BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **23**, No 1
Sofia • 2023 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2023-0002

Type-2-Soft-Set Based Uncertainty Aware Task Offloading Framework for Fog Computing Using Apprenticeship Learning

Bhargavi K.[1], *Sathish Babu B.*[2], *Sajjan G. Shiva*[3]

[1]*Department of CSE, Siddaganga Institute of Technology, Tumakuru, Karnataka, India*
[2]*Department of AI and ML, R V College of Engineering, Bengaluru, Karnataka, India*
[3]*Department of CS, University of Memphis, Memphis, Tennessee, USA*
E-mails: bhargavik@sit.ac.in bsbabu@rvce.edu.in sshiva@memphis.edu

***Abstract*:** *Fog computing is one of the emerging forms of cloud computing which aims to satisfy the ever-increasing computation demands of the mobile applications. Effective offloading of tasks leads to increased efficiency of the fog network, but at the same time it suffers from various uncertainty issues with respect to task demands, fog node capabilities, information asymmetry, missing information, low trust, transaction failures, and so on. Several machine learning techniques have been proposed for the task offloading in fog environments, but they lack efficiency. In this paper, a novel uncertainty proof Type-2-Soft-Set* (*T2SS*) *enabled apprenticeship learning based task offloading framework is proposed which formulates the optimal task offloading policies. The performance of the proposed T2SS based apprenticeship learning is compared and found to be better than Q-learning and State-Action-Reward-State-Action* (*SARSA*) *learning techniques with respect to performance parameters such as total execution time, throughput, learning rate, and response time.*

***Keywords*:** *Task offloading, Uncertainty, Apprenticeship learning, Fog computing, Q-learning, Sarsa learning.*

1. Introduction

Fog computing is an extension of cloud computing in which the computing is brought to edge devices, which is also referred as edge computing. It allows a substantial number of operations related to computation, storage and networking to be performed over the edge devices [1-3]. There exist many tasks which involve long latency for completion and eventually burden the links. Such tasks are offloaded to nearby edge computing devices to achieve lower delay and accelerate the speed of completion of tasks. But fog environments comprises of a lot of uncertainties create hurdles in achieving efficiency during the task offloading process. Some of the potential uncertainties caused in a fog environment are improper identification of the category of tasks, low trust towards the task's load, fluctuating capacity of the computing

nodes, varying demands of the customers, poor task location awareness, unpredictable delay in task assignment, task collision, missing information about the computational fog nodes, information asymmetry, transaction failures, and so on. Among all these uncertainties, improper handling of fluctuating requirements of the tasks and the resources of the fog node resources leads to fog degradation. Hence, there is a necessity to develop the task offloading strategy which is uncertainty proof [4, 5].

Type-2-Soft-Set (T2SS) framework is one of the recent and popular mathematical frameworks which deal with parametric uncertainty by parameterizing already parameterized set of input in dynamic computation environment. The efficiency achieved by T2SS in handling the data imprecision is found to be good compared to other mathematical frameworks like fuzzy set, rough set, probability theory, interval-valued set, hesitant set, and so on [6, 7]. The application of T2SS can be found in solving many decision-making problems in the fields like medical informatics, information science, engineering, environment, economics, finances, business and so on [8-10].

The conventional reinforcement learning algorithms suffer from the problem of reward computation [11, 12].This problem is overcome by the inverse reinforcement learning, which learns to compute reward based on the behavior and policies by experts and is popularly referred to as apprenticeship learning [13, 14]. The reward function is not specified explicitly in apprenticeship learning, instead a set of demonstrations of the experts are mentioned, from which the unknown reward function is recovered [15-17]. The apprenticeship learning agent tries to draw the policy which will achieve performance close to the performance of the expert with minimum training. By integrating the T2SS framework with reliable apprenticeship learning algorithm the uncertainty in the tasks and resources can be handled efficiently. Hence in this paper an apprenticeship learning algorithm powered with the T2SS framework is used to perform offloading of tasks in a fog computing environment.

The objectives of the paper are as follows: handle the uncertainties in the mobile devices, fog nodes, and cloud center resources using T2SS mathematical framework; design a novel T2SS based apprenticeship learning framework for the task offloading in a fog computing environment; expected value analysis of the proposed task offloading framework in finite and infinite fog computing scenarios; evaluate the performance of the proposed task offloading framework in comparison with Q-learning and SARSA learning techniques using artificial dataset, uncertainty of requests and uncertainty of fog nodes.

The remaining sections of the paper are organized as follows; Section 2 discusses related work; Section 3 presents the system model of the proposed task offloading framework; Section 4 discusses the proposed T2SS based apprenticeship learning based framework for the task offloading with supporting algorithms; Section 5 deals with results and discussion by considering the artificial dataset, uncertainty of the tasks and fog nodes; and finally Section 6 draws the conclusion.

2. Related work

Hussein and Mousa [14] present a novel task offloading approach based on ant colony optimization for fog computing. The main reason for going with fog computing is that it improves the Quality of service of the applications which are sensitive foe delay by reducing the latency. The tasks associated with Internet of Things (IoT) applications must be properly distributed across the fog nodes else they suffer from higher response time. Here a hybrid algorithm is developed by combining two nature-inspired algorithms that are ant colony optimization and particle swarm optimization to perform load balancing task. From the experimental results obtained it is observed that the performance of the proposed algorithm is better than the conventional round robin algorithm. However the task offloading policies are of poor quality as they are subjected to premature convergence and the policies get stagnated soon. The IoT nodes produces data which dynamically changes, based on the changing experiment scenarios but the policies formulated are less adaptive. Multiple objectives with respect to power consumption, and cost involved in computation and communication are also ignored.

Adhikari, Mukherjee and Srirama [15] discuss priority and deadline aware task offloading mechanism which is leveraged with multiple levels of queues for fog computing system. IoT applications related to smart city projects put forth the demand for substantial amount of computational resources for processing in real-time environment. Due to the increased physical distance between the IoT devices and cloud server several challenges are imposed which include latency, response time, congestion, and delay. So fog computing demands a fair task offloading technique which determines ideal fog node for processing the incoming tasks. In view of this a deadline and priority dependent task scheduling scheme is proposed for handling computation intensive tasks. For every incoming tasks priority is assigned by considering the deadline and is allocated to suitable multiple feedback queue. The optimal fog node is chosen by considering resource availability and transmission time consumed by the IoT applications. The proposed technique reduces the offloading time of the incoming higher priority tasks by meeting their deadline. However the lower priority task gets postponed for indefinite period of time and suffers from starvation problem. Even the resources of fog nodes gets blocked without being efficiently allocated between the tasks.

Zhang et al. [16] deal with fair task offloading mechanism for fog computing networks. Here an analytical framework is proposed for fair task offloading. During offloading the task delay is reduced by efficiently distributing the tasks among battery-powered fog nodes. For each fog node fairness metric is computed in the fog computing network. For each of the tasks, task delay and energy consumption of the task is computed. The task offloading happens in two stages, in the first step the offloading fog nodes are identified based on the fairness metric. In the second step the tasks is offloaded among the selected fog nodes by using the rules which reduce the task delay incurred. Simulation results provide inference that the performance of the proposed task offloading framework is good pertaining to fairness index of energy consumption and delay incurred in task offloading. But the practical application of the framework is less as it is designed by considering single terminal node and

presence of multiple terminal nodes in fog network is completely ignored. The deployment density of fog nodes is not taken into consideration and fails to achieve proper balance between fog Quality of Service (QoS) and the cost involved in the fog nodes deployment.

Alfakih et al. [17] deal with task offloading procedure embedded with proper allocation of resources using SARSA learning for fog environment.The SARSA learning algorithm is employed to make resource management decisions by offloading the computations that are sensitive towards delay and computation to reliable edge computing devices. The offloading problem is viewed as optimization problem which takes into account two performance metrics that are power consumption and delay involved in task computation. The offloading decisions are made in three different ways that are offloading to the nearest edge server available, offloading to the edge server located at adjacent locations, and last one is offloading to the remotely available cloud server. While making offloading decisions the problems associated with the mobility of mobile devices from one region to another region is handled efficiently. The SARSA learning agent uses epsilon greedy policy for selecting best possible action for the task offloading by gaining maximum possible rewards. The efficiency of the task offloading decisions is analysed by considering many real world applications. The performance of the proposed SARSA learning has been found to be better compared to the conventional reinforcement learning. The adjacent edge devices are effectively utilized for processing the offloaded tasks and offloading decisions are made by taking into account previous workload and current workload in online mode. But with the increase in the number of requests from the mobile devices, the SARSA algorithm easily gets trapped in local minimum solution and accuracy achieved is low.

Rahbari and Nickray [18] present a task offloading mechanism for mobile fog computing using regression tree technique. Heavy task offloading is one of the serious kind of challenge observed in fog computing because of time and energy constraints. The best suitable fog node is identified using regression tree algorithm. The power consumption of mobile devices at initial stage is recorded. If the measured power consumption level is greater than the power consumption of the Wi-Fi then appropriate task offloading decisions are taken. The parameters considered in identifying the best fog nodes are computation cost, operation speed, resource availability, integrity of resources, and user authentication. Module placement method is employed to quickly find the suitable edge nodes for execution of the computation intensive tasks by considering the decision parameters like speed and cost of operations. The output of module placement method is evaluated using the Markov Decision Process (MDP) and the arrival rate of the tasks is assumed to follow Poisson distribution process. The offloading decisions taken by regression algorithm are optimized by applying probability over the network resource utilization. The training and testing phases of regression algorithm are not time consuming as the fog devices arrival rate is analyzed properly using MDP. But the regression tree model used for sampling of tasks in fog environment gets stuck in overfitting problem.

Yao and Ansari [19] discuss the application of Lyapunov based reinforcement learning algorithm for efficient offloading of tasks in fog computing.

The task allocation in fog-based IoT networks is investigated for distributing the task on the fog nodes by efficiently adapting to the varying wireless channel conditions and fog resources. It is possible to predict the nature of incoming next task so online algorithm is designed to make the offloading decisions on fly. As the local information about the tasks is not complete so complete historical data of the task is utilized in mapping task onto appropriate fog nodes. The Lyapunov optimization method creates the virtual queue to process the arrival and departure of the tasks in fog network. Based on Lyapunov optimization over incomplete task information, the reinforcement learning agent takes action over many steps of training to take actions by gaining maximum number of rewards. The derivation of complexity of the proposed optimization technique is performed on theoretical bound. Based on the simulation output produced, it is determined that the technique is able to attain best performance in terms of asymptotic terminologies. However, de facto standard rules are not available for properly defining the Lyapunov optimization function which results in improper trade-off between exploration and exploitation.

To summarize, most of the existing works exhibit the following drawbacks.

• Unable to determine the uncertainty in the mobile device's tasks, fog node, and cloud center resources.

• Drop in the accuracy of task allocation as it fails to handle dynamically changing topology of the fog computing network.

• Conventional reinforcement learning suffers from poor reward computation as the task offloading policies drawn are not near to expert policies.

• Computational complexity of the existing machine learning approaches is high as the number of parameters increase during optimization and frequent overlapping computation states in hidden layers.

• The accuracy of offloading of tasks onto the computation resources is less as the arrival rate of the tasks is assumed to follow Poisson distribution process, but in real-time fog computing scenario the task arrival rate is random.

3. System model

The system consists of *m* mobile devices, *n* number of fog nodes and *o* number of cloud centers:

(1) $$\mathrm{MD} = \{\mathrm{md}_1, \mathrm{md}_2, \mathrm{md}_3, \ldots, \mathrm{md}_m\},$$

(2) $$\mathrm{FN} = \{\mathrm{fn}_1, \mathrm{fn}_2, \mathrm{fn}_3, \ldots, \mathrm{fn}_n\},$$

(3) $$\mathrm{CC} = \{\mathrm{cc}_1, \mathrm{cc}_2, \mathrm{cc}_3, \ldots, \mathrm{cc}_o\}.$$

A fog node is assumed to have finite number of fog resources:

(4) $$\mathrm{fn}_i = \{\mathrm{fr}_i\}_{i=0}^{i=p},$$

where *p* represents the maximum amount of resources contained in a fog node.

A cloud is assumed to have infinite number of resources:

(5) $$\mathrm{cc}_i = \{\mathrm{cc}_i\}_{i=0}^{i=\infty}.$$

A task request from each of the mobile device is identified by a unique identifier and is composed of attributes, namely task arrival time, task deadline, and task size:

(6) $$\mathrm{md}_i = \langle t_{id}, t_{at} = (\mathrm{ta}, \mathrm{td}, \mathrm{ts})\rangle.$$

A fog node is identified by a unique identifier and is composed of attributes namely fog resource speed, fog resource capacity, and fog resource storage:

(7) $$\mathrm{fn}_i(r_i) = \langle \mathrm{fr}_{id}, \mathrm{fr}_{at} = (\mathrm{frs}, \mathrm{frc}, \mathrm{frs})\rangle.$$

A cloud center is identified by a unique identifier and is composed of attributes, namely cloud resource speed, cloud resource capacity, and cloud resource storage:

(8) $$\mathrm{cc}_i(r_i) = \langle \mathrm{cr}_{id}, \mathrm{cr}_{at} = (\mathrm{crs}, \mathrm{crc}, \mathrm{crs})\rangle.$$

A mobile device submits *n* task requests to the task offloading framework to distribute the tasks among the fog nodes:

(9) $$\mathrm{md}_i = \{\mathrm{md}_i(t_1), \ldots, \mathrm{md}_i(t_n)\} \rightarrow \mathrm{fn}_i = \{\mathrm{fn}_i, \ldots, \mathrm{fn}_i(\mathrm{fr}_n)\}.$$

The uncertainty of the tasks in the mobile devices varies between low, medium or high:

(10) $$U(\mathrm{md}_i) = \langle U(\mathrm{md}_i(t_i^l), U(\mathrm{md}_i(t_2^m), \ldots, U(\mathrm{md}_i(t_n^h)\rangle.$$

A fog node contains *n* resources to process the incoming task requests. The uncertainty of the resources in the fog nodes vary between low, medium or high:

(11) $$U(\mathrm{fn}_i) = \langle U(\mathrm{fn}_i(\mathrm{fr}_i^l), U(\mathrm{fn}_i(\mathrm{fr}_2^m), \ldots, U(\mathrm{fn}_i(\mathrm{fr}_n^h)\rangle.$$

The uncertainty of the resources in the cloud centers vary between low, medium or high,

(12) $$U(\mathrm{cc}_i) = \langle U(\mathrm{cc}_i(\mathrm{cr}_i^l), U(\mathrm{cc}_i(\mathrm{cr}_2^m), \ldots, U(\mathrm{cc}_i(\mathrm{cr}_n^h)\rangle.$$

The T2SS is applied over the task requests of the mobile requests to handle the uncertainties in the task requests:

(13) $$\mathrm{T2SS}(\mathrm{md}_i) = \{\mathrm{T2SS}(\mathrm{md}_i(t_1))\cup \mathrm{T2SS}(\mathrm{md}_i(t_2) \cup \ldots \cup \mathrm{T2SS}(\mathrm{md}_i(t_n))\}.$$

The T2SS is applied over the fog nodes to handle the uncertainties in the resources of the fog nodes:

(14) $$\mathrm{T2SS}(\mathrm{fn}_i) = \{\mathrm{T2SS}(\mathrm{fn}_i(\mathrm{fr}_1))\cup \ldots \cup \mathrm{T2SS}(\mathrm{fn}_i(\mathrm{fr}_n))\}.$$

The T2SS is applied over the cloud centers to handle the uncertainties in the resources of the cloud centers:

(15) $$\mathrm{T2SS}(\mathrm{cc}_i) = \{\mathrm{T2SS}(\mathrm{cc}_i(\mathrm{cr}_1))\cup \ldots \cup \mathrm{T2SS}(\mathrm{cc}_i(\mathrm{cr}_n))\}.$$

The Apprenticeship Learning based Task Offloading Agent AL-TOA formulates the *p* task offloading policies after effectively handling the uncertainties in the mobile devices and fog nodes:

(16) $$\prod = \langle \prod_1, \prod_2, \prod_3, \ldots, \prod_p\rangle.$$

4. Proposed approach

The proposed T2SS based apprenticeship learning task offloading framework for fog-computing environment is shown in Fig. 1. The framework contains three functional components: Input Uncertainty Handler (I-UH), Computing Devices Uncertainty Handler (CD-UH), and Apprenticeship Learning based Task Offloading Agent (AL-TOA). The IDUH handles the uncertainties in the mobile device request parameters. The CD-UH handles the uncertainties in the fog nodes resource parameters. After that the AL-TOA operates over the uncertainty free request and resource parameters to formulate optimal task offloading policies.

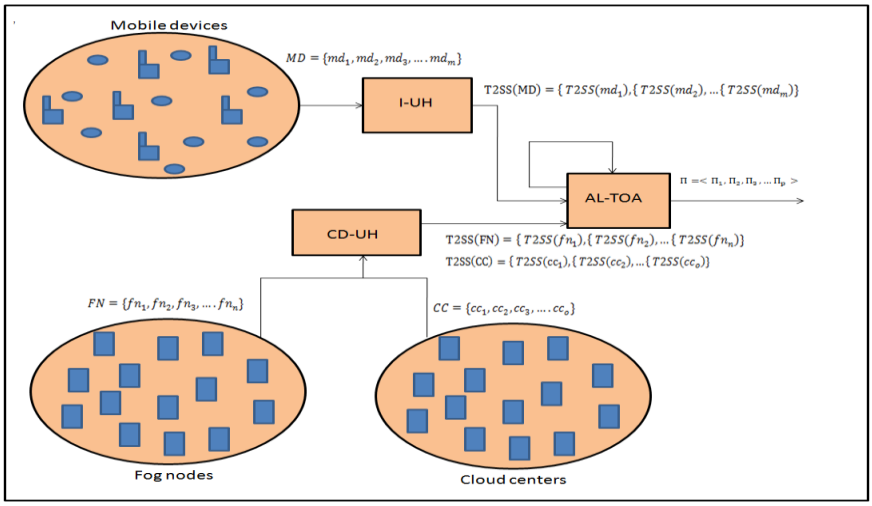

Fig. 1. Proposed T2SS based Apprenticeship learning task Offloading framework

4.1. Input Uncertainty Handler (I-UH)

I-UH functional module of the task offloading framework inputs the mobile device task requests to generate the T2SS based mobile device task requests. The T2SS parameterizes the already parameterized set of mobile device task requests to remove the parameter uncertainties in the mobile device task requests. The working of I-UH functional module is given in Algorithm 1. I-UH functional module is composed of two stages that are training and testing stages. During training stage, for every ordered pair of mobile devices T2SS of mobile devices are gets created. Similarly during testing stage aggregate of T2SS of mobile devices gets computed. Finally the T2SS version of mobile devices composed of tasks is outputted.

Algorithm 1. Working of I-UH

Step 1. Begin

Step 2. *Input* mobile devices

$$\mathrm{MD} = \langle \mathrm{md}_1(t_k), \mathrm{md}_2(t_k), \mathrm{md}_3(t_k), \ldots, \mathrm{md}_m(t_k) \rangle$$

Step 3. *Output* T2SS based mobile devices

$$\mathrm{T2SS-MD} = \langle \mathrm{T2SS}(\mathrm{md}_1(t_k)), \mathrm{TESS}(\mathrm{md}_2(t_k)), \ldots, \mathrm{T2SS}(\mathrm{md}_m(t_k)) \rangle$$

Step 4. Training of I-UH

Step 5. For every training $\mathrm{md}_i \in \mathrm{MD}$ do

Step 6. For every ordered training mobile device pair $(\mathrm{md}_i(t_k), \mathrm{md}_i(t_k)^{\mathrm{at}})$ in MD do

Step 7. $\forall\, \mathrm{md}_i(t_k)^{\mathrm{at}} \in \mathrm{md}^{\mathrm{at}}$ initialize $\sigma(\mathrm{md}_i(t_k)^{\mathrm{at}} = \emptyset$

Step 8. Compute training T2SS of mobile devices

Step 9. $\mathrm{T2SS-MD} = \{\sigma(\mathrm{md}_i(t_k)^{\mathrm{at}}) | \mathrm{md}_i(t_k)^{\mathrm{at}} | \mathrm{md}_i(t_k)^{\mathrm{at}} \in \mathrm{md}^{\mathrm{at}}\}$

Step 10. End For

Step 11. End For

Step 12. Testing of I-UH

Step 13. For every testing $\mathrm{md}_i \in \mathrm{MD}$ do

Step 14. For every ordered testing mobile device pair $(\mathrm{md}_i(t_k), \mathrm{md}_i(t_k)^{\mathrm{at}})$ in MD do

Step 15. Aggregate $\mathrm{T2SS}-\mathrm{MD} :: \mathrm{T2SS}-\mathrm{MD} \cup (\mathrm{md}_i(t_k), \mathrm{md}_i(t_k)^{\mathrm{at}})$

Step 16. End For

Step 17. End For

Step 18. Output $\mathrm{T2SS}-\mathrm{MD} =$
$\langle \mathrm{T2SS}(\mathrm{md}_1(t_k)) \ldots \mathrm{T2SS}(\mathrm{md}_m(t_k)) \rangle$

Step 19. End

4.2. Computing Devices Uncertainty Handler (CD-UH)

The CD-UH functional module of the task offloading framework consists of two sub functional components: T2SS based fog nodes uncertainty handler FN-UH and T2SS based cloud center uncertainty handler CC-UH. The FN-UH inputs the fog node resources to generate the T2SS based fog node resources. The CC-UH inputs the cloud center resources to generate the T2SS based cloud center resources. The main purpose of the algorithm is to apply T2SS framework to parameterize the already parameterized set of fog nodes and cloud center resources to remove the parameter uncertainties in the resources of the fog nodes and cloud centers. The working of CD-UH functional module is given in Algorithm 2. The CD-UH functional module is composed of two stages that are training and testing stages. During training stage, for every ordered pair of training fog nodes T2SS of fog nodes are created and for every pair of cloud centers, T2SS of cloud centers get created. Similarly during testing stage, for every pair of testing fog nodes and cloud centers, enumeration of T2SS of fog nodes and cloud centers are formulated.

Algorithm 2. Working of CD-UH

Step 1. Begin

Step 2. *Input* fog nodes and cloud centers
$\mathrm{FN} = \langle \mathrm{fn}_1(\mathrm{fr}_k), \mathrm{fn}_2(\mathrm{fr}_k), \mathrm{fn}_3(\mathrm{fr}_k), \ldots, \mathrm{fn}_n(\mathrm{fr}_k) \rangle$
$\mathrm{CC} = \langle \mathrm{cc}_1(\mathrm{cr}_k), \mathrm{cc}_2(\mathrm{cr}_k), \mathrm{cc}_3(\mathrm{cr}_k), \ldots, \mathrm{cc}_o(\mathrm{cr}_k) \rangle$

Step 3. *Output*
$\mathrm{T2SS}-\mathrm{FN} = \langle \mathrm{T2SS}(\mathrm{fn}_1(\mathrm{fr}_k)), \mathrm{T2SS}(\mathrm{fn}_2(\mathrm{fr}_k)), \ldots, \mathrm{T2SS}(\mathrm{fn}_n(\mathrm{fr}_k)) \rangle$
$\mathrm{T2SS}-\mathrm{CC} = \langle \mathrm{T2SS}(\mathrm{cc}_1(\mathrm{cr}_k)), \mathrm{T2SS}(\mathrm{cc}_2(\mathrm{cr}_k)), \ldots, \mathrm{T2SS}(\mathrm{cc}_n(\mathrm{cr}_k)) \rangle$

Step 4. Training of FN-UH

Step 5. For every training fog nodes$\mathrm{fn}_i \in \mathrm{FN}$ do

Step 6. For every ordered training fog nodes $(\mathrm{fn}_i(\mathrm{fr}_k), \mathrm{fn}_i(\mathrm{fr}_k)^{\mathrm{at}})$ in FN do

Step 7. $\forall\, \mathrm{fn}_i(\mathrm{fr}_k)^{\mathrm{at}} \in \mathrm{fn}^{\mathrm{at}}$ initialize $\sigma(\mathrm{fn}_i(\mathrm{fr}_k)^{\mathrm{at}} = \emptyset$

Step 8. Compute training T2SS of fog nodes

Step 9. $\mathrm{T2SS}-\mathrm{FN} = \{\sigma(\mathrm{fn}_i(\mathrm{fr}_k)^{\mathrm{at}}) | \mathrm{fn}_i(\mathrm{fr}_k)^{\mathrm{at}} | \mathrm{fn}_i(\mathrm{fr}_k)^{\mathrm{at}} \in \mathrm{fn}^{\mathrm{at}}\}$

Step 10. End For

Step 11. End For

Step 12. Training of CC-UH

Step 13. For every training cloud center $\mathrm{cc}_i \in \mathrm{CC}$ do

Step 14. For every ordered training mobile device pair $(\mathrm{cc}_i(\mathrm{cr}_k), \mathrm{cc}_i(\mathrm{cr}_k)^{\mathrm{at}})$ in CC do

Step 15. $\forall\ \mathrm{cc}_i(\mathrm{cr}_k)^{\mathrm{at}} \in \mathrm{cc}^{\mathrm{at}}$ initialize $\sigma(\mathrm{cc}_i(\mathrm{cr}_k)^{\mathrm{at}} = \emptyset$

Step 16. Compute training T2SS of fog nodes

Step 17. $\mathrm{T2SS} - \mathrm{CC} = \{\sigma(\mathrm{cc}_i(\mathrm{cr}_k)^{\mathrm{at}}) | \mathrm{cc}_i(\mathrm{cr}_k)^{\mathrm{at}} | \mathrm{cc}_i(\mathrm{cr}_k)^{\mathrm{at}} \in \mathrm{cc}^{\mathrm{at}}\}$

Step 18. End For

Step 19. End For

Step 20. Testing of $\mathrm{FN} - \mathrm{UH}$

Step 21. For every testing $\mathrm{fn}_i \in \mathrm{FN}$ do

Step 22. For every ordered testing fog node pair $(\mathrm{fn}_i(\mathrm{fr}_k), \mathrm{fn}_i(\mathrm{fr}_k)^{\mathrm{at}})$ in FN do

Step 23. Aggregate $\mathrm{T2SS} - \mathrm{FN} :: \mathrm{T2SS} - \mathrm{FN} \cup (\mathrm{fn}_i(\mathrm{fr}_k), \mathrm{fn}_i(\mathrm{fr}_k)^{\mathrm{at}})$

Step 24. End For

Step 25. End For

Step 26. Output $\mathrm{T2SS} - \mathrm{FN} = \langle \mathrm{T2SS}(\mathrm{fn}_1(\mathrm{fr}_k)), \ldots, \mathrm{T2SS}(\mathrm{fn}_n(\mathrm{fr}_k)) \rangle$

Step 27. Testing of CC-UH

Step 28. For every testing $\mathrm{cc}_i \in \mathrm{CC}$ do

Step 29. For every ordered testing fog node pair $(\mathrm{cc}_i(\mathrm{cr}_k), \mathrm{cc}_i(\mathrm{cr}_k)^{\mathrm{at}})$ in CC do

Step 30. Aggregate $\mathrm{T2SS} - \mathrm{CC} :: \mathrm{T2SS} - \mathrm{CC} \cup (\mathrm{cc}_i(\mathrm{cr}_k), \mathrm{cc}_i(\mathrm{cr}_k)^{\mathrm{at}})$

Step 31. End For

Step 32. End For

Step 33. *Output* $\mathrm{T2SS} - \mathrm{CC} = \langle \mathrm{T2SS}(\mathrm{cc}_1(\mathrm{cr}_k)), \ldots, \mathrm{T2SS}(\mathrm{cc}_n(\mathrm{cr}_k)) \rangle$

Step 34. End

4.3. Apprenticeship Learning based Task Offloading Agent (AL-TOA)

The AL-TOA functional component of the task offloading framework inputs the T2SS based mobile device task requests, T2SS based fog node resources, and T2SS based cloud center resources to formulate the task offloading policies using apprenticeship learning. The agent keeps the set of expert tasks offloading actions as the basis and mimics the best actions to arrive at best offloading policies. The working of AL-TOA functional module is given in Algorithm 3. The working of AL-TOA functional module is composed of two stages that are training of AL-TOA and testing of AL-TOA. During training of AL-TOA, the Q-values of fog nodes and cloud centers gets updated with best task offloading policies by keeping the expert task offloading policies. During testing of AL-TOA, highest value *Q*-functions gets enumerated to generate global optimal task offloading policies.

Algorithm 3. Working of AL-TOA

Step 1. Begin

Step 2. *Input*

$T2SS - MD = \langle \mathrm{T2SS}(\mathrm{md}_1(t_k)), \mathrm{TESS}(\mathrm{md}_2(t_k)), \ldots, \mathrm{T2SS}(\mathrm{md}_m(t_k)) \rangle$

$T2SS - FN = \langle \mathrm{T2SS}(\mathrm{fn}_1(\mathrm{fr}_k)), \mathrm{T2SS}(\mathrm{fn}_2(\mathrm{fr}_k)), \ldots, \mathrm{T2SS}(\mathrm{fn}_m(\mathrm{fr}_k)) \rangle$

Step 3. *Output* $\prod = \langle \prod_1, \prod_2, . \prod_p \rangle$

Step 4. Initialize the expert actions set $\mathrm{EA} = \{\mathrm{ea}_1 = \phi, \mathrm{ea}_2 = \phi, \mathrm{ea}_3 = \phi, \ldots, \mathrm{ea}_n = \phi\}$

Step 5. Initialize *Q*-values of fog nodes with state S_t expert actions ea_t

$Q_{\text{fn}}(S_t, \text{ea}_t) = \{Q_{\text{fn}_1}(S_t, \text{ea}_t) = \phi, Q_{\text{fn}_2}(S_t, \text{ea}_t) = \phi, \dots, Q_{\text{fn}_n}(S_t, \text{ea}_t) = \phi\}$

Step 6. Initialize *Q*-values of cloud centers with expert actions

$Q_{\text{cc}}(S_t, \text{ea}_t) = \{Q_{\text{cc}_1}(S_t, \text{ea}_t) = \phi, Q_{\text{cc}_2}(S_t, \text{ea}_t) = \phi, \dots, Q_{\text{cc}_n}(S_t, \text{ea}_t) = \phi\}$

Step 7. CALL Training of AL-TOA subroutine

Step 8. STAGE-1: Training of AL-TOA

Step 9. CALL Testing of AL-TOA subroutine

Step 10. STAGE-2: Testing of AL-TOA//

Step 11. Stop

Algorithm 3.1. Working of STAGE-1: Training AL-TOA subroutine

Step 1. Begin

Step 2. For every training $\text{T2SS}(\text{md}_i(t_k)), \text{T2SS}(\text{fn}_i(\text{fr}_k)),$ and $\text{T2SS}(\text{cc}_i(\text{cr}_k))$ do

Step 3. Choose any expert action $\text{ea}_i \in \text{EA}$ and observe the reward $r_i \in R$

Step 4. Compute laxity time of task in fog nodes $\text{Lax}\left(\text{T2SS}(\text{fn}_i(\text{fr}_k))\right)$, i.e.,

$\text{Lax}\left(\text{T2SS}(\text{fn}_i(\text{fr}_k))\right) = D(\text{T2SS}(\text{md}_i(t_k)) - \text{ET}(\text{T2SS}(\text{md}_i(t_k)) - \text{CT}(\text{T2SS}(\text{md}_i(t_k)))$, where $D(\text{T2SS}(\text{md}_i(t_k))$ represent the deadline, $\text{ET}(\text{T2SS}(\text{md}_i(t_k))$ represent execution time, and $\text{CT}(\text{T2SS}(\text{md}_i(t_k))$ represent the completion time

Step 5. Compute laxity time of task in cloud center $\text{Lax}\left(\text{T2SS}(\text{cc}_i(\text{cr}_k))\right)$

$\text{Lax}\left(\text{T2SS}(\text{cc}_i(\text{cr}_k))\right) = D\left(\text{T2SS}(\text{md}_i(t_k)) - \text{ET}(\text{T2SS}(\text{md}_i(t_k)) * \text{pp} - \text{CT}\left(\text{T2SS}(\text{md}_i(t_k))\right) - \text{DS}(\text{T2SS}(\text{md}_i(t_k)))/\text{TR}(\text{T2SS}(\text{md}_i(t_k))\right)$,

where $\text{DS}\left(\text{T2SS}(\text{md}_i(t_k))\right)$ represent the data size, and transmission rate represented by $\text{TR}\left(\text{T2SS}(\text{md}_i(t_k))\right)$

Step 6. IF $\text{Lax}\left(\text{T2SS}(\text{fn}_i(\text{fr}_k))\right) > 0$ &&$\text{Lax}\left(\text{T2SS}(\text{cc}_i(\text{cr}_k))\right) < 0$

Step 7. Update *Q*-value of fog nodes

$Q_{\text{fn}_i}(s_t, \text{ea}_t) = Q_{\text{fn}_i}(s_t, \text{ea}_t) + [\alpha \text{r}_t + \beta \max Q_{\text{fn}_i}(s_{t+1}, \text{ea}_{t+1}) - Q_{\text{fn}_i}(s_t, \text{ea}_t)]$

Step 8. ELSE IF $\text{Lax}\left(\text{T2SS}(\text{fn}_i(\text{fr}_k))\right) = 0$ &&$\text{Lax}\left(\text{T2SS}(\text{cc}_i(\text{cr}_k))\right) = 0$

Step 9. Update *Q*-value of fog nodes

$Q_{\text{cc}_i}(s_t, \text{ea}_t) = Q_{\text{cc}_i}(s_t, \text{ea}_t) + [\alpha \text{r}_t + \beta \max Q_{\text{cc}_i}(s_{t+1}, \text{ea}_{t+1}) - Q_{\text{cc}_i}(s_t, \text{ea}_t)]$

Step 11. ELSE

Step 12. Update *Q*-value of both fog nodes and cloud centers

$Q_{\text{fn}_i}(s_t, \text{ea}_t) = Q_{\text{fn}_i}(s_t, \text{ea}_t) + [\alpha \text{r}_t + \beta \max Q_{\text{fn}_i}(s_{t+1}, \text{ea}_{t+1}) - Q_{\text{fn}_i}(s_t, \text{ea}_t)]$

$Q_{\text{cc}_i}(s_t, \text{ea}_t) = Q_{\text{cc}_i}(s_t, \text{ea}_t) + [\alpha \text{r}_t + \beta \max Q_{\text{cc}_i}(s_{t+1}, \text{ea}_{t+1}) - Q_{\text{cc}_i}(s_t, \text{ea}_t)]$

Step 13. END IF

Step 14. Compute the value function of task offloading

$V(\pi)(s_t, \text{ea}_t) = E(\pi)\{r_t | s_t\}1$

Step 15. Compute the action value function of task offloading

$$\prod_i = \delta + \delta\, Q\left(\arg\max_{\mathrm{ea}_i} Q_{\mathrm{fn}_i}(s_t, \mathrm{ea}_t)\right) + \delta\, Q\left(\arg\max_{\mathrm{ea}_i} Q_{\mathrm{cc}_i}(s_t, \mathrm{ea}_t)\right)$$

Step 16. Output the computed task offloading policy $\prod_i$

Step 17. End For

Step 18. Formulate the policy set $\prod = \prod \cup \prod_i$

Step 19. End

Algorithm 3.2. Working of STAGE-2 Testing AL-TOA subroutine

Step 1. Begin

Step 2. For every training $\mathrm{T2SS}(\mathrm{md}_i(t_k))$, $\mathrm{T2SS}(\mathrm{fn}_i(\mathrm{fr}_k))$, and $\mathrm{T2SS}(\mathrm{cc}_i(\mathrm{cr}_k))$ do

Step 3. For every task offloading policy $\prod_i$ do

Step 4. Enumerate the value function of task offloading function

$$\prod = \prod \cup Q\left(\arg\max_{\mathrm{ea}_i} Q_{\mathrm{fn}_i}(s_t, \mathrm{ea}_t)\right) \cup Q\left(\arg\max_{\mathrm{ea}_i} Q_{\mathrm{cc}_i}(s_t, \mathrm{ea}_t)\right)$$

Step 5. End For

Step 6. End For

Step 7. Output the task offloading polices $\prod = \langle \prod_1, \prod_2, \ldots, \prod_p \rangle$

Step 8. End

5. Results and discussion

This section provides the details of the experimental setup and performance comparisons of ALA with SARSA [17] and QL [18], towards effective task offloading in a fog computing environment.

5.1. Experimental setup

For the experimental evaluation of the proposed task offloading framework ALA iFogSim simulator is used. The iFogSim simulator is one of the popular toolkits used extensively to perform visual modeling and simulation of the task offloading and resource allocation10-15 strategies in fog/edge computing environment. The performance of the proposed offloading strategy is compared with other two well-known offloading strategies like QL and SARSA in three different dimensions artificial dataset, uncertainty of tasks and uncertainty of fog nodes towards the considered performance metrics [19]. Simulation parameters setup are as follows:

- simulation duration = 120-240 s;
- status observation period = 10-15 ms;
- uplink latency (IoT device to nodes = 10-15 ms;
- nodes to cloud= 140-180 ms);
- processing time (client module = 20-60 ms;
- filter module = 10-50 ms;
- analysis module = 100-200 ms;
- event handling module = 20-60 ms);
- deadline to deliver applications = 300-400 ms;
- delay involved in connected to central manager = 45-65 ms;

– delay involved in fog nodes communication = 45-75 ms;
– data receiving frequency = 3-9 s).

5.2. Artificial dataset

The artificial dataset considered for evaluation is composed of 1000 resource intensive tasks, 300 fog nodes, and four different instances. Each of the instances is represented in the form of PQR. In which *P* represent the In-Consistency (IC) in the task distribution, *Q* represent the uncertainty of requests, and *R* represent uncertainty of fog nodes. The value taken for *Q* and *R* is either Low (L) or High (H) which gets represented in the form of ICLL, ICLH, ICHL, and ICHH [19].

5.2.1. Total execution time

A graph of artificial dataset instances versus total execution time is shown in Fig. 2. With respect to ICLL instance, the total execution time incurred by ALA is low; the total execution time incurred by QL is moderate whereas the total execution time incurred by SARSA is too high. With respect to ICLH instance, the total execution time incurred by ALA is too low; the total execution time incurred by QL is moderate whereas the total execution time incurred by SARSA is too high. With respect to ICHL instance, the total execution time incurred by ALA is moderate; the total execution time incurred by QL is high whereas the total execution time incurred by SARSA is moderate. With respect to ICHH instance, the total execution time incurred by ALA is too low; the total execution time incurred by QL and SARSA are moderate.

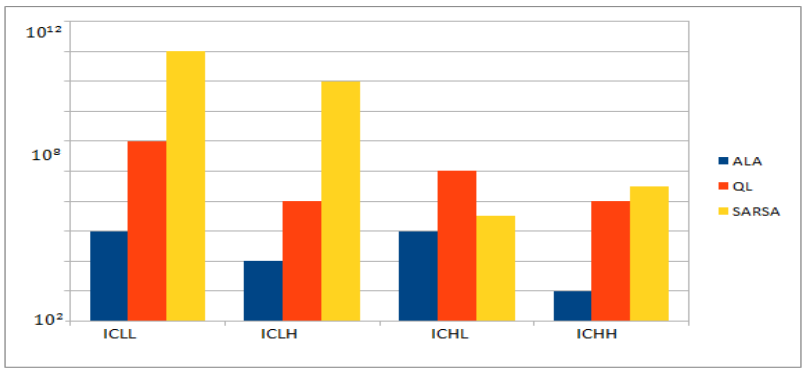

Fig. 2. Artificial dataset instances vs. total execution time

5.2.2. Throughput

A graph of artificial dataset instances versus throughput is shown in Fig. 3. With respect to ICLL instance, the throughput achieved is above average; the throughput achieved by QL and SARSA are moderate. With respect to ICLH instance, the throughput achieved by ALA is too high; the throughput achieved by QL is moderate whereas the throughput achieved by SARSA is too low. With respect to ICHL instance, throughput achieved by ALA is above moderate; the throughput achieved

by QL is moderate whereas the throughput achieved by SARSA is too low. With respect to ICHH instance, the total execution time incurred by ALA and QL is above average; the throughput achieved by SARSA is too low.

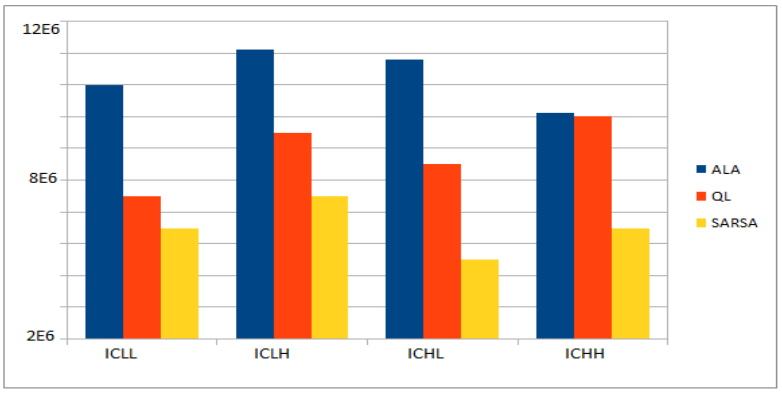

Fig. 3. Artificial dataset instances vs. throughput

5.2.3. Learning rate

A graph of artificial dataset instances versus learning rate is shown in Fig. 4. With respect to ICLL instance, the learning rate of ALA is above average; the learning rate achieved by QL is too low and the learning rate achieved by SARSA is moderate. With respect to ICLH instance, the learning rate attained by ALA and QL are too high; the learning rate attained by SARSA is too low. With respect to ICHL instance, learning rate attained by ALA is high and the learning rate attained by QL and SARSA are moderate. With respect to ICHH instance, the learning rate attained by ALA is high but the learning rate attained by QL and SARSA are below average.

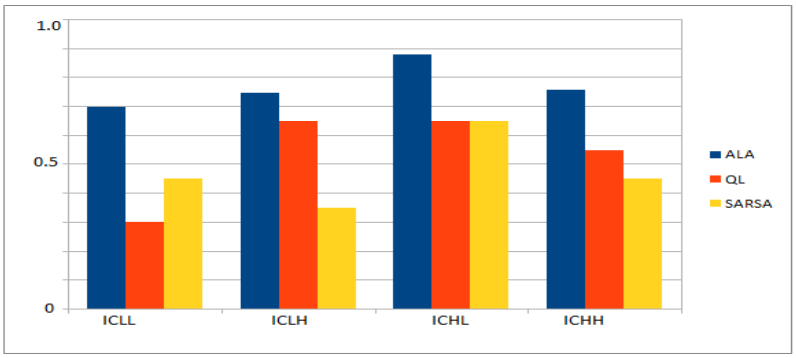

Fig. 4. Artificial dataset instances vs. learning rate

5.2.4. Response time

A graph of artificial dataset instances versus response time is shown in Fig. 5. With respect to ICLL instance, the response time of ALA is moderate; the response time incurred by QL and SARSA are high. With respect to ICLH instance, the response time incurred by ALA is too low and the response time incurred by QL and SARSA

are high. With respect to ICHL instance, the response time incurred by ALA is too low and the response time incurred by QL is too high whereas the response time incurred by SARSA is moderate. With respect to ICHH instance, the response time incurred by ALA is too low, but the response time incurred by QL and SARSA are too high.

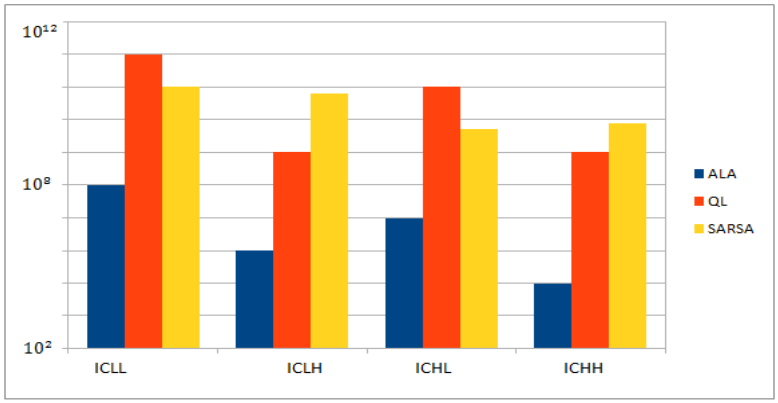

Fig. 5. Artificial dataset instances vs. response time

Considering the artificial dataset with minimum uncertainty among requests and fog nodes, the performance achieved by the proposed ALA framework compared to AL and SARSA is found to be good towards the performance metrics like total execution time, throughput, learning rate, and response time as the reward computation burden is reduced and the chances of task offloading policies getting trapped into suboptimal choices are very less due to efficient balance between exploration and exploitation phases of the learning.

5.3. Uncertainty of requests

The uncertainty of the requests is measured on a scale which ranges between 0 and 1. The uncertainty of the requests are computing by calculating the weighted average of the four vital uncertainty causing factors which includes requests migration, requests fluctuating load, early preemption of requests, and uneven coupling of the requests.

5.3.1. Total execution time

A graph of uncertainty of requests versus total execution time is shown in Fig. 6. During low uncertainty level of requests, the total execution time incurred by ALA is low, the total execution time incurred by QL is moderate, and the total execution time incurred by SARSA is too high. During moderate uncertainty level of requests, the total execution time incurred by ALA still remains to be low and the total execution time incurred by QL and SARSA are high. During high uncertainty level of requests, the total execution time incurred by ALA still remains to be low and the total execution time incurred by QL and SARSA are moderate.

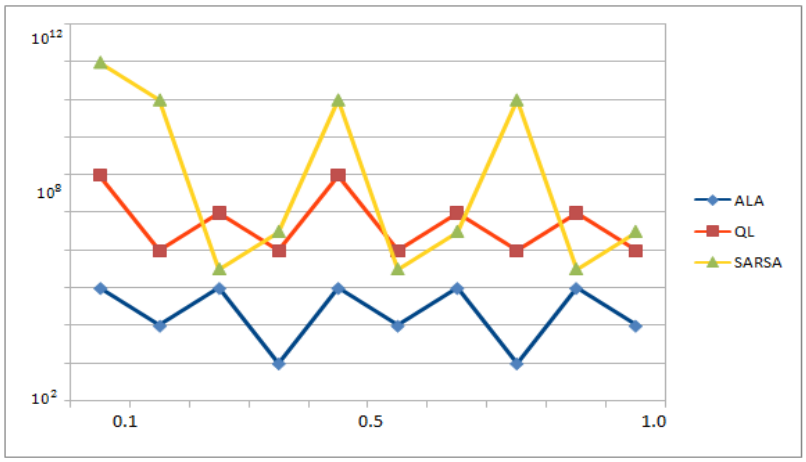

Fig. 6. Uncertainty of requests vs. total execution time

5.3.2. Throughput

A graph of uncertainty of requests versus throughput is shown in Fig. 7. During low uncertainty level of requests, the throughput achieved by ALA is high, the throughput achieved by QL is moderate, and the throughput achieved by SARSA is too high. During moderate uncertainty level of requests, the throughput achieved by ALA high, the throughput achieved by QL is too low, and the throughput achieved by QL is moderate. During high uncertainty level of requests, the throughput achieved by ALA is high, the throughput achieved by QL is too low, and the throughput achieved by QL is moderate.

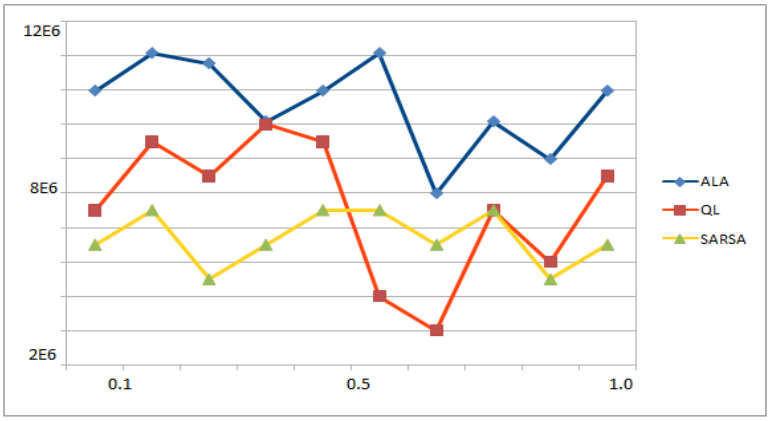

Fig. 7. Uncertainty of requests vs. throughput

5.3.3. Learning rate

A graph of uncertainty of requests versus learning rate is shown in Fig. 8. During low uncertainty level of requests, the learning rate attained by ALA is high, but the learning rate attained by QL and SARSA are low. During moderate uncertainty level

of requests, the learning rate attained by ALA is above moderate but the learning rate attained by achieved by QL and SARSA are too low. During high uncertainty level of requests, the learning rate attained by ALA is above moderate, whereas the learning rate attained by QL and SARSA are too low.

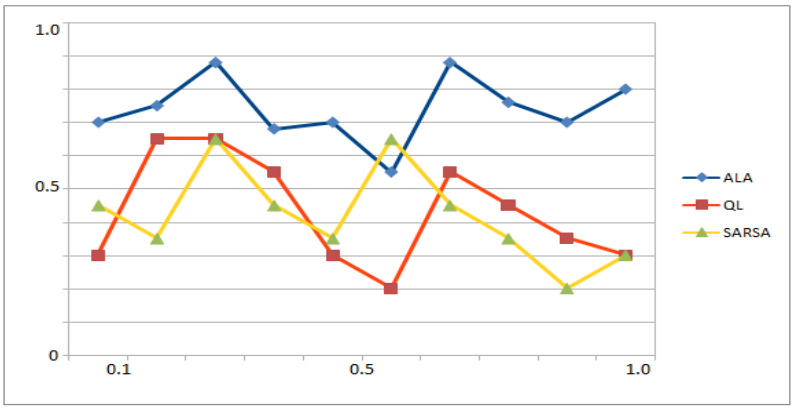

Fig. 8. Uncertainty of requests vs. learning rate

5.3.4. Response time

A graph of uncertainty of requests versus response is shown in Fig. 9. During low uncertainty level of requests, the response time incurred by ALA is moderate, but the response time incurred by achieved by QL and SARSA are high. During moderate uncertainty level of requests, the response time incurred by is too low, the response time incurred by QL and SARSA are too high. During high uncertainty level of requests, the response time incurred by ALA is moderate, the response time incurred by QL and SARSA are too high.

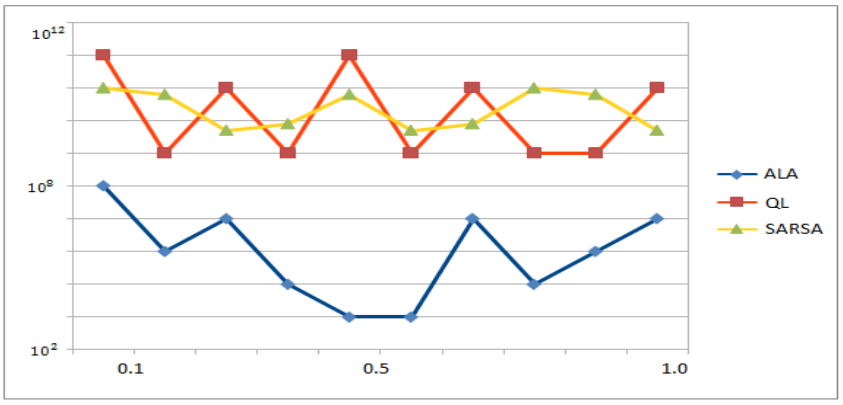

Fig. 9. Uncertainty of requests vs. response time

The performance of the proposed ALA framework is evaluated by considering artificial dataset containing uncertainty of requests. The factors causing uncertainty in tasks include fluctuating tasks requests, improper placement of fog nodes, uneven

coupling of the requests, and early pre-emption of requests. The performance achieved by the proposed ALA framework is found to be good compared to QL and SARSA towards the performance metrics like total execution time, throughput, learning rate, and response time as the apprenticeship learning agent effectively uses the newly acquired domain information to override the old policies to form optimal task scheduling policies.

5.4. Uncertainty of fog nodes

The uncertainty of the fog nodes is measured on a scale which ranges between zero and one. The uncertainty of the fog nodes are computing by calculating the weighted average of the four vital uncertainties causing factors which includes fog nodes processing speed, fog nodes storage, fog nodes placement, and load on fog nodes.

5.4.1. Total Execution Time

A graph of total execution time versus uncertainty of fog nodes is shown in Fig. 10. During low uncertainty level of fog nodes, the total execution time incurred by ALA is low, the total execution time incurred by QL is moderate, and the total execution time incurred by SARSA is too high. During moderate uncertainty level of fog nodes, the total execution time incurred by ALA still remains to be low and the total execution time incurred by QL is too high and the total execution time incurred by SARSA is moderate. During high uncertainty level of fog nodes, the total execution time incurred by ALA still remains to be low, the total execution time incurred by QL is very high and the total execution time incurred by SARSA is moderate.

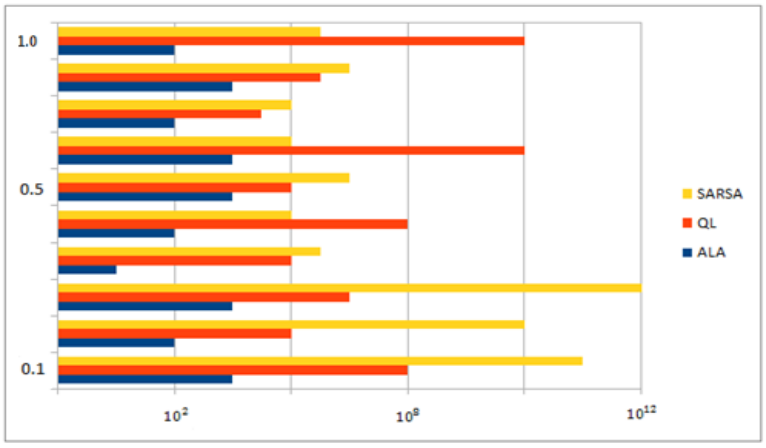

Fig. 10. Total execution time vs. fog nodes uncertainty

5.4.2.Throughput

A graph of throughput versus throughput is shown in Fig. 11. During low uncertainty level of fog nodes, the throughput achieved by ALA is high and the throughput achieved by the QL and SARSA are low. During moderate uncertainty level of fog

nodes, the throughput achieved by ALA still remains to be high, the throughput achieved by QL is too low, and the throughput achieved by QL is moderate. During high uncertainty level of fog nodes, the throughput achieved by QL is too low by ALA still remains to be high, the throughput achieved by QL is moderate and the throughput achieved by SARSA is low.

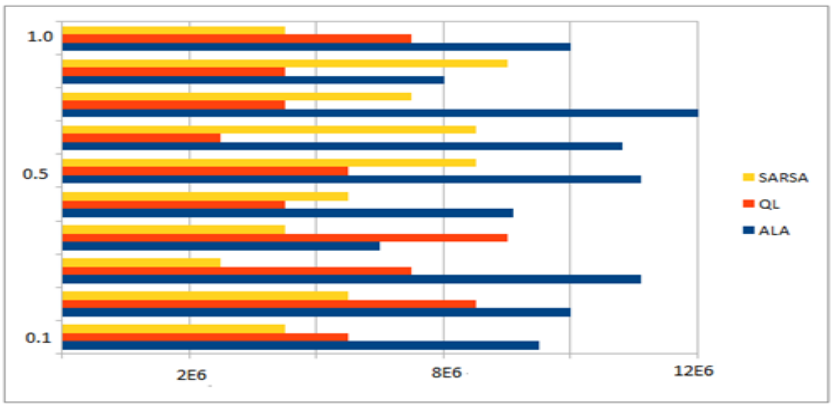

Fig. 11. Throughput vs. fog nodes uncertainty

5.4.3. Learning rate

A graph of learning rate versus uncertainty of fog nodes is shown in Fig. 12. During low uncertainty level of fog nodes, the learning rate attained by ALA and QL are high, and the learning rate attained by SARSA is low. During moderate uncertainty level of fog nodes, the learning rate attained by ALA still remains to be high and the learning rate attained by QL and SARSA are too low. During high uncertainty level of fog nodes, the learning rate attained by ALA still remains to be high, the throughput achieved by QL and SARSA are low.

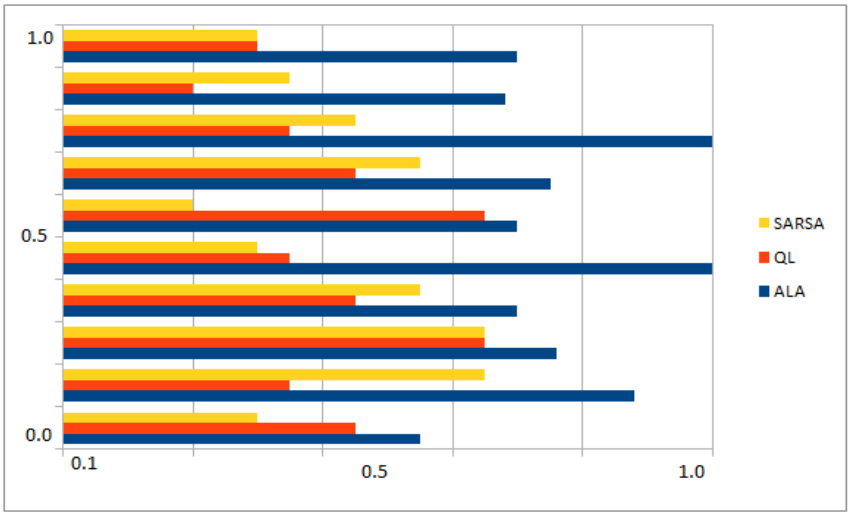

Fig. 12. Learning rate vs. fog nodes uncertainty

5.4.4. Response time

A graph of throughput versus response time is shown in Fig. 13. During low uncertainty level of fog nodes; the response time incurred by ALA is moderate and the response time incurred by QL and SARSA are high. During moderate uncertainty level of fog nodes; the throughput achieved by ALA still remains to be low, the throughput achieved by QL is too high, and the response time incurred by achieved by SARSA is too high. During high uncertainty level of fog nodes; the response time incurred by ALA is moderate, but the response time incurred by QL and SARSA are high.

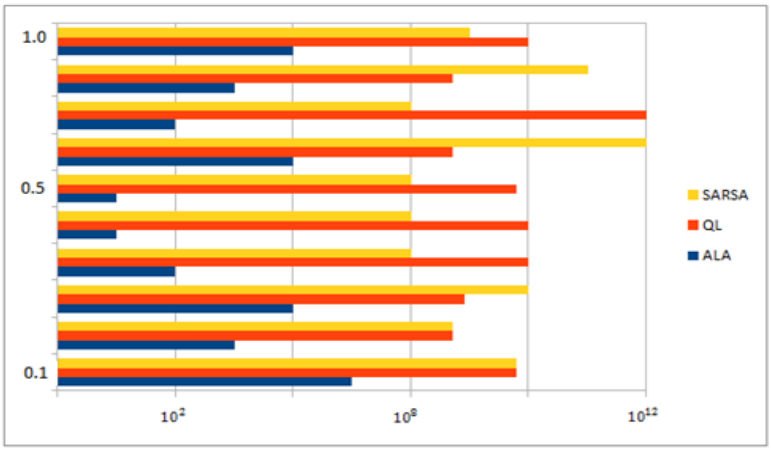

Fig. 13. Response time vs. fog nodes uncertainty

The performance of the proposed ALA framework is evaluated by considering artificial dataset containing uncertainty of fog nodes. The factors causing uncertainty of fog nodes include fluctuating load towards fog nodes, lower trust towards fog nodes storage, and inefficient processing speed of the fog nodes. The performance achieved by the proposed ALA framework is found to be good compared to QL and SARSA towards the performance metrics like total execution time, throughput, learning rate, and response time as the successful reward computation ratio is high even the state space of requests and fog nodes are larger.

6. Conclusion

The paper presents a novel T2SS based apprenticeship learning framework for the task offloading in fog computing which formulates optimal offloading policies without the need for specifying reward function explicitly. Expected value analysis of the proposed apprenticeship learning based framework is done by considering QL and SARSA techniques in finite and infinite fog computing scenarios. The experimental evaluation of the proposed framework is carried out using iFogSim simulator by considering three different dimensions including artificial dataset, uncertainty of the tasks and uncertainty of the fog nodes. The proposed

apprenticeship learning framework outperforms the QL and SARSA with respect to performance metrics like total execution time, throughput, learning rate, and response time. However practical implementation of the proposed approach is difficult because of its complex structure and properties. It is also subjected to frequent failures while solving large scale uncertainty of continuity among edge devices in fog computing environment.

References

1. Bonomi, F., R. Milito, J. Zhu, S. Addepalli. Fog Computing and Its Role in the Internet of Things. – In: Proc. of 1st Edition of the MCC Workshop on Mobile Cloud Computing, 2012, pp. 13-16.
2. Wen, Z., R. Yang, P. Garraghan, T. Lin, J. Xu, M. Rovatsos. Fog Orchestration for Internet of Things Services. – IEEE Internet Computing, Vol. **21**, 2016, No 2, pp. 16-24.
3. Manzali, Y., M. El Far, M. Chahhou, M. Elmohajir. Enhancing Weak Nodes in Decision Tree Algorithm Using Data Augmentation. – Cybernetics and Information Technologies, Vol. **22**, 2022, No 2, pp. 50-65.
4. Jeaunita, T. J., V. Sarasvathi. A Multi-Agent Reinforcement Learning-Based Optimized Routing for QoS in IoT. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 4, pp. 45-61.
5. Toshev, A. Particle Swarm Optimization and Tabu Search Hybrid Algorithm for Flexible Job Shop Scheduling Problem-Analysis of Test Results. – Cybernetics and Information Technologies, Vol. **19**, 2019, No 4, pp. 26-44.
6. Maji, P. K., R. Biswas, A. Roy. Soft Set Theory. – Computers & Mathematics with Applications, Vol. **45**, 2003, No 4, pp. 555-562.
7. Zhang, Z., S. Zhang. Type-2 Fuzzy Soft Sets and Their Applications in Decision Making. – Journal of Applied Mathematics, 2012.
8. Zhang, Z., S. Zhang. A Novel Approach to Multi Attribute Group Decision Making Based on Trapezoidal Interval Type-2 Fuzzy Soft Sets. – Applied Mathematical Modelling, Vol. **37**, 2013, No 7, pp. 4948-4971.
9. Alcantud, J. C. R. Some Formal Relationships among Soft Sets, Fuzzy Sets, and Their Extensions. – International Journal of Approximate Reasoning, Vol. **68**, 2016, pp. 45-53.
10. Moreno, J. E., M. A. Sanchez, O. Mendoza, A. Rodriguez-Diaz, O. Castillo, P. Melin, J. R. Castro. Design of an Interval Type-2 Fuzzy Model with Justifiable Uncertainty. – Information Sciences, Vol. **513**, 2020, pp. 206-221.
11. Abbeel, P., A. Y. Ng. Apprenticeship Learning via Inverse Reinforcement Learning. – In: Proc. of 21st International Conference on Machine Learning, 2004, 1.
12. Szepesvari, C. Algorithms for Reinforcement Learning. – Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. **4**, 2010, No 1, pp. 1-103.
13. Al-Quran, A., N. Hassan, E. Marei. A Novel Approach to Neutrosophic Soft Rough Set under Uncertainty. – Symmetry, Vol. **11**, 2019, No 3, 84.
14. Hussein, M., M. Mousa. Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization. – IEEE Access, Vol. **8**, 2020, pp. 37191-37201.
15. Adhikari, M., M. Mukherjee, S. Srirama. DPTO: A Deadline and Priority-Aware Task Offloading in Fog Computing Framework Leveraging Multilevel Feedback Queueing. – IEEE Internet of Things Journal, Vol. **7**, 2020, pp. 5773-5782.
16. Zhang, G., F. Shen, Y. Yang, H. Qian, W. Yao. Fair Task Offloading among Fog Nodes in Fog Computing Networks. – In: Proc. of IEEE International Conference on Communications (ICC'18), 2018, pp. 1-6.
17. Alfakih, T., M. M. Hassan, A. Gumaei, C. Savaglio, G. Fortino. Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA. – IEEE Access, Vol. **8**, 2020. pp. 54074-54084.

18. R a h b a r i, D., M. N i c k r a y. Task Offloading in Mobile Fog Computing by Classification and Regression Tree. – Peer-to-Peer Networking and Applications, Vol. **13**, 2020, No 1, pp. 104-122.
19. Y a o, J., N. A n s a r i. Task Allocation in Fog-Aided Mobile IoT by Lyapunov Online Reinforcement Learning. – IEEE Transactions on Green Communications and Networking, Vol. **4**, 2019, No 2, pp. 556-565.

Received: 11.08.2022; Second Version: 14.12.2022; Accepted: 28.12.2022