International Journal of Advances in
Soft Computing and Intelligent Systems (IJASCIS)
2022, Vol 01, No 01, 13-22

Science Transactions © 2022

Original Paper

COMPUTATION TASK OFFLOADING IN MOBILE EDGE COMPUTING USING DEEP LEARNING

Shital R. Langote[1] and Tanuja S. Dhope(Shendkar)[2]

[1] Department of Electronics and Telecommunication, JSPM's Rajarshi Shahu College of Engineering, Tathawade,Pune, India
[2] Department of Electronics and Communication, Bharati Vidyapeeth (Deemed to be university) College of Engineering,Pune,India
[1]langotesh97@gmail.com , [2]tanuja_dhope@yahoo.com

ABSTRACT

Today, in the rapidly evolving world of technology and the internet, no matter where you go, you expect computationally intensive tasks to be completed with less power consumption and minimum delay. We can do this by expanding cloud computing capabilities as far as possible from the cell tower, mobile edge computing (MEC) enables mobile computing near the mobile devices. As a result of MEC, procedures may be carried out at core network, centralized headquarters, and various network aggregation points. In this paper, We used deep learning algorithm to take offloading decisions when there are multiple tasks executing simultaneously in a cellular network or on one user equipment (UE).This deep learning algorithm uses Q learning method and helps for better resource management in MEC. A proposed algorithm analyzes CPU utilization requirements for a particular task to decide which task must be delegated to the edge server, which minimizes the power consumption and delay required for execution. The tasks may include IOT applications, image recognition, image processing, interactive gaming, web browser, video, smart phones, tablets, robots, drones etc.

KEYWORDS

Computational intensive tasks, Edge server, Q learning algorithm, Task offloading, Deep learning, Mobile edge computing.

1. INTRODUCTION

In its earliest form, MEC originated from an ETSI (European Telecommunications Standards Institute) initiative which was meant to place the cellphone network's edge terminals. In recent years, these have encompassed to include the fixed-line telecommunications infrastructure as well. It expands the functionality of cloud computing in order to get it to the network's edge. Multi-Access Edge Computing (MEC) is also termed mobile edge computing. MEC's capabilities are different from traditional cloud computing because the network can aggregate processes in locations close to the user and device, unlike remote cloud servers. In addition to reducing overcrowding in cellular infrastructure and lowering delay, MEC enhances user quality of experience (QoE) by shifting cloud computing to local servers.

Computations are offloaded to a distant server by transferring computationally expensive system elements to a remote [2]. MCC, emerging from cloud computing, was developed to meet the computing needs of new smartphone-based applications. If a remote computing job should be outsourced to the MEC server, or performed natively just on equipment, every smartphone has the option to select. A comparison of energy prices is used to make this choice [3]. In today's world, computing offloading discusses both boosting Smartphone performance as well as attempting to guarantee energy savings simultaneously [6]. Although meeting the severe

delay prerequisites, MEC allows the edge to perform computation-intensive applications rather than user equipment. The research group is addressing the accompanying problems in MEC computation offloading: offloading decision taking, wireless distribution of resources, and computing resource allocation [7]. Furthermore, IoT users will engage in detecting and processing tasks later in 5G networks, which will be user-centric. Cloud-based services or edge computing services are required to be used for computation-extensive activities [8]. In practise, offloading computation activities by means of MEC causes information transmission through wireless links. If a large number of application stations forcefully offload their computational resources to the edge node, a severe blockade on wireless connections is possible, causing MEC to be delayed significantly. To take advantage of compute offloading, we'll require a combined management system for computation offloading and the related wireless resource distribution, which really had to be pulled together in consideration of a great deal of analyst [10].

While offloading tasks to edge servers, there are many questions like whether to offload the task to an edge server and if the answer is yes, then to which edge server should offload the task. To answer the above question, it is required to analyse the load level of mobile edge servers [18]. The task/data offloading decision is critical because it is expected to have a straightforward influence on the user application's Quality of Service (QoS), including the associated delay introduced by the offloading process [11]. When the load at the edge node is heavy as a result of an incredibly large number of user gadgets using the similar edge network for all of their jobs at the same time, it may trigger significant processing delays and the drop-off of certain tasks [18]. The MEC idea's reasoning architecture is used for getting cloud computing applications nearer to smartphone users by locating multiple information centres at the network's node. The network terminal can apply to a variety of locations, interior sites such as Wi-Fi and 3G/4G wireless routers, or the cell infrastructure at the Cell Tower [11]. MEC is a decent stage that can be used for various situations and purposes, such as offloading smartphone software, the Internet of Things (IoT), and as an edge caching agency [12]. The decision to offload computations in a MEC setting should be improved and controlled using the concepts of the Optimal Stopping Theory (OST) [14]. While mobile users may be able to connect to a faraway cloud from their mobile devices, there may be considerable connection delays during the actual data. There is a resulting decrease in QoS, particularly those activities that must adhere to stringent schedules, such as workflow applications (WA). If the propagation delay is too high, the job will not be completed [15].

The Markov decision method (MDP) is a fundamental hypothesis of reinforcement learning that is normally utilised by scientists. MDP is a complex programming approach that can be used to find the best value [16]. The most recent type of SDN has the basic capacity to remove the obstacles that hinder data processing at each node from reaching its maximum capabilities. To limit resources, money, and reduce latency, distributed computing must be done in a centralised network. [17] suggests a new software defined edge cloudlet (SDEC) structure enabling resource distribution in MEC based on the incorporation of SDN and MEC. Only a fraction of service programmes can be cached by the edge server because of the restricted storage capacity (WDs) [19]. Computation offloading is useful because the entire cost of offloading computations is less than that of whole cost of local computing. For users to settle computation offloading choices, beneficial computation offloading is needed [20]. A Markov decision process (MDP) is used to demonstrate the job offloading decision process, that has considerably a discrete time stochastic control process [21]. Using 5G to connect billions of smart devices is expected to deliver extended coverage, greater throughput, decreased latency, and high network connection density [22]. Due to the fact that MEC has restricted computing capacity than central MCC, it is critical to distribute these infrastructures productively. With the recommended offloading decision-based state-action-reward-state-action (ODSARSA) using reinforcement

learning resource allocation, meeting quality of service (QoS) criteria (e.g., latency) would be possible with limited effort [23].

This paper describes the work done related to task offloading in mobile edge computing in section 2. Section 3 describes the system model as local computing, edge computing model, Q learning algorithm for task offloading, and plots for different task offloading algorithms. Section 4 presents the conclusion based on our results.

2. RELATED WORK

In [10] Liang Huang et al., limited the total cost, which included the overall time it took to complete the users' activities, energy utilization, and the cost of the edge server's usefulness. They used a DQN-based methodology, modeling all potential responses as state spaces and travel between states as behavior, to take advantage of DQN's advantages. Broad mathematical assessments revealed that their proposed algorithm had a steady combination execution and offered a solution that gave a near ideal arrangement. Md. Sajjad Hossain et al., in [16] For task offloading, RL was suggested as a Q-Learning based multi-user framework. The key objective of these research is to restrict the processing latency and power consumption of all customers in the node handling framework in IIoT organizations. For the IIoT environment, a network architecture model is provided as a weighted amount of power utilization and process execution latency for radio sensors nearby the node server. Nahida Kiran et al., in [17] By considering a novel software defined edge cloudlet (SDEC) based RL optimization architecture to handle the power reducing issue, we tackled the resource distribution issue in wireless MEC by considering the chance of Software Defined Network(SDN) powered MEC operation. Results showed that the suggested paradigm outperformed other standard approaches with regards to saving the battery power of a consumer computer using RL-based Q-learning and cooperative Q-learning answers for the immovable issue.

Ming Tang et al., in [18] To limit the estimated long-term cost, a task offloading problem was devised, considering the load level dynamics at the edge nodes (considering the latency of the process and the punishments for those tasks being dropped). They utilized queuing mechanisms to show the compilation and transmission processes of the tasks in this issue since they were non-distinct and delay-sensitive tasks. To accomplish the anticipated long-term cost minimization when accounting, in order to deal with unpredictability in load dynamics at the network edge, a model-free DRL-based distributed, which allows each cellular device to settle on its own transferring choice without understanding the mission system or transferring choices of other cellular devices. They used the long short-term memory (LSTM), dueling deep Q-network (DQN), and double-DQN methodologies to enhance the calculation of the predicted long-term expense in the suggested paradigm. In [21], Bingxin Zhang et al. proposed an RL-based MDP to solve the computation task offloading and energy management issue in multi-UE and multi-RRH MEC systems. The proposed RL-based algorithm will accomplish close ideal machine efficiency as contrasting to the ES algorithm. While managing a huge scope network, the proposed RL based algorithm can accomplish great execution regardless of in the event that it is from the perspective of the machine or a person. Taha Alfakih et al., in [23] suggest a reinforcement-learning-based SARSA approach to take care of the optimization issue of deciding whether to offload to the closest edge server, neighboring edge server, or remote cloud to minimise device costs, such as energy utilization and processing time delay. On this issue, it was exhibited that OD-SARSA outperformed RL-QL. As a result, by offloading to neighboring edge servers, the proposed strategy addresses the majority of CPSS challenges and accomplishes optimum volume, variety, velocity, and veracity outcomes.

Fengxian Guo et al., in [7] it was studied if an energy-efficient computation offloading management strategy could be devised for a MEC infrastructure consisting of several SBSs (small cell base stations) serving multiple users while taking interruption migration into consideration. They begin by presenting the computation offloading system and formulating the issue as an NP-hard mixed integer non-linear programming (MINLP) problem. We devise a suboptimal genetic algorithm (GA) based computation algorithm to solve this problem (GACA). At last long, simulation is used to investigate this algorithm's combination, and the proposed algorithm's success is checked by contrasting it to other baseline algorithms. Yeongjin Kim et al., in [9], the energy-proficient computing offloading issue was investigated in a mobile computer with LTE and Wi-Fi interfaces that was running applications. By changing the Lyapunov optimization approach, a dynamic control paradigm in a mobile computing offloading device is proposed to optimise energy resource performance under various functional constraints. Not only does the power utility efficiency capture the power efficiency of cell phones, but it can also capture throughput fairness amongst applications.

Ibrahim Alghamdi et al., in [11], It would be the first time that an offloading method for choosing a MEC server was viewed as an OST issue in the form of a user travelling between various MEC servers deployed at the network edge. This system is a generic paradigm which could be applied to a variety of situations and it is not difficult to incorporate and carry out in a decision-making device. In MEC conditions, the proposed time-optimized task offloading decision algorithm finds the optimum task transferring delay when considering the planned transmitting latency and the anticipated calculating time for processing the task at MEC server. In [12] creator broaden their past system in [11] and give numerous assessments by contrasting the alternative offloading models that have been proposed. Mobile nodes may pick which MEC server to use and when to offload computational operations, reducing total latency, according to a model provided by the researchers. Wenchen Zhou et al., in [13] concentrate on device utilisation in situations in which only one MD will scale the CPU frequency and distribute computing activities across several MEC servers. In order to simultaneously optimise the balance among delay and power in a multiserver MEC system, the undertaking mission assignment judgement and processing scalability capabilities will be used.

Kai Peng et al., in [15] thought about a joint power usage, time utilization, along with price-cost management for WAs (Workflow Applications) was considered, with the fulfillment time of the WA (Workflow application) being a constraint requirement. Multi-objective computation offloading for WAs (MCOWA) was presented as a solution, which would be considered to be non sequencing evolutionary algorithm. So that the algorithm phase may meet the needs of this problem, just several variables have been tweake. Jia Yan, et al., in [19] introduced a two-stage dynamic game involving imperfect information is studied for the first time in order to concurrently coordinate integrated delivery caching and steer compute job offloading in MEC. This is the first stage of the system, in which the BS tries to optimise its projected profit by optimising its resource caching selections and the programme pricing for WDs' (wireless devices) job implementations within the constraints computing. WDs in Stage II conduct a Bayesian subgame to maximise its individual offloading selections to minimise their own costs by predicting another WDs' actions for specified pricing of service programmes. Nanliang Shan et al., in [20] the concept of beneficial computation offloading is proposed. A theoretical game-based multiuser multichannel distributed computing offloading algorithm is suggested. In a multichannel wireless interference scenario, multiuser computing offloading optimization is an NP-Hard challenge. The Nash equilibrium's efficiency is determined by the number of users that profit from computing offloading and the overall device expense.

3. SYSTEM MODEL

In this paper, we considered the energy sensitive UEs that require low power consumption, but delay insensitive (e.g. IOT devices, sensor nodes). We consider N no. of energy sensitive UEs that are executing simultaneously on a server and the server needs to decide which task has to be executed first from the task queue, so that the power required for particular task execution should be reduced along with delay. An edge server can take on the computation-intensive operation when User device does not have the sources of energy to accomplish it locally. We use deep learning based Q-learning algorithm to take offloading decision. Q learning algorithm takes action based on state and reward of the Q function at state t.

3.1. Local Computing

Let us consider E be the energy consumption of each UE required to execute locally. n will be the no. of UE, f_n will be the power coefficient of energy consumed per CPU cycle for local computation, q_n will be the is the required CPU cycles per bit, $\propto_n$ is the ratio of task computed locally, y_n is the size of computation task. So the power consumption required for n^{th} UE to execute locally can be given by-

$$E_n^{UE} = (f_n q_n) \propto_n y_n \qquad (1)$$

3.2. Edge Computing Model

Let us consider N no. of UEs are there which are expecting tasks to offload and execute on edge server due to insufficient power resources available at local server. All the tasks are there in task queue Q_{t0}. The task queue is supposed to update every time when the Q value function is changed based on reward and state. Whenever no. of tasks (Component list/UEs) in the task queue increases the processes that are being used getting increases (Shown in Fig. 2).

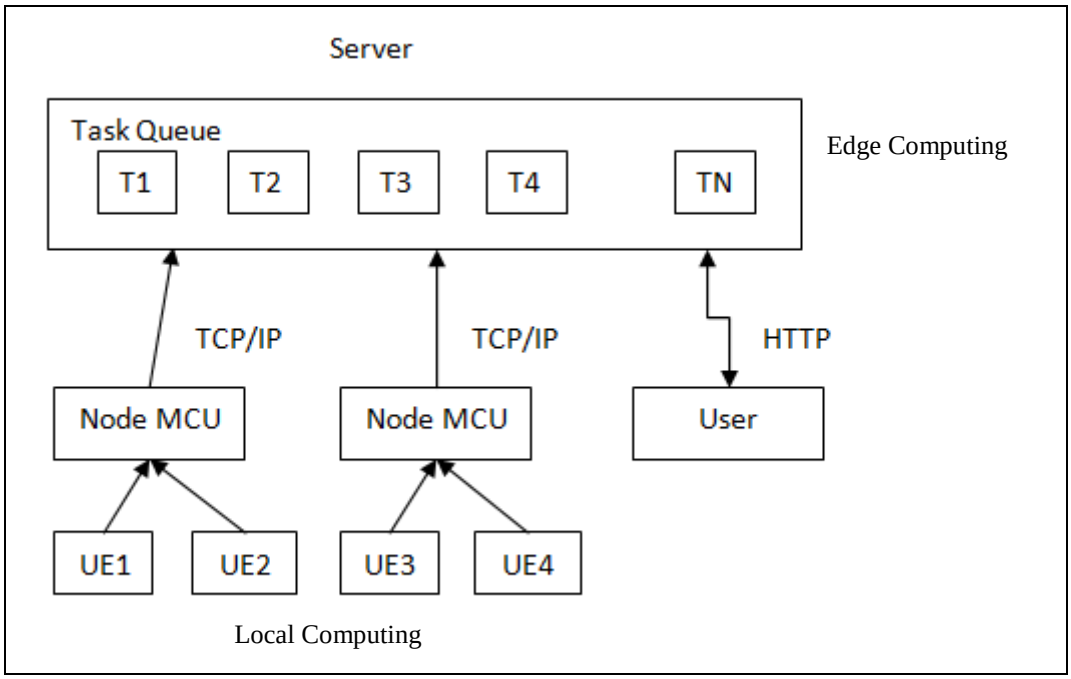

Fig. 1. Block Diagram of Task off loading model

In the above diagram, we have shown our experimental model that represents workload transferring in mobile edge computation. On the server, we are uploading tasks that come from user equipment. We are using the TCP/IP Protocol with Node

MCU for the transmission of tasks from user equipment to the server. The tasks we consider here are insensitive to delays and sensible in terms of power consumption, e.g. image processing, health care applications, agriculture applications, digital image processing etc. If the task requires less power consumption to execute than others, then the server will track the Q value using a Q learning algorithm and it will update the complete task queue. The algorithm for same is given below-

Algorithm:

Input : Q_t, Q_{t0}, Pre_node, Comp_list, Trans_amount
Output : Trans_energy
Initialization : Trans_energy $\rightarrow$ 0;
If $Q_t \neq 0$ and Pre_node ($Q_t(0)(0)$) $\leq$ Comp_list then
Trans_energy $+=$ ε ptr
If Trans_amount $\geq Q_t(0)(2)$ then
Q_{t0}.append ($Q_t(0)$)
sort tasks on Q_{t0}
else
$Q_t(0)(2) -=$ Trans_amount

We identify the total no. of UEs available in task queue , if there exist only one UE then no need calculate the Q value and take decisions for offloading task and the transmission energy is calculated by using the epsilon greedy model. We further check if there are multiple UEs available in task queue and transmission amount required for previous node is more than the next task available in task queue then just sort the task queue based on transmission amount required for processing and offload that particular task and execute on edge server from queue after sorting, So that it will take less power for execution. The transmission amount here is outcome of the Q value function based on previous state and maximum reward. Reinforcement learning is based The Q learning algorithm learns in an interactive environment which encompasses both positive and negative reinforces (i.e. reward and punishment) by performing correctly and incorrectly, respectively. This will be estimated by using the Q value function, which helps to reduce power consumption. The Q value function is defined as follow-

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\,(R_{t+1} + \gamma\, max_a\, Q_t(s_{t+1}, a) - Q_t(s_t, a_t)) \qquad (2)$$

Table 1. Notation list used in paper

Parameter	Meaning
Q_t	Q value function at time t
Q_{t0}	Task queue
Pre_node	Previous node in task queue
Comp_list	List of components in queue
Trans_energy	Transmission energy
Trans_amount	Transmission amount
A	Learning rate
Γ	Discount factor

s_t	State
a_t	Action
Q_{t+1}	New state
R_{t+1}	Reward

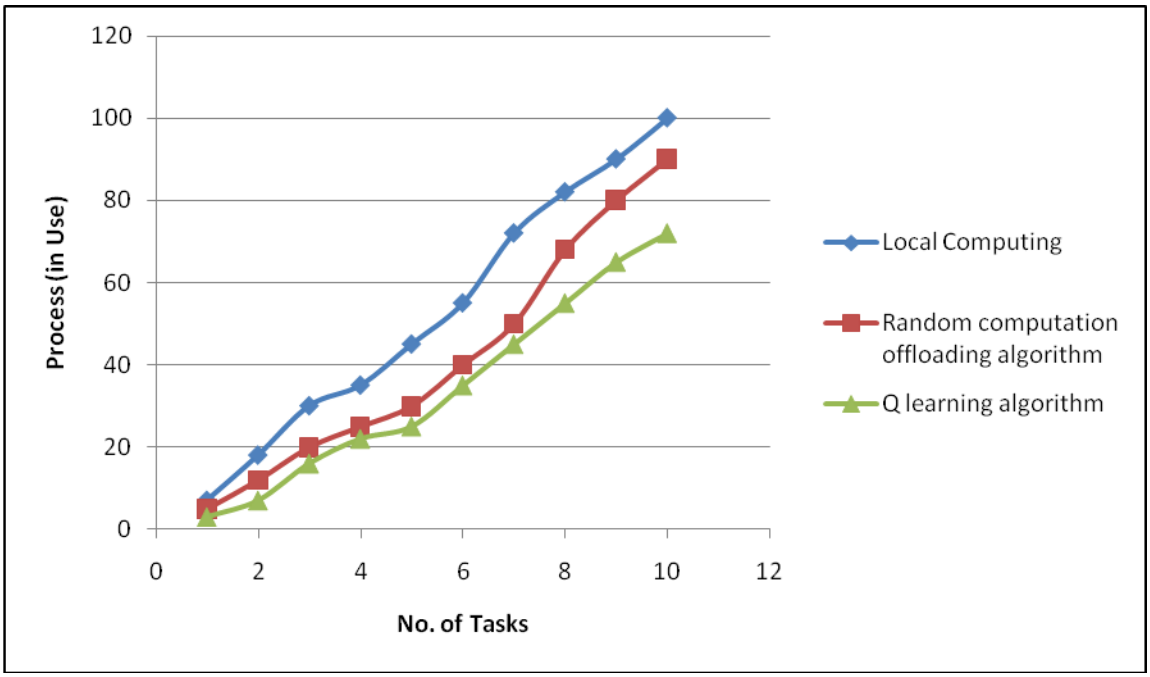

Fig. 2. Plot of different algorithm for no. of tasks vs process in use

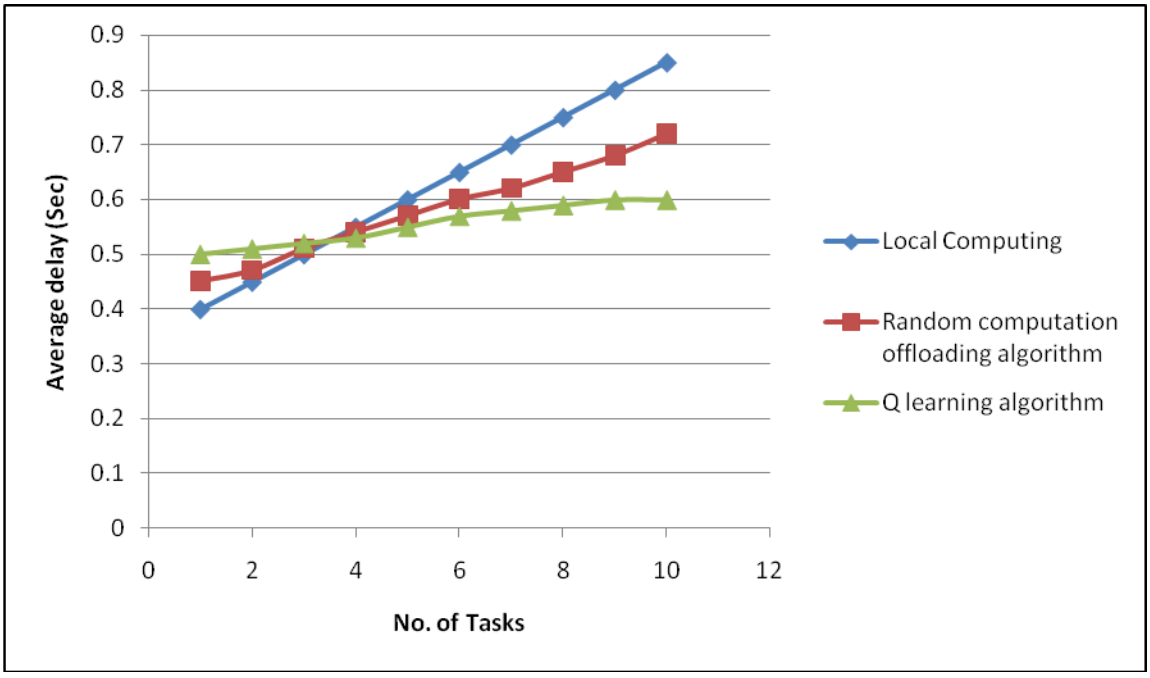

Fig. 3. Plot of different algorithm for no. of tasks vs average task delay

Fig. 2 and Fig. 3 shows plot for Q learning algorithm along with local computing of task and random algorithm taken for offloading task. Fig. 2 shows plot for no. of tasks that are being executing on server versus no. of processes in use for them. Fig. 3 shows plot for no. of tasks that are being executing on server versus average delay (in seconds) required to execute them. From fig. 2, it is clear that as no. of tasks increases processes that are in use gets increases. As compared to local computing and random computation offloading algorithm, Q learning algorithm use less no. of processes to execute tasks which ultimately results in less power consumption. Fig. 3 plot shows that as no. of tasks increases average delay for task execution gets increases. Q- learning algorithm requires less time for task execution as compared to local computing and random computation offloading algorithm when there are more no. of tasks executing simultaneously.

4. CONCLUSIONS

Deep reinforcement learning algorithms perform vital role in offloading computational processes in mobile edge computing. In this paper we assume that multiple tasks are executing at a time on different user equipments, the tasks are insensitive to delay and sensible for power consumption. We connect User Equipments to server through TCP/IP protocol using node MCU. Based on the transmission amount required to execute tasks on server, task queue updates on server and reward value gets updated in Q learning algorithm. Reinforcement learning based Q learning algorithm takes both reward and punishment in account so that power consumption is reduced. Instead of offloading workload to the distant server, by offloading to the node server improves power utilization, reduce processing latency and overall infrastructure's cost. Many intriguing problems and approaches will be addressed in the future, including deep learning-based offloading for multi-user scenarios and congestion control.

ACKNOWLEDGEMENTS

During this time, I'd want to express my gratitude to all of the staff members of the Electronics and Telecommunication Engineering Department. Thank you to the JSPM's Rajarshi Shahu College of Engineering, Tathawade, Pune, for giving me with access to the Internet and needed journals and books for my research. It was a great help to me.

REFERENCES

[1] Elhadj Benkhelifa, Thomas Welsh, Loai Tawalbeh, Yaser Jararweh, Anas Basalamah, User Profiling for Energy Optimisation in Mobile Cloud Computing. (Procedia Computer Science 52 (2015) 1159 – 1165).

[2] Khadija Akherfi ,Micheal Gerndt , Hamid Harroud, Mobile cloud computing for computation offloading:Issues and challenges [2016]

[3] Ke Zhang, Longjiang Li, Sabita Maharjan, Yan Zhang,et.al., Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks. (Article in IEEE Access. January 2016 DOI: 10.1109/ACCESS.2016.2597169).

[4] Pavel Mach, Member, IEEE, and ZdenekBecvar, Member, IEEE, Mobile Edge Computing: A Survey on Architecture and Computation Offloading. (VOL. 19, NO. 3, THIRD QUARTER 2017).

[5] Zhaohui Luo, MinghuiLiWang, Zhijian Lin, Lianfen Huang, XiaojiangDu andMohsen Guizani, Energy-Efficient Caching for Mobile Edge Computing in 5G Networks. (Appl. Sci. 2017, 7, 557; doi:10.3390/app7060557).

[6] Sandeep Kumar, Abhirup Khanna, Manan Tyagi, Vivudh Fore, A Survey of Mobile Computation Offloading: Applications, Approaches and Challenges. (June 2018).

[7] FengxianGuo_, Heli Zhang_, Hong Ji_, Xi Li_, Victor C.M. Leung, Energy Efficient Computation Offloading for Multi-access MEC enabled Small Cell Networks. (978-1-5386-4328-0/18/$31.00 ©2018 IEEE).

[8] Abbas Kiani Student Member, IEEE, and Nirwan Ansari Fellow, IEEE, Edge Computing Aware NOMA for 5G Networks. (DOI 10.1109/JIOT.2018.2796542, IEEE Internet of Things Journal)

[9] Yeongjin Kim, Student Member, IEEE, Hyang-Won Lee, Member, IEEEand Song Chong, Member, IEEE, Mobile Computation Offloading for Application Throughput Fairness and Energy Efficiency. (DOI 10.1109/TWC.2018.2868679, IEEE Transactions on Wireless Communications).

[10] Liang Huang, Xu Feng, Cheng Zhang, Liping Qian, Yuan Wu, Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. Digital Communications and Networks 5 (2019) 10–17.

[11] Ibrahim Alghamdi, Christos Anagnostopoulos, Dimitrios P. Pezaros, Time-Optimized Task Offloading Decision Making in Mobile Edge Computing. [2019]

[12] Ibrahim Alghamdi *, Christos Anagnostopoulos and Dimitrios P. Pezaros, Delay-Tolerant Sequential Decision Making for Task Offloading in Mobile Edge Computing Environments. [2019]

[13] Wenchen Zhou,1 Weiwei Fang ,1 Yangyang Li,2 Bo Yuan,1 Yiming Li,1 and Tian Wang et al., Markov Approximation for Task Offloading and Computation Scaling in Mobile Edge Computing (Hindawi, Mobile Information Systems, Volume 2019, Article ID 8172698, 12 pages, https://doi.org/10.1155/2019/8172698)

[14] Alghamdi, I., Anagnostopoulos, C. and Pezaros, D.P. (2020), On the Optimality of Task Offloading in Mobile Edge Computing Environments. (In: IEEE Global Communications Conference 2019, Hawaii, USA, 09-13 Dec 2019, ISBN 9781728109626).

[15] Kai Peng, Maosheng Zhu, Yiwen Zhang, Lingxia Liu, Jie Zhang, Victor C.M. Leung and Lixin Zheng, An energy- and cost-aware computation offloading method for workflow applications in mobile edge computing. (Peng et al. EURASIP Journal on Wireless Communications and Networking (2019) 2019:207).

[16] Md. Sajjad Hossain, Cosmas Ifeanyi Nwakanma, Jae Min Lee, Dong-Seong Kim et al., Edge computational task offloading scheme using reinforcement learning for IIoT scenario. (M.S. Hossain, C.I. Nwakanma, J.M. Lee et al. / ICT Express 6 (2020) 291–299)

[17] Nahida Kiran, Chunyu Pan, and Yin Changchuan, Reinforcement Learning for Task Offloading in Mobile Edge Computing for SDN based Wireless Networks. (978-1-7281-7219-4/20/$31.00 ©2020 IEEE).

[18] Ming Tang and Vincent W.S. Wong, Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems. (arXiv:2005.02459v1 [cs.NI] 10 Apr 2020).

[19] Jia Yan, Student Member, IEEE, Suzhi Bi, Senior Member, IEEE,et al., Pricing-Driven Service Caching and Task Offloading in Mobile Edge Computing. (arXiv:2011.02154v1 [cs.NI] 4 Nov 2020)

[20] Nanliang Shan , Yu Li , and Xiaolong Cui et al., A Multilevel Optimization Framework for Computation Offloading in Mobile Edge Computing. (Hindawi, Mathematical Problems in Engineering, Volume 2020, Article ID 4124791, 17 pages, https://doi.org/10.1155/2020/4124791)

[21] Bingxin Zhang , Guopeng Zhang , Weice Sun, and Kun Yang et al., Task Offloading with Power Control for Mobile Edge Computing Using Reinforcement Learning-Based Markov Decision

Process. (Hindawi, Mobile Information Systems, Volume 2020, Article ID 7630275, 6 pages,https://doi.org/10.1155/2020/7630275)

[22] Sungwook Kim, New Application Task Offloading Algorithms for Edge, Fog, and Cloud Computing Paradigms. (Hindawi, Wireless Communications and Mobile Computing, Volume 2020, Article ID 8888074).

[23] TahaAlfakih, Mohammad Mehedi Hassan, (Senior Member, IEEE),AbduGumaei, Claudio Savaglio, And Giancarlo Fortino, (Senior Member, IEEE), Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA. (Digital Object Identifier 10.1109/ACCESS.2020.2981434).

[24] Md Delowar Hossain ID , Tangina Sultana et al., Fuzzy Based Collaborative Task Offloading Scheme in the Densely Deployed Small-Cell Networks with Multi-Access Edge Computing.

[25] Miranda McClellan , Cristina Cervelló-Pastor and SebastiàSallent, Deep Learning at the Mobile Edge: Opportunities for 5G Networks. (Appl. Sci. 2020, 10, 4735; doi:10.3390/app10144735).

[26] Salman Raza1, Wei Liu et al., An efficient task offloading scheme in vehicular edge computing. (Journal of Cloud Computing: Advances, Systems and Applications (2020))