimization of cloud and edge processing for fog radio access networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 315–319.

[8] M. Peng, Y. Sun, X. Li, Z. Mao, and C. Wang, "Recent advances in cloud radio access networks: System architectures, key techniques, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2282–2308, Aug. 2016.

[9] Z. Zhao, M. Peng, Z. Ding, W. Wang, and H. V. Poor, "Cluster content caching: An energy-efficient approach to improve quality of service in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1207–1221, May 2016.

[10] M. Peng, C. Wang, V. Lau, and H. V. Poor, "Fronthaul-constrained cloud radio access networks: Insights and challenges," *IEEE Wireless Commun.*, vol. 22, no. 2, pp. 152–160, Apr. 2015.

[11] W. Wang, V. K. N. Lau, and M. Peng, "Delay-aware uplink fronthaul allocation in cloud radio access networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 7, pp. 4275–4287, Jul. 2017.

[12] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[13] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Netw.*, vol. 31, no. 1, pp. 52–58, Jan. 2017.

[14] G. P. Fettweis, "The tactile Internet: Applications and challenges," *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[15] Q. Zheng, K. Zheng, H. Zhang, and V. C. M. Leung, "Delay-optimal virtualized radio resource scheduling in software-defined vehicular networks via stochastic learning," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7857–7867, Oct. 2016.

[16] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Müller, T. Elste, and M. Windisch, "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 70–78, Feb. 2017.

[17] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge mesh: A new paradigm to enable distributed intelligence in Internet of Things," *IEEE Access*, vol. 5, pp. 16441–16458, 2017.

[18] A.-C. Pang, W.-H. Chung, T.-C. Chiu, and J. Zhang, "Latency-driven cooperative task computing in multi-user fog-radio access networks," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 615–624.

[19] G. M. S. Rahman, M. Peng, K. Zhang, and S. Chen, "Radio resource allocation for achieving ultra-low latency in fog radio access networks," *IEEE Access*, vol. 6, pp. 17442–17454, 2018.

[20] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. 81st IEEE Veh. Technol. Conf. (VTC Spring)*, May 2015, pp. 1–6.

[21] T.-C. Chiu, W.-H. Chung, A.-C. Pang, Y.-J. Yu, and P.-H. Yen, "Ultra-low latency service provision in 5G fog-radio access networks," in *Proc. IEEE 27th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Sep. 2016, pp. 1–6.

[22] E. Balevi and R. D. Gitlin, "Optimizing the number of fog nodes for cloud-fog-thing networks," *IEEE Access*, vol. 6, pp. 11173–11183, 2018.

[23] T. Gao, M. Chen, H. Gu, and C. Yin, "Reinforcement learning based resource allocation in cache-enabled small cell networks with mobile users," in *Proc. IEEE/CIC Int. Conf. Commun. China*, Oct. 2017, pp. 1–6.

[24] D.-N. Vu, N.-N. Dao, and S. Cho, "Downlink sum-rate optimization leveraging hungarian method in fog radio access networks," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2018, pp. 56–60.

[25] Y.-J. Liu, S.-M. Cheng, and Y.-L. Hsueh, "eNB selection for machine type communications using reinforcement learning based Markov decision process," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11330–11338, Dec. 2017.

[26] M. Condoluci, T. Mahmoodi, E. Steinbach, and M. Dohler, "Soft resource reservation for low-delayed teleoperation over mobile networks," *IEEE Access*, vol. 5, pp. 10445–10455, 2017.
[27] H. A. M. Name, F. O. Oladipo, and E. Ariwa, "User mobility and resource scheduling and management in fog computing to support IoT devices," in *Proc. 7th Int. Conf. Innov. Comput. Technol. (INTECH)*, Aug. 2017, pp. 191–196.
[28] M. Peng and K. Zhang, "Recent advances in fog radio access networks: Performance analysis and radio resource allocation," *IEEE Access*, vol. 4, pp. 5003–5009, 2016.
[29] T. Park, N. Abuzainab, and W. Saad, "Learning how to communicate in the Internet of Things: Finite resources and heterogeneity," *IEEE Access*, vol. 4, pp. 7063–7073, 2016.
[30] M. Yan, G. Feng, and S. Qin, "Multi-RAT access based on multi-agent reinforcement learning," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.
[31] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 680–692, Jan. 2018.
[32] H. Zhang, Y. Qiu, K. Long, G. K. Karagiannidis, X. Wang, and A. Nallanathan, "Resource allocation in NOMA based fog radio access networks," 2018, *arXiv:1803.05641*. [Online]. Available: https://arxiv.org/abs/1803.05641
[33] N. Mostafa, I. Al Ridhawi, and M. Aloqaily, "Fog resource selection using historical executions," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Apr. 2018, pp. 272–276.
[34] H. Xiang, W. Zhou, M. Daneshmand, and M. Peng, "Network slicing in fog radio access networks: Issues and challenges," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 110–116, Dec. 2017.
[35] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.
[36] T. Dang and M. Peng, "Delay-aware radio resource allocation optimization for network slicing in fog radio access networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.
[37] L. Tang, X. Zhang, H. Xiang, Y. Sun, and M. Peng, "Joint resource allocation and caching placement for network slicing in fog radio access networks," in *Proc. IEEE 18th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2017, pp. 1–6.
[38] Y. Sun, M. Peng, S. Mao, and S. Yan, "Hierarchical radio resource allocation for network slicing in fog radio access networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3866–3881, Apr. 2019.
[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

ALMUTHANNA NASSAR received the B.Sc. degree in electrical engineering from Jordan University of Science and Technology, Irbid, Jordan, in 2006, and the M.Sc. degree in electrical engineering from King Saud University, Riyadh, Saudi Arabia, in 2014. He is currently pursuing the Ph.D. degree in electrical engineering with the University of South Florida, Tampa, FL, USA, where he is also with the Secure and Intelligent Systems Laboratory, EE Department. He was a Radio Access Network Planning Specialist Manager with Etihad Etisalat Company (mobily), Riyadh, from 2009 to 2017. He has more than ten years of industry experience in RAN planning and design of multi-technology cellular networks. His current research interests include the IoT, 5G communications, and reinforcement learning.

YASIN YILMAZ (S'11–M'14) received the Ph.D. degree in electrical engineering from Columbia University, New York, NY, USA, in 2014. He is currently an Assistant Professor of electrical engineering with the University of South Florida, Tampa, FL, USA. His current research interests include statistical signal processing, machine learning, and their applications to cybersecurity, cyber-physical systems, the Internet-of-Things networks, communication systems, energy systems, transportation systems, and social and environmental systems. He was a recipient of the Collaborative Research Award from Columbia University, in 2015, and the Research Initiation Award from the Southeastern Center for Electrical Engineering Education, in 2017.