ACT on Monte Carlo FogRA for Time-Critical Applications of IoT

A. S. Gowri, P. Shanthi Bala, Zion Ramdinthara, T. Siva Kumar
Department of Computer Science, School of Engineering and Technology
Pondicherry University, Puducherry, India

***Abstract*—The need for instantaneous processing for Internet of Things (IoT) has led to the notion of fog computing where computation is performed at the proximity of the data source. Though fog computing reduces the latency and bandwidth bottlenecks, the scarcity of fog nodes hampers its efficiency. Also, due to the heterogeneity and stochastic behavior of IoT, traditional resource allocation technique does not suffice the time-sensitiveness of the applications. Therefore, adopting Artificial Intelligence (AI) based Reinforcement Learning approach that has the ability to self-learn and adapt to the dynamic environment is sought. The purpose of the work is to propose an Auto Centric Threshold (ACT) enabled Monte Carlo FogRA system that maximizes the utilization of Fog's limited resources with minimum termination time for time-critical IoT requests. FogRA is devised as a Reinforcement Learning (RL) problem, that obtains optimal solutions through continuous interaction with the uncertain environment. Experimental results show that the optimal value achieved by the proposed system is increased by 41% more than the baseline adaptive RA model. The efficiency of FogRA is evaluated under different performance metrics.**

Keywords*—Cloud; edge; fog; Internet of Things (IoT); Reinforcement Learning (RL)

I. INTRODUCTION

The evolution of smart devices has led to the proliferation of the Internet of Things (IoT), thus making the world, a better-connected place for wireless communication and high-speed applications. The "Little Data, Big Stream" notion of IoT is mostly about time-sensitive applications [1]. The number of connected devices per capita is anticipated to be 01 trillion devices by 2025. This massive usage of IoT will generate mobile data services of 150 zettabytes by 2025 which corresponds to 5-7 times of IP traffic today [2]. The immense amount of mobile data can be processed instantaneously, only if the computing facility is available near the data source.

Ultra-low latency, energy efficiency, distributed processing, and storage are a few of the expectations of IoT applications. So far, the cloud was tailored to tackle these demands. But, in reality, the remotely located cloud causes delay and hinders the QoS requirements of the IoT requests [3]. By the time the data from IoT is transmitted to the centralized cloud, the inevitability to act on it might be gone, which cost lives. Disastrous management, Industrial IoT, and real-time aeronautical cum nuclear reactions are some of the time-critical IoT applications where the response delay of even nanoseconds makes a huge difference [4]. Hence, a fog computing paradigm that addresses time-critical applications in its proximity is recommended.

Fog computing is a distributed paradigm where the processing nodes are dispersed geographically near the data source. The proximal distribution of the processing elements promises ultra-low latency for time-critical applications [5]. IoT requires a Resource Allocation (RA) mechanism that prioritizes time-critical tasks over others. Also, a sufficient amount of fog resources may not be available, as and when required. Hence, allocation of compute nodes to the incoming IoT requests, in a resource-constrained fog environment is challenging [6]. The RA mechanism has to adopt an optimal strategy to make a sequence of smart decisions [7]. Although evolutionary algorithms, dynamic programming, and policy gradient are commonly used to derive optimal policy, these techniques require prior knowledge of the model [8]. But, the proposed work FogRA is a sequential decision-making problem in a model-free environment.

The environment involves IoT devices whose behaviour is stochastic and hence the dynamics of the model are not known. In such a case, Reinforcement Learning (RL) is sought to solve the FogRA problem for two reasons. First, RL is a machine learning, trial and error methodology that can self-learn and adapt through continuous interaction in an uncertain environment. Second, by its origin in Markov Decision Process (MDP), RL is a sequential decision theory that generates high-quality decisions in the long term [9].

Monte Carlo (MC), MC-Exploring Starts (MC-ES), and On-Policy Monte Carlo Control (OMC) are variants of RL algorithms that compute optimal policy. They are the straightforward methods that do not bootstrap and eliminate the curse of dimensionality problem [8]. They compute the optimal value by averaging the samples obtained through various iterations and derive the optimal policy from it.

Studies reveal that the existing RA works focused more on the development of frameworks that reduced latency and increased the quality of service [10]. The works carried out using feed-forward NN, MINLP, and other optimization techniques also obtained better outcomes [11]. But these techniques were heuristic-based and time-consuming. Q-learning algorithm of RL was mostly used to devise RA strategy. The combination of RL algorithms with queuing theory and neural networks too proved more efficient but involved cost overhead.

Moreover, the earlier works addressed intra-dependent and parallel tasks, while focus on time-critical applications was hardly found. Further, the earlier RL-based RA works were mostly proactive, in the sense that they depend on history for

prediction and hence mostly model-based [12]. But, the proposed work allocates the fog resources on demand. The reactive technique does not require any dynamics of the model. The availability of the scarce fog resources, and the heterogenous latency requirement of IoT describes the uncertain behavior of the environment. Allocation of fog's limited resources and prioritizing the time-critical requests are highly challenging that need attention. The proposed work is about designing an efficient FogRA system that handles the challenges of model-free stochastic environment at ease.

The proposed work employs ACT enabled MC approach to construct the FogRA system for time-critical IoT applications. The FogRA system involves a smart agent that finds the optimal policy to allocate the fog's limited resources to the time-critical requests. The significance of the work is perceived by its usage in the modern network environment and its potential for the fog computing paradigm. The main contribution of the intended work is summarized as follows:

- A Reinforcement Learning (RL) based FogRA system is developed in which the Fog nodes are used as computing resources, to process incoming requests.
- In the pursuit of maximizing fog utilization, the Fog Controller Agent (FCA) undergoes learning to allocate its resources to time-critical application requests.
- The ACT-enabled FogRA system is proposed using Monte Carlo (MC), MC-Exploring Starts (MC-ES), and On-Policy Monte Carlo Control (OMC).
- The proposed system is compared with an existing Adaptive RA system to evaluate its performance.
- Result demonstrates that the optimal long-term reward achieved by ACT-enabled FogRA is 41% more than the existing RA model.

The rest of the article is structured as follows: Section II discusses the background knowledge that substantiates FogRA as an RL problem. An overview of the RL approach and the formulation of FogRA as an MDP are briefly discussed in this section. The system model of the proposed work is elaborated in Section III. The experimental results are evaluated and analyzed in Section IV, and Section V concludes the paper with prospects for the future.

II. Reinforcement Learning Background

A. Resource Allocation as RL Problem

As mobile apps go more and more 24/7, online solutions that deliver instantaneous decisions are highly sought. The time-critical IoT requests demand response with almost negligible delay [13]. To achieve delay-less response, fog nodes that reside near the edge devices are used as computing elements. But the distributed nature of fog creates a scarcity of its resources. This makes the allocation of fog resources tedious, hence is considered an essential problem to be dealt with [14]. Due to the limited resources, not all the requests are served in the fog layer. It has to be constantly monitored whether the incoming request is time-critical or not. Prioritizing the time-critical applications to be served in the fog delivers prompt service with negligible and tolerable delay [15].

The FogRA comprises a Fog Controller Agent (FCA) that allocates the available fog resources to the most time-critical tasks. Through continuous interaction with the environment, the agent derives the optimal policy through which it decides whether to allocate the fog resource or not. The problem that involves learning and decision-making as a continuous process is defined as the RL problem [16]. As the allocation of fog resources to IoT requests need constant learning, and decision making, FogRA is considered an RL problem.

RL problem consists of the agent and the environment as the two main components as shown in Fig. 1. An agent is a learner and decision-maker. The environment is the entity with which the agent interacts [17]. Markov Decision Process (MDP) is used to articulate the interaction between the agent and the environment in terms of states, actions, and rewards, MDP is the mathematical framework to define how the environment behaves in response to the agent's action. The Markov process is a memoryless random process [8]. It states that the future is independent of the past given its present. Any environment in which the current state is sufficient to determine the next state, irrespective of its previous states (history) is said to possess the Markov property. Thus, MDP eliminates the need to preserve the value of past states thereby reducing the memory cost considerably.

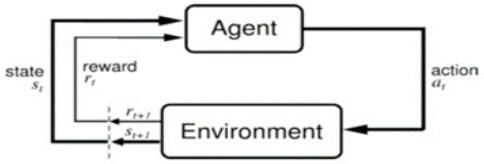

Fig. 1. Agent-Environment Interaction in RL [8].

The agent observes the environment's state ($s_t \in$ S) and performs one of the actions ($a_t \in$A(s)) at each time step (t) of the interaction. The agent's activity results in a reward ($r_{t+1} \in$R) and a transition to the following state (s_{t+1}). Hence a sequence of trajectory $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2 r_3 \ldots\ldots\ldots$ is observed [18]. The crucial constraint on the agent is that it operates in an uncertain environment.

In most problems, the agent is not guided about the action that it has to carry out, instead, by trial and error, it learns the right action. The agent likely acts differently in a state for the first time, rather than after visiting the same state many times [19]. With experience, the agent learns the consequence of its action on that state. The consequence of an action influences not only the immediate reward but the next state and its subsequent rewards.

RL is a computational approach to designing a goal-directed learning agent that interacts with an uncertain environment [20]. In the Long-term, the agent seeks to maximize the cumulative payoff. At every time step (t), the agent receives a reward in the form of a scalar value which is either positive or negative. A positive value specifies how good the current action is, whereas a negative value indicates the penalty for the wrong action. The sum of the rewards starting from time 't' until the termination time 'T' is defined as returns (G_t). Thus, returns is

the metric that the agent aims to maximize in the long run. The definition of returns varies depending upon whether the task taken into consideration is episodic or continuous.

In Episodic tasks, the interaction between the agent and the environment breaks once the terminal state is reached [8]. The next episode starts in a state independent of the previous one ended. Tasks in episodes of this kind are called episodic tasks. Returns (G_t) for the episodic tasks starting from time 't' is expressed as the sum of undiscounted rewards as shown in equation (1).

$$G_t = r_{t+1} + r_{t+2} + r_{t+2} + r_{t+3} + \cdots \ldots \ldots \ldots r_T \quad (1)$$

where r_{t+1} is the reward obtained at time-step (t+1) as an effect of the action (a_t) taken at time-step (t) on the state (s_t). The Time of termination (T), varies from episode to episode. The FogRA problem considered in the work is episodic. Each episode ends when all the fog resources are allocated to the incoming requests. Then the system is reset with the maximum number of fog nodes for the next episode.

On the other hand, for the tasks with continuous states, the interaction between the agent and the environment does not break, but rather continues without any terminal state [8]. Such tasks that do not have an identifiable terminal state are called continuous tasks. The rewards obtained at each time-step of continuous task accumulate to a big value that becomes uncountable. Also, the immediate reward got is more valuable than the one obtained in the future [21]. Hence, the value of the future reward is discounted by a factor of gamma (γ).

The value of gamma ranges from 0 to 1. When γ=0, the agent is myopic and concerned about maximizing the immediate reward, whereas γ=1 specifies that the agent gives more importance to the future reward. The literature study shows that values between 0.2 and 0.8 were found to be optimal in many scenarios [15]. Thus, returns (G_t) for the continuous tasks starting from time 't' is expressed as the sum of discounted rewards as shown in equation (2).

$$G_t = r_{t+1} + \gamma^1 r_{t+2} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \quad (2)$$

where G is the reward gained at time-step (t+1) as feedback of the action (a_t) taken at time-step (t) on the state (s_t). The FogRA problem in the work is evaluated as an episodic task, with an initial and terminal state.

The concepts of Value function and Policy play a major role in implementing RL methodology [8]. The value function is the sum of all rewards that are expected in the future from every subsequent state. While the reward signal defines what is beneficial in the immediate sense, the value function signifies what is beneficial in the long run. In short, the reward is the immediate feedback for the current action, whereas the value function is the long-term estimation of rewards [18]. The goal of the agent is to maximize the long-term rewards in the form of the value function. Also, there is no value function without reward. As the iteration proceeds, the actions made by the agent are based on the value function.

The state value function $V(s)$ given by the equation (3), is the expectation of the returns (G_t) at the time 't' from the state s. It is the sum of immediate reward (r_{t+1}) and the discounted value of the next state $V(s_{t+1})$ estimated iteratively, following the sequence of observation, starting from the state.

$$V(s) = \mathbb{E}[G_t | s_t = s] = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1}) | s_t = s] \quad (3)$$

The value of a state is expressed as a function of expectation because future states are stochastic. Similarly, the action-value function ($Q(s,a)$) given by equation (4), is the expectation on the Returns (G_t) for the action $'a'$, taken at time t, in the state $'s'$.

$$Q(s,a) = \mathbb{E}[G_t | s_t = s, a_t = a]$$
$$= \mathbb{E}[r_{t+1} + \gamma Q[s_{t+1}, a_t | s_t = s, a_t = a] \quad (4)$$

The value obtained by these value functions helps the agent to decide the best action at a particular instant of time. Hence it is essential to know the value of the state and state-action pair. Identifying the best action for a state is termed as policy. Policies are the rules or the strategy the agent adopts to maximize the return in the long run [22]. Policies determine the optimal action that the agent has to adopt to achieve its goal.

The value of a state $V(s)$ following the policy (π) is written as $V_\pi(s)$. Consequently, the value of taking action $'a'$ in state $'s'$and following the policy (π) thereafter is denoted as $Q_\pi(s,a)$. The agent looks for a policy that delivers maximum value for the state rather than the highest immediate reward. Unlikely, it is difficult to determine the state value $V(s)$, compared to the reward. Because rewards are the instantaneous feedback from the environment while value functions are long-run values that are estimated iteratively until the value converges with an optimal policy [15]. The value of the state resulting after convergence is the optimal value through which an optimal policy is derived. The agent uses the optimal policy to take the best action thereafter.

The Bellman Optimality equation is one of the methods to find the optimal value function [8]. The optimal value function $V^*(s)$ is one which yields the highest returns compared to all other value functions. $V^*(s)$ is expressed as a value function obtained by taking maximum over the policy (π) as given in the equation (5).

$$V^*(s) = max_\pi V_\pi(s) \quad (5)$$

where $V^*(s)$ signifies the maximum quantity of long-term returns that is obtained from the system. Optimal policies also share optimal state-action value pairs $Q^*(s,a)$. The Bellman Optimality equation for Q^* is given by equation (6), which states that the optimal state action-value function is the one that is the maximum of all action functions following the policy π.

$$Q^*(s,a) = max_\pi Q_\pi(s,a) \quad (6)$$

The goal of the RL problem is to find the optimal policy that yields the highest returns in the long run. A policy π is better than another policy π' if the value of a state obtained by following policy π is greater than the value of the state obtained by the following policy π' ie., $V_\pi(s) \geq V_{\pi'}(s)$. The optimal policy is derived by taking the maximum over the policy 'π' under the state 's' of the value function as given in equation (7).

$$\pi^* = argmax_\pi V_\pi(s), \forall s \quad (7)$$

Computation of optimal policy involves iterative estimation of the value function. RL is a method for the efficient estimation of value functions and optimal policy. As the FogRA problem taken into account is model-free, Monte Carlo (MC), MC Exploring Starts (MC-ES), and On-Policy Monte Carlo Controls methodologies are employed to compute the optimal policy.

B. Monte Carlo Approach

Monte Carlo (MC) is a model-free RL algorithm that learns by averaging the samples drawn from the observation [15]. It learns to derive the policy in an environment where transition probabilities and reward distribution are not known [8]. Hence, the proposed FogRA problem adopts the Monte Carlo method to estimate the optimal policy. An episode is generated starting from an initial state till the terminal state. The trajectory of an episode in the MC algorithm is shown in Fig. 2.

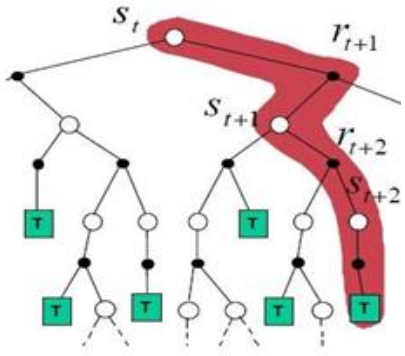

Fig. 2. Back-up Diagram for Monte Carlo Approach [8].

At any instant of time 't' the agent observes the state 's_t' chooses an action 'a_t', transits to a new state s_{t+1}, and gains a reward 'r_{t+1}'. Returns G_t is computed for every episode. Then the value of the state $V(s)$ is estimated by averaging the returns. This process is repeated till $V(s)$ converges for all states and thus obtains the optimal value $V^*(s)$ for every state. Then the optimal policy is derived for each state using the equation (7).

Given some experience (samples), FCA estimates the value of states following policy $V_\pi(s)$, for all the non-terminal states s_t that occur in the trajectory. The agent waits until the returns G_t following the visit is known, then use the returns as a target to update $V_\pi(S_t)$ as given in equation (8).

$$V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha[G_t - V_\pi(S_t)] \quad (8)$$

where G_t, is the actual returns following the time 't', and $'\alpha'$ is the step size at which the agent learns. Alpha $'\alpha'$ is the learning rate hyperparameter. It balances the weight that has been observed in the recent past with the weight of the newly observed target. Equation (8) is also called Exponential Recency Weighted Average (ERWA), which estimates the incremental moving average by giving more weight to the immediate reward [18]. It very much suits a non-stationary environment like FogRA in which the action and the reward undergo continuous learning.

C. Monte Carlo Exploring Starts (MC-ES)

Monte Carlo (MC) estimates the optimal value function $V(s)$ for all states appearing in the episode. It simply looks ahead and chooses the action that leads to the best combination of reward and the next state, which means it does not consider the choice of actions in the state. Another issue is that MC works better when the state space is known in advance. Hence, finding $Q(s,a)$ through MC-ES is the best way, as it considers the choice of actions across all the states and finds the optimal policy by maximizing the action value that produces high returns in the long run [23].

The difference is that the existing adaptive RA model used the MC approach to estimate the value of state '$V(s)$', while the MC-ES uses the state-action value pair '$Q(s,a)$' to improve the policy. The change in the algorithm is that, the returns 'G_t' is computed as $G_t(s,a)$ rather than $G_t(s)$. Hence, MC-ES evaluates $Q(s,a)$ for all state-action pairs as given in equation (9).

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[G_t(s,a) - Q(s_t,a_t)] \quad (9)$$

Then the optimal policy $\pi^*(a/s)$ is derived by choosing the action with maximum action value for each state $s_t \in S$ as given by the equation (10).

$$\pi^*(a/s) \leftarrow argmax_a\, Q(s,a) \quad (10)$$

D. On Policy Monte Carlo Control (OMC)

The estimation of state-action value function $Q(s,a)$ in MC-ES give rise to two issues. First, it increases the state space from *S to (SxA)* leading to memory and time complexity [8]. Secondly, the agent might not be able to explore all state-action pairs, if it acts too greedy towards the policy from the start. Hence, the value of $Q(s,a)$ and the policy obtained through MC-ES, cannot be optimal without a proper balance of exploration and exploitation [23]. Instead, the On-policy Monte Carlo Controls (OMC) estimates the optimal policy through the epsilon (ϵ) greedy approach. The agent picks a random action at epsilon (ϵ) times and acts greedy during the period (1- ϵ) times thus balancing the explore-exploit instability [24]. The Epsilon is a value chosen between zero and one by trial and error, and it is problem-dependent.

III. The Proposed Work

A. System Model

The FogRA system comprises IoT devices and Fog nodes as its core components. The fog nodes reside between the IoT devices and the cloud. These fog nodes are equipped with computing cum network functionality to process incoming requests. FogRA is modeled as an RL problem, where a fog node act as the Fog Controller Agent (FCA). Monitoring the number of fog nodes utilized, observing the time criticality of the incoming request, and then deciding to act accordingly is the prime process of FCA. The fog nodes and the requests from the IoT devices at the edge network make up the environment. The agent-environment interaction of the RL-based FogRA system is portrayed in Fig. 3.

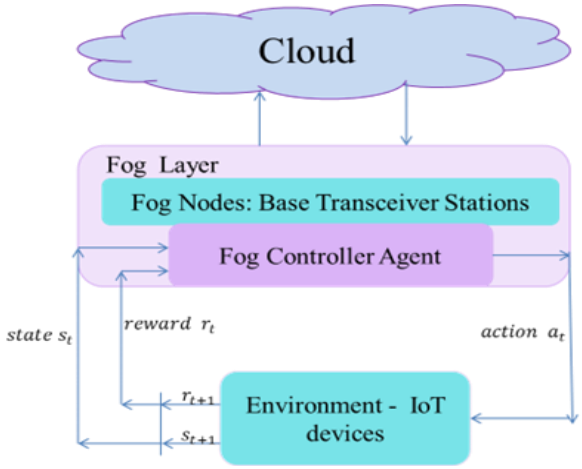

Fig. 3. The FogRA System.

At any instant of time, the time-criticality of the incoming request and the number of utilized fog nodes define the state (s_t). The FCA's decision either to accept the request in fog or decline it forms the action. As a consequence of the action taken, the FCA receives either a positive or negative reward as feedback. The heterogeneity of time-criticality from the IoT and the fog's limited resources make the environment stochastic.

The objective of the ACT-enabled FogRA system is to allocate the fog's limited resources to the time-critical applications within a short time. In the pursuit of the objective, the system maximizes the accepted requests, rewards, and value of state-action pairs which is termed Fog Utilization (U_{Fog}). Fog's limited resource forms the constraint of the resource allocation problem. The objective of the FogRA system is expressed as a multi-objective optimization problem as given in the equation (11).

$$Maximize\ U_{Fog} = \sum_{t=0}^{T} N_{accept} + \sum_{t=0}^{T} rwd_{accept} + VF_{opt}$$

and $Minimize\ F_{converge} = \sum_{t=0}^{T} \mathbb{1}_{Time\ to\ obtain\ VF_{opt}}$

$$s.t.c\ \sum_{t=0}^{T} N_{accept} = N_{max} \quad (11)$$

where, $'T'$ is the terminal time-step of each episode. N_{accept} refers to the number of requests accepted by the Fog layer, rwd_{accept} symbolizes the amount of reward received for allocating fog node, VF_{opt} represents the optimal value achieved and N_{max} signifies the number of fog nodes present in the fog layer. $F_{converge}$ refers to the finish time by which all states converge to their optimal value. Hence, the work concentrates on maximizing Fog Utilization and minimizing the convergence time.

FogRA is modeled as MDP to define the mathematical framework of the RL problem. The fog nodes are defined as a set of states in the Markov model, and at each time step, it is in one of the states $(s_0, s_1, s_2, s_3, \ldots\ldots\ldots s_n)$. At any instant of time t = 1,2, …T, the state is represented by s_t. To achieve the objective, the FCA undergoes learning to choose the best action in a current state.

B. Problem Definition

The Fog Controller Agent (FCA) perceives the state of the environment by constant interaction with it. With the continuous stream of application requests, the FCA takes RA decisions regularly. As the current action taken by the FCA, influences the next state and in turn possibility of future actions and rewards, the FogRA problem is formulated as MDP. The MDP to allocate fog nodes for the time-critical IoT requests is defined as MDP_FogRA = $\{S, A, P, R, \gamma\}$ where,

- S = {(m, n_t, c_t)/m=10, and $0 \leq n \leq N_{max}$ and $1 \leq c \leq C$ } is the set of possible states of the MDP. At any instant of time, the state of the fog node is expressed as $\{s_0, s_1, s_2, s_3 \ldots\ldots\ldots s_t \in S\}$ in which,

  m $\in \mathbb{N}$ defines the number of applications considered in the work.

  $n_t \in \mathbb{N}$ is the number of fog nodes utilized at any instant of time $'t'$ bounded by N_{max}.

  $c_t \in \mathbb{N}$ is the time criticality of the incoming request at any instant of time $'t'$. It is a random number in the range.

  of one and ten with C =10 as the highest priority.

- A = {a∈ ('accept', 'decline')} is the action set where 'accept' denotes allocation of fog node, and 'decline' indicates refusal of fog node.

- $P = S \times A \times S \rightarrow [0,1]$ is the probability of transition P $(s'|s,a)$ to a new state s' from state 's' when action 'a' is taken.

- $R = S \times A \rightarrow \mathbb{N}$ denotes the expected reward when the environment is in state 's' and action 'a' is taken.

- $\gamma \rightarrow 0 < \gamma < 1$ is the discount factor that computes the present value of the expected future reward.

Allocation of fog resources to the IoT application requests depends on the number of available resources and the time criticality of the incoming request. The state of the MDP at any instant of time (t) as defined in equation (12),

$$s_t = m.n_t + c_t \quad (12)$$

where $n_t \in \{0,1,2,3\ldots\ldots\ldots N_{max}\}$ is the number of fog nodes utilized/allocated. The current state is based on the number of utilized fog nodes, and the time-criticality of the successive task. The time criticality of the service request arriving at time $'t'$ is denoted by "c_t". It is obtained based on the latency requirement of the incoming request presented in Table I. "c_t" is calibrated by a ten-point scale ranging between 1 and 10, with 10 being the highest priority. Starting with ten applications (m), the total number of states encountered is $m(N_{max}) + C$. The next state s_{t+1} depends on the current state and the action taken, thereby reflecting the Markov property.

TABLE I. LATENCY TIME-CRITICALITY MAPPING

Applications	Latency Requirement in ms	Time Criticality "c_t"
Tactile Internet	<=5	10
Industrial IoT	6-20	9
Smart Vehicles	21-100	8
Collaborative Research/Educational tools	101-150	7
Audio Conference between Patient-Doctor	151-200	6
Internet Chat	201-250	5
Tele Surgery and Tele Ultra Sonography	251-300	4
Video Conference between Patient-Doctor	301-400	3
Web browsing	401-450	2
Automated Notifications	>=10 secs	1

When the fog node is in one of the possible states $s_t \in S = \{1, 2, 3, \ldots.. C\ (N_{max} + 1)\}$, the agent either accepts the incoming task or declines it. Thus, 'A' is the set of actions {accept, decline} where the action 'accept' denotes allocation of a fog resource and 'decline' indicates forwarding the request without allocation of a fog resource. The action chosen is based on the sum of the immediate reward and the value of the possible next state as given in equation (13).

$$action = \begin{cases} accept\ in\ fog, & if \quad [rta_{immediate} + \gamma . V^*(s_{accept})] > [rtd_{immediate} + \gamma . V^*(s_{decline})] \\ decline, & otherwise \end{cases} \tag{13}$$

where the immediate reward is based on the time criticality and intended action as described in the reward system. $rta_{immediate}$ and $rtd_{immediate}$ describes the reward obtained on acceptance and declination respectively. The value of a state denotes the quantity of goodness for the agent to remain in that state. It is determined by its immediate reward and the discounted value of the next possible state. The value of the terminal states is zero.

Similarly, the action from the current state in terms of the state-action pair $Q(s, a)$ is given in equation (14). The choice of the equation for action selection depends on whether the algorithm is MC, MC-ES, or OMC respectively.

$$action = \begin{cases} accept\ in\ fog, & if\ (rta_{immediate} + \gamma Q^*[s, accept]) > (rtd_{immediate} + \gamma Q^*[s, decline]) \\ decline, & otherwise \end{cases} \tag{14}$$

An effective reward system influences the learning ability of an agent. With the right reward as the feedback, the FCA learns better and fast. In FogRA the reward is defined as a function of time criticality, priority threshold, and action taken as shown in Table II. Every incoming request is associated with a time-criticality value (c_t) which is appended in an array. Then the median of the array value is computed in every episode to obtain the Priority threshold (pth_t) at the time 't'. Considering the chance of uneven distribution of c_t values, FogRA chooses the median as the measure of central tendency.

TABLE II. REWARD SYSTEM

Action taken	Time-Criticality	Reward
accept	$c_t \geq pth_t$	pth_t
decline	$c_t \geq pth_t$	$-pth_t$
accept	$c_t < pth_t$	$-pth_t/2$
decline	$c_t < pth_t$	$pth_t/2$

At any instant of time, if the time-criticality of the arrived request (c_t) is greater than the priority threshold (pth_t), then the request is considered to be of high priority. Accepting such a request indicates the right action of the agent. Hence, the agent is encouraged with a positive value of the priority threshold as a reward. Declining the high priority request indicates a wrong action, for which the agent is rewarded with a negative value of pth_t. On the other hand, a time-criticality value less than that of the priority threshold indicates the less time-sensitiveness of the incoming request. Accepting the less time-sensitive task is considered a wrong action anyway, hence the agent is rewarded with $(-pth_t/2)$. Otherwise, the agent is rewarded with $(pth_t/2)$ for declining the less time-sensitive request. Thus, the reward obtained at time 't' depends on the priority threshold computed at that time step.

The probability of state transition (P) and reward distribution (R) describes the dynamics of the environment. When the dynamics of the environment are known, the optimal policy is directly obtained through Dynamic Programming (DP) [8]. Also, DP is bound to fixed values of P, R, and policy π. But, the FogRA environment considered in the work is neither stationary nor policy deterministic. With $m(N_{max}) + C$ number of states and 2 actions per state, the problem of computing the transition probability for all states grows exponentially and cumbersome rather than solving the RA problem itself. Hence, Monte Carlo (MC) method that learns the optimal policy without P and R is sought. Hence, the proposed work does not consider the distribution parameters and employs MC to derive the optimal policy.

C. ACT enabled MC for FogRA

Time criticality (c_t) of an incoming request is defined in the range of one to ten, with ten designated as the highest priority. As Fog resources are limited, only those requests whose time criticality value is greater than the priority threshold is given importance to get processed in fog nodes. But, fixing a suitable priority threshold (pth_t) at every time step is not a trivial task in FogRA whose environment is stochastic and non-stationary. An increased pth_t value leads to poor performance of the FCA, in which the number of accepted requests becomes less than the decline. A decrease in the pth_t results in the allocation of fog nodes to non-time-critical applications, which conflicts with the objective of the system. Hence the work suggests Auto Centric Threshold (ACT) that self-generates the threshold at every time step. The incoming time criticality (c_t) is appended in an array. Then, the priority threshold (pth_t) is computed as the median of the time criticality (c_t) values stored in the array at every time step.

Initially, the FogRA system allocates the fog resource with the priority threshold computed at time t=0. As the episode proceeds, the FCA learns the best policy in which only those requests with a critical value greater than the recently computed priority threshold are allocated the fog resource. The algorithm to learn the optimal policy by the ACT-enabled MC for the FogRA is given in Algorithm 1.

Algorithm 1: ACT MonteCarlo

Initialize $V(s) = 0 \; \forall S$
Initialize return (s) // an empty list to save returns of all state in every iteration
Read N_{max} // *no-of-fog-nodes*
For episode count = 1 ,2, 3, ……do
  While utilized fog node ≤ N_{max}
    Read latency requirement of the incoming request
    Obtain the time-criticality "c_t" of the incoming request from Table 1
    Obtain the state using equation (12)
    Compute the Priority-Threshold (pth_t)
    rta = get reward ('accept', c_t, pth_t)
    rtd = get_reward ('decline', c_t, pth_t)
    Take action on the incoming task based on the equation (13)
    get reward (action, c_t, pth_t)
    Append the reward in the rewards list
    increment the utilized_fog_node by one
  Compute the returns G_t for all states appearing in the episode till the terminal state
  $V(s)$ ← incremental average based on the equation (8)
  If $V(s)$ converges for all states then
    $V^*(s) \leftarrow V(s)$ for all states
    Break
  Endif
Endfor
Use the optimal state value $V^*(s)$ to update the optimal action in equation (13)

The FogRA system is implemented as an episodic task. Once a task arrives, based on its latency requirement, the time-criticality is obtained from the latency time-criticality mapping table. The value of states $V(s)$ is initialized with Zero for all states. Based on equation (13), FCA decides whether to accept the request in the fog or decline it. By continuous interaction with the environment, the FCA updates the state value of all states and derives the optimal policy for every state.

Given the episode, generated by following the policy π (ie., action taken based on equation (13), the ACT-enabled MC optimal policy algorithm evaluates $V_\pi(S_t)$ for all the non-terminal states S_t occurring in that episode. The returns G_t is computed as the sum of undiscounted rewards starting from the initial state till the terminal state of the episode. The returns of every state are used as the target to update the value of the respective state $V_\pi(S_t)$ as given in equation (8). Episodes are generated as long as the state values converge for all states. Once state value converges, the optimal policy is obtained based on equation (7).

IV. Experimental Analysis and Discussion

The MDP for FogRA system adopts the definition of state from adaptive resource allocation but differs in certain aspects [15]. First, the proposed work obtains the time-criticality value (c_t) from the Latency Time-criticality mapping Table I, which maps the preference level of the IoT request with its criticality value. Secondly, as the proposed MDP is episodic, the returns are computed with undiscounted rewards. Thirdly, priority threshold is computed by a self-generated Auto Centric Threshold (ACT) method which suits more for the stochastic nature of the IoT environment. Finally, unlike constant value-based rewards, the novel reward system acts in coherence with the priority threshold and provides the best incentive for the agent to learn better and faster.

The experiment is implemented in python Spyder. The simulation was conducted for the latency generated at random. Then the priority level of the request is obtained from the latency-time criticality mapping Table I. The simulation parameter and their corresponding values are maintained as given in the Table III.

TABLE III. Simulation Parameters

Parameter	Description	Values
α	Learning Rate	0.02
pth_t	Priority Threshold	Median of the time-criticality values
ε	Probability of Random Action	0.2
ϒ	Discount Factor	0.6
C	Highest Priorty level	10
N_{max}	Max.number of Fog Nodes	15

This section analyses the experimental results obtained through MC, MC-ES, and OMC to evaluate the performance of FCA. The work considers ten IoT applications, hence the value of $'m'$ is set to 10 in the work. With the number of fog nodes N_{max} as 15, and the maximum time criticality value C as 10, the MDP leads to $m(N_{max}) + C$ possible states, which are given by $s_t \in S = \{1,2,3, \ldots. m(N_{max}) + C\}$. Initially, at the time t=0, all fog nodes are available, so the number of occupied fog nodes n_t remains zero in the equation (12) with the possible initial state as $s_0 \in \{1,2,3 \ldots . C\}$. The MDP terminates at time T when all the fog nodes are occupied, ie., $n_t = N_{max}$ with the possible terminal states as $s_T \in \{CN_{max} + 1, CN_{max} + 2, CN_{max} + 3 \ldots \ldots . . C(N_{max} + 1)\}$. Hence, the proposed work is experimented with 160 states, with {1, 2, 3, ….10} as one of the possible initial states and {151,152,153……160} as one of the terminal states.

A. Adaptive MC vs ACT MC

First, the performance of FogRA is evaluated with the adaptive MC and the proposed ACT-based MC model. Fig. 4 reflects the value of a few states estimated by the Adaptive MC algorithm in the earlier work [15]. The adaptive model converges after 10000 episodes resulting in the optimal state value 16. Fig. 5 shows that the ACT-based MC derives an optimal state value of 17.5 which is 9.375 % more than the Adaptive MC. The metrics observed in the implementation of Adaptive and ACT-based MC are tabulated in Table IV.

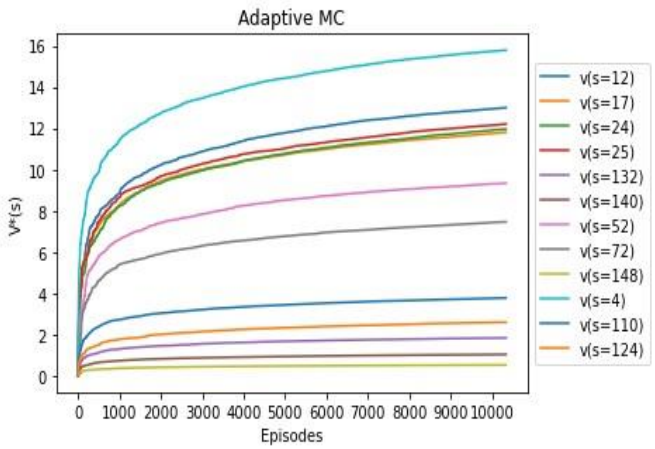

Fig. 4. Optimal State Value in Adaptive RA.

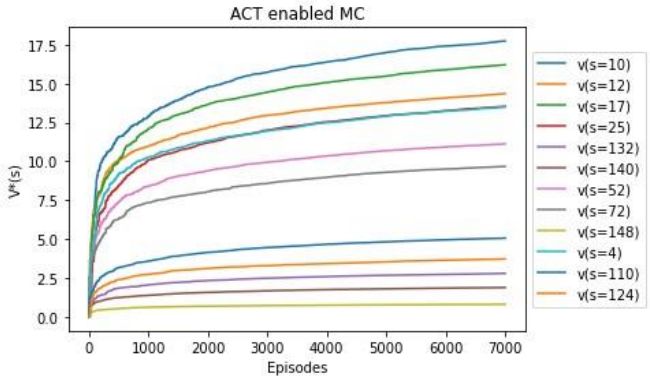

Fig. 5. Optimal State Value with ACT-MC.

The estimation of $V^*(s)$ obtained by the MC algorithm fits more for the model-based MDP. As FogRA is a model-free system, the exploration-oriented ACT MC-ES (Monte Carlo Exploring Starts), is implemented to compare its results with the Adaptive model.

B. Adaptive MC-ES vs ACT MC-ES

The FogRA system is then evaluated in MC-ES for both the Adaptive and ACT model. The ACT-based MC-ES algorithm estimates $Q(s,a)$ rather than $V(s)$. Hence, the state space increases from s to (s x a), and the action selection depends on equation (14). Fig. 6 and Fig. 7 show the convergence of optimal action-value Q*(state, accept) and Q*(state, decline) in Adaptive RA for twelve states chosen at random.

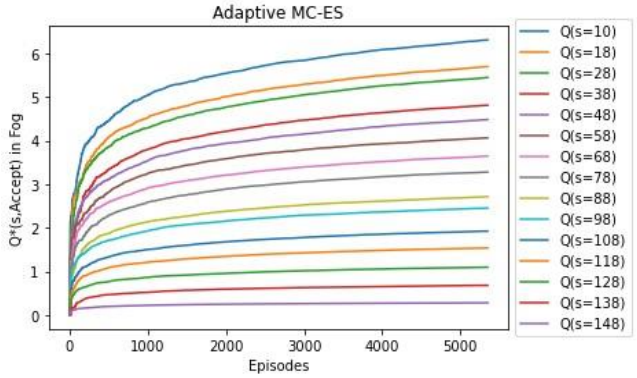

Fig. 6. Q*(s, Accept) for Adaptive MC-ES.

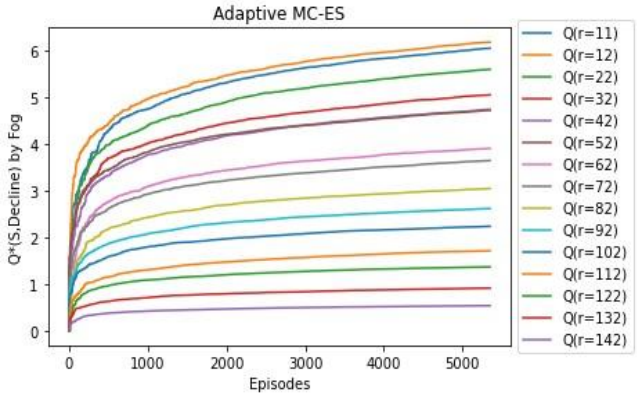

Fig. 7. Q*(s, Decline) for Adaptive MC-ES.

Unlike the Adaptive approach, the values obtained by ACT-based MC-ES in Fig. 8 and Fig. 9 show improved optimal values comparatively. The estimated $Q^*(s,a)$ value in the Adaptive approach is lesser than the ACT model. The reason behind the poor $Q^*(s,a)$ value is that the Adaptive MC-ES is not able to explore widely in an episodic environment with a limited number of fog nodes.

Table IV shows the improved value from 16 in the MC model to 18.7025 in ACT-based MC-ES. The increase in the percentage of 16.8906% indicates the robustness of the ACT-enabled FogRA over the Adaptive system even in an environment with scarce resources. Still, the improvement is not considered favourable as the action selection process of MC-ES is greedy without enough exploration. Hence, the ACT-based FogRA System is tuned towards a balanced mix of exploration and exploitation in the OMC approach.

C. Adaptive OMC vs ACT OMC

The OMC algorithm estimates the value of state-action pair $Q(s,a)$, but the action based on equation (14) is decided only after a balanced trial of exploration and exploitation. FCA controls the level of exploration and exploitation based on a ϵ-greedy policy and is hence named as On-Policy Monte Carlo Control (OMC). The agent explores such that a random action is picked epsilon times and the greedy action based on equation (14) is picked for the (1- ϵ) times. In this case, epsilon (ϵ) is chosen as a small positive number of 0.2 after trial and error. Fig. 10 and Fig. 11 show the convergence of optimal action value $Q^*(state, accept)$ and $Q^*(state, decline)$ for twelve states chosen at random using an Adaptive approach.

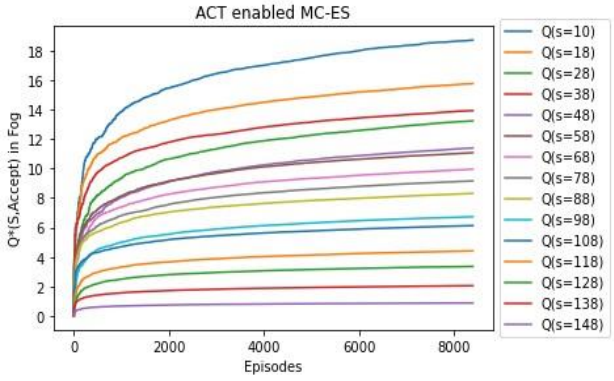

Fig. 8. Q*(s, Accept) for ACT MC-ES.

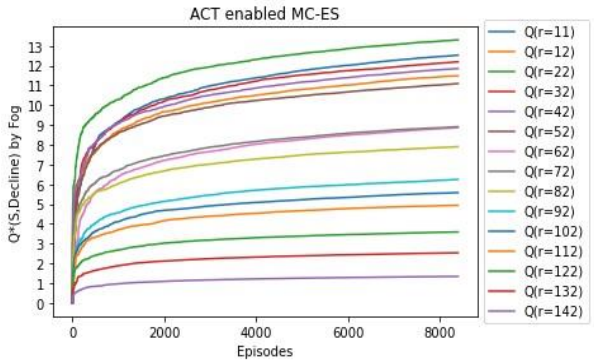

Fig. 9. Q*(s, Decline) for ACT MC-ES.

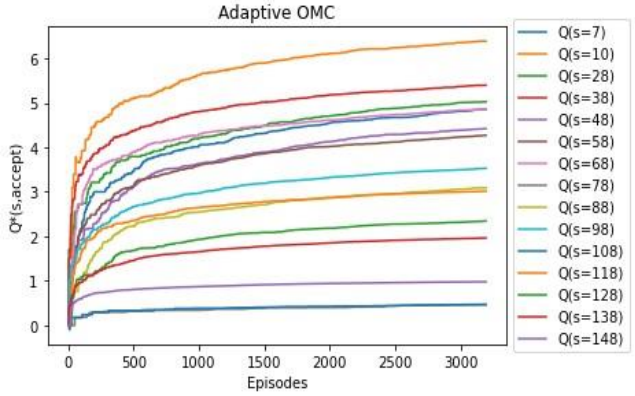

Fig. 10. Q*(s, Accept) for Adaptive OMC.

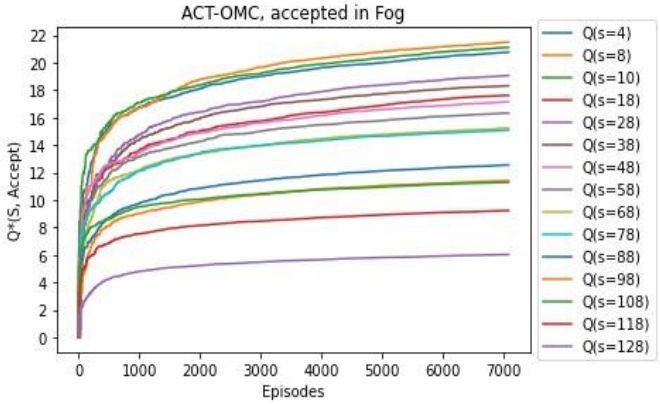

Fig. 12. Q*(s, Accept) for ACT OMC.

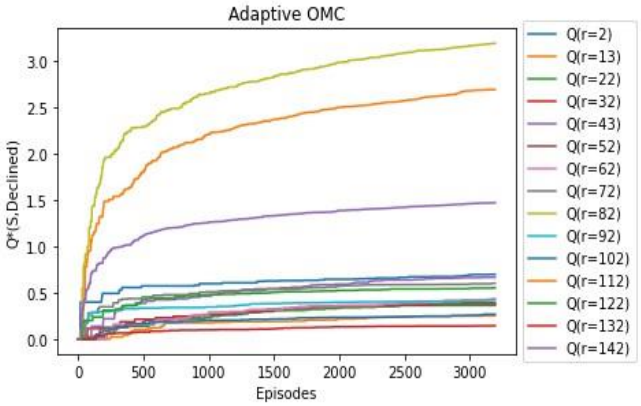

Fig. 11. Q*(s, Decline) for Adaptive OMC.

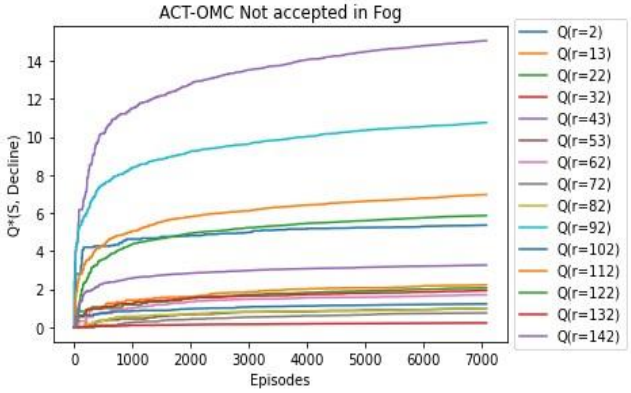

Fig. 13. Q*(s, Decline) for ACT OMC.

Despite its balanced explore-exploit strategy, the adaptive OMC terminates fast, resulting in minimal $Q^*(s,a)$ value. It expresses the inability of the adaptive system for exploration. Whereas the ACT enabled OMC outperforms its counterpart by improved $Q^*(s,a)$ with minimal termination time in Fig. 12 and Fig. 13. The numerical results obtained during execution are tabulated in Table IV.

The optimal state-action value estimated by ACT-based OMC is increased by 41% more than the Adaptive MC. Thus, the ACT-based OMC surpasses other methods by maximizing long-term returns. The high value of the long-term returns with minimum termination time denotes that the FCA has learned the best policy at a fast rate, thanks to the smart learning ability of the ACT-based On Policy Monte Carlo Control.

TABLE IV. PERFORMANCE EVALUATION

Model	Time in ms	Total no. requests accepted	Total no. requests declined	% requests accepted	% requests declined	Total accepted rewards	Total declined rewards	% accepted rewards	% declined rewards	% penalysed episode	% successful episode	V(s), q(s,accept), q(s,decline)	
Adaptive MC	43.05	154215	154673	49.92586	50.0741	308412	154673	66.5994	33.4006	0.855948	99.14405	16	
ACT MC	27.95	104970	72774	59.05685	40.9432	574736.75	201538.25	74.0378	25.9622	0.01429	99.98571	17.5	
Adaptive MC-ES	15.52	81525	80909	50.18962	49.8104	163050	80909	66.835	33.165	0.772769	99.22723	6.5515	6.64784
ACT MC-ES	23.91	130125	90017	59.10957	40.8904	713079	249600	74.0724	25.9276	0	100	18.7025	13.1128
Adaptive OMC	8.77	49755	17563	73.9104	26.0896	36531	2692	93.1367	6.86332	4.492011	95.50799	6.38753	3.5608
ACT OMC	19.19	113745	29342	79.49359	20.5064	305482	10856	96.5682	3.43177	0.316497	99.6835	22.5601	14.05

V. Conclusion and Future Works

In this work, an RL-based ACT-enabled FogRA system is implemented for time-critical applications of IoT. The proposed system instantaneously decides whether to allocate fog resources or not, based on the time-criticality of the incoming request and the availability of fog resources. In the pursuit to derive the optimal policy, the algorithm trains the FCA to learn better decision-making. The RA problem is executed for the incoming requests both on the Adaptive and ACT-enabled FogRA systems. Results show that the ACT-enabled FogRA system prioritizes time-critical applications to allocate fog nodes. Also, utilization of fog resources is maximized in terms of the percentage of requests accepted by fog, rewards obtained and optimal value achieved. The performance assessment of the ACT-enabled FogRA system exhibits an improved optimal state-action value by 41% more than the Adaptive model. In future work, the FogRA system is planned for continuous tasks with the auto-scaling feature of fog resources. Also, it is planned to extend the work as RL enabled Energy Efficient FogRA system that minimizes the energy cost with maximized performance.

References

[1] A. Khanna and S. Kaur, "Internet of Things (IoT), Applications and Challenges: A Comprehensive Review," Wirel. Pers. Commun., vol. 114, no. 2, pp. 1687–1762, Sep. 2020.

[2] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," Concurr. Comput. Pract. Exp., vol. 32, no. 7, Apr. 2020.

[3] L. F. Rahman, T. Ozcelebi, and J. Lukkien, "Understanding IoT Systems: A Life Cycle Approach," Procedia Comput. Sci., vol. 130, pp. 1057–1062, 2018.

[4] M. Mohammed Sadeeq, N. M. Abdulkareem, S. R. M. Zeebaree, D. Mikaeel Ahmed, A. Saifullah Sami, and R. R. Zebari, "IoT and Cloud Computing Issues, Challenges and Opportunities: A Review," Qubahan Acad. J., vol. 1, no. 2, pp. 1–7, Mar. 2021.

[5] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," Concurr. Comput. Pract. Exp., vol. 32, no. 7, Apr. 2020.

[6] F. Murtaza, A. Akhunzada, S. ul Islam, J. Boudjadar, and R. Buyya, "QoS-aware service provisioning in fog computing," J. Netw. Comput. Appl., vol. 165, p. 102674, Sep. 2020.

[7] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions," ACM Comput. Surv., vol. 53, no. 4, pp. 1–43, Jul. 2021.

[8] R. S. Sutton and A. G. Barto, Reinforcement learning: an introduction. Cambridge, Mass: MIT Press, 1998.

[9] M. Faraji Mehmandar, S. Jabbehdari, and H. Haj Seyyed Javadi, "A dynamic fog service provisioning approach for IoT applications," Int. J. Commun. Syst., vol. 33, no. 14, p. e4541, Sep. 2020.

[10] S. Kalantary, J. Akbari Torkestani, and A. Shahidinejad, "Resource discovery in the Internet of Things integrated with fog computing using Markov learning model," J. Supercomput., vol. 77, no. 12, pp. 13806–13827, Dec. 2021.

[11] M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, "Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment," Softw. Pract. Exp., vol. 51, no. 8, pp. 1745–1772, Aug. 2021.

[12] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "Resource provisioning for IoT services in the fog computing environment: An autonomic approach," Comput. Commun., vol. 161, pp. 109–131, Sep. 2020.

[13] A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga, "Resource Management Techniques for Cloud/Fog and Edge Computing: An Evaluation Framework and Classification," Sensors, vol. 21, no. 5, p. 1832, Mar. 2021.

[14] O. Fadahunsi and M. Maheswaran, "Locality sensitive request distribution for fog and cloud servers," Serv. Oriented Comput. Appl., vol. 13, no. 2, pp. 127–140, Jun. 2019.

[15] A. Nassar and Y. Yilmaz, "Reinforcement Learning for Adaptive Resource Allocation in Fog RAN for IoT With Heterogeneous Latency Requirements," IEEE Access, vol. 7, pp. 128014–128025, 2019.

[16] M. K. Pandit, R. N. Mir, and M. A. Chishti, "Adaptive task scheduling in IoT using reinforcement learning," Int. J. Intell. Comput. Cybern., vol. 13, no. 3, pp. 261–282, Jan. 2020.

[17] F. Bahrpeyma, H. Haghighi, and A. Zakerolhosseini, "An adaptive RL based approach for dynamic resource provisioning in Cloud virtualized data centers," Computing, vol. 97, no. 12, pp. 1209–1234, Dec. 2015.

[18] T. V. Maia, "Reinforcement learning, conditioning, and the brain: Successes and challenges," Cogn. Affect. Behav. Neurosci., vol. 9, no. 4, pp. 343–364, Dec. 2009.

[19] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," in 2006 IEEE International Conference on Autonomic Computing, Dublin, Ireland, 2006.

[20] S. Shukla, M. F. Hassan, M. K. Khan, L. T. Jung, and A. Awang, "An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment," PLOS ONE, vol. 14, no. 11, p. e0224934, Nov. 2019.

[21] M. Ghobaei-Arani, S. Jabbehdari, and M. A. Pourmina, "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach," Future Gener. Comput. Syst., vol. 78, pp. 191–210, Jan. 2018.

[22] K. Gai and M. Qiu, "Optimal resource allocation using reinforcement learning for IoT content-centric services," Appl. Soft Comput., vol. 70, pp. 12–21, Sep. 2018.

[23] C. Wang, s. Yuan, k. Shao, and k. Ross, "on the convergence of the monte carlo ex- ploring starts algorithm for reinforcement learning," p. 33, 2022.

[24] "Miguel Morales - Grokking Deep Reinforcement Learning (2020, Manning Publications) - libgen.li.pdf."