Computational Task Offloading Using Deep Q Learning in Mobile Edge Computing

Tanuja Satish Dhope[1][0000-0003-2907-8509] ,Tanmay Dikshit[2], Kumar Kartik[3], Unnati Gupta[4]

[1-4]Electronics and Communication, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India,
[1]tanuja_dhope@yahoo.com,[2]tanmaydikshit12@gmail.com,[3]kumarkartik1920@gmail.com,
[4]unnatigupta1511@gmail.com

Abstract. Because of the growing proliferation of networked IoT devices and the demanding requirements of IoT applications, existing cloud computing architectures have encountered significant challenges**.** A novel mobile edge Computing (MEC) can bring cloud computing capabilities to the edge network and support computationally expensive applications. By shifting local workloads to edge servers, it enhances the functionality of mobile devices and the user experience. Computation offloading is a crucial mobile edge computing technology to enhance the performance and minimize the delay. In this paper, the deep Q learning method has been utilized to make offloading decisions whenever numerous workloads are running concurrently on one user equipment (UE) or on a cellular network, for better resource management in MEC. The suggested technique determines which tasks should be assigned to the edge server by examining the CPU utilization needs for each task. This reduces the amount of power and execution time needed.

Keywords: Computation offloading ,edge server ,mobile edge server ,Deep Q Learning

1 Introduction

Existing cloud computing architectures have faced considerable hurdles as a result of the ongoing proliferation of networked IoT devices and the demanding needs of IoT applications, notably in terms of network congestion and data privacy. Relocating computing resources closer to end users can help overcome these problems and improve cloud efficiency by boosting its processing power[1]. This strategy has developed with the introduction of many paradigms; fog computing ,edge computing, all of which have the same objective of increasing the deployment of resources at the network edge. The significant issues that traditional cloud computing as centralised is experiencing include increased latency in real-time applications, low spectral efficiency. As a result of new technologies, distributed computing capabilities are increasingly being used by organisation’s or network's edge devices in an effort to explain all these challenges. Mobile Edge Computing (MEC) enables certain apps to be offloaded from resource-constrained devices like smartphones, saving resources. MEC's characteristics set it apart from typical cloud computing because, unlike remote cloud servers, the network can aggregate tasks in areas near the user and device. By moving cloud processing to local servers, MEC improves user quality of experience (QoE), in addition to reducing congestion in cellular infrastructure and cutting delay [2].

There are a number of considerations that need to be answered before jobs are offloaded to edge servers, including whether to do so and, if so, which edge server. Analysing the load on mobile edge servers is necessary to respond to the question above. The task/data offloading decision is important because it is predicted to have a straightforward impact on the Quality of Service (QoS) of the user application, including the resulting latency caused by the offloading mechanism[3]. When there is a lot of stress at the edge node as a result of a staggeringly high number of user devices using the same edge network for every task as it could result in considerable processing delays and the cessation of some processes[4]. The reasoning architecture of the MEC notion is utilised to obtain cloud computing applications. By placing several information centres at the network's node, users of smartphones will be more accessible. The network terminal can refer to a multitude of places, including indoor areas like Wi-Fi and 3G/4G.In today's world, computing offloading discusses both boosting Smartphone performance as well as attempting to guarantee energy savings simultaneously [5]. Although meeting the delay prerequisites, MEC allows the edge to perform computation-intensive applications rather than user equipment. Additionally, IoT users will participate in later detecting and processing duties in user-centric 5G networks[7].In reality, using MEC to offload computation processes results in wireless networks being used to transmit data. It is feasible for wireless connections to become severely congested if many application stations forcefully dump their processing resources to the edge node, which would dramatically slow down MEC[8]. A unified management system for CO and the accompanying wireless resource distribution in order to benefit from compute offloading is required[10].

In Section 2, this paper describes the job offloading research in MEC. The task offloading system model is described in Section 3 as local computing, edge computing, and the Deep Q learning method. Section 4 elaborates on the results and charts for various task offloading techniques. The analysis's conclusion is presented in Section 5.

2 Literature Survey

In [10] , many edge computing paradigms and their various applications, as well as the difficulties that academics and industry professionals encounter in this fast-paced area has been examined. Author suggested options, including establishing a middleware-based design employing an optimizing offloading mechanism, which might help improve the current frameworks and provide the Moblie Cloud Computing (MCC) users more effective and adaptable solutions by conserving energy, speeding up reaction times, and lowering execution costs.

In [11] author has given an energy efficient computation offloading (EECO) method, which combinedly optimizes the decisions of CO and allocation of radio resources algorithms minimizes the cost of system energy within the delay constraints in 5G heterogeneous networks.

An energy-efficient caching (EEC) techniques for a backhaul capacity-limited cellular network to reduce power consumption while meeting a cost limitation for computation latency has been proposed in [12]. The numerical findings demonstrate that 20% increase in delay efficiency. The proposed method may be very close to the ideal answer and far superior to the most likely outcome, i.e. the approximation bound.

With the use of two time-optimized sequential decision-making models and the Optimal Stopping Theory, author [13] address the issue of where to offload from and when to do so. Real-world data sets are used to offer a performance evaluation, which is then contrasted with baseline deterministic and stochastic models. The outcomes demonstrate that, in cases involving a single user and rival users, our technique optimises such decisions.

[14] examine the multi-objective computation offloading (CO) approach for workflow applications (MCOWA) in MEC which discovers the best application approach while adhering to WA deadline constraints. Numerous experimental evaluations have been carried out to demonstrate the usefulness and efficiency of suggested strategy.

In MEC wireless networks, an SDN-based solution for offloading compute. Based on reinforcement learning, a solution to the energy conservation problem that considers both incentives and penalties has been assessed[15].

Distributed offloading method with deep reinforcement learning that allows mobile devices to make their offloading decisions in a decentralised way has been proposed. Simulation findings demonstrated that suggested technique may decrease the ratio of dropped jobs and average latency when compared to numerous benchmark methods [16].A multilayer CO optimisation framework appropriate for multiuser multichannel multi server situations in MEC has been suggested. Energy consumption and latency parameters are used for CO decision from the perspective of edge users. In [17] multi objective decision-making technique has been proposed to decrease energy consumption and delay of the edge client.

3. SYSTEM MODEL

We took into account energy-sensitive UEs in this paper, such as IOT devices and sensor nodes, which have low power requirements but are not delay-sensitive. We take into account N energy-sensitive UEs that are running concurrently on a server, and the server must choose which task from the task queue needs to be done first in order to reduce the power and execution time for each work. When a user device lacks the energy resources to complete the computation-intensive task locally, an edge server can step in. To make decisions on offloading, we employ deep Q-learning algorithm. Based on the state and reward of the Q function at state t, the Q learning algorithm acts.

3.1. Local Computing

Let's assume that E represents the energy needed for each UE to operate locally. n=The number of UE, p^n =the power coefficient of energy used for local computing per CPU cycle, $c^{n=}$ the CPU cycles desired for each bit in numbers, $\beta_n$the percentage of tasks computed locally, and $S^n{}_=$ the size of the computation task are all represented by the numbers n. Therefore, the amount of energy needed for UE to operate locally can be determined by discretion.

$$E_n^{UE} = p^n c^n \beta^n S^n \qquad (1)$$

3.2. Edge Computing Model

Let's assume that N number of UEs are anticipating tasks to be offloaded and executed on edge server since local server does not have enough power resources. The task queue contains every single task. When the Q value function is modified based on reward and state, the task queue is supposed to update each time. The processes that are being used grow if the number of tasks (Component list/UEs) in the task queue rises.

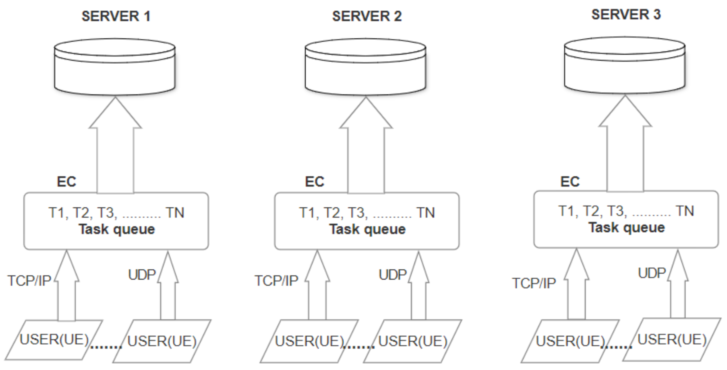

Fig.1: Block Diagram of Edge Computing Model

System model consists of workload offloading in MEC(Fig.1).The task has been uploaded to the any of 3 servers based on tasks that has come from UE. Selection of the any one of the server is based on Deep Q learning algorithm. We have used TCP/IP,UDP for transmitting task from UE to edge server depending on type of application viz. image processing ,AR/VR, health care applications, agriculture applications. The server will track the Q value using a Q learning algorithm and it will update the entire task queue if one task consumes less power than others.

Q-Learning is a reinforcement learning algorithm used in MEC that trains itself based on parameters supplied during environment building and server allocation algorithms. Later, an optimum job distribution on the servers can be accomplished using the learned model. The tasks that offload delay will be more difficult with local computing. In MEC, the state space could stand in for the present network conditions, device status, and resource availability (such as CPU, memory, and bandwidth). Making judgements about how to allocate resources requires access to this state information. Q-learning assesses the effectiveness of activities conducted in a specific condition using a reward mechanism. Rewards in MEC can be determined based on a variety of performance indicators, including latency, energy use, throughput, or user happiness. Higher rewards are associated with better decisions.
The learning method entails exploring the state-action space iteratively and updating the Q-table based on the rewards gained. Q-learning uses the Bellman equation to update Q-values iteratively:

$$P(s, a) = P(s, a) + \alpha * [R(s, a) + \gamma * \max(P(s', a')) - P(s, a)] \qquad 2$$

where
`P(s, a)` is the Q-value for state `s` and action `a`.
`α` is the learning rate.
`R(s, a)` is the immediate reward for taking action `a` in state `s`.
`γ` is the discount factor.
`max(Q(s', a'))` represents the maximum Q-value for the next state `s'` and all possible actions `a'`.

Algorithm:

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

4. RESULT ANALYSIS

We have considered total three edge servers and tasks which are requesting for the edge server services .The following parameters has been taken into account for Deep Q learning algoirithm.

Learning rate for the neural network's optimizer	0.1
Reward decay factor in the Q-learning update	0.001
Initial epsilon-greedy exploration probability	0.99
Frequency of updating target network parameters	200
Size of replay memory	10KB

Batch size for training	32
exploration probability	0.9
Maximum number of episodes for training	3000

The other parameters like transmit power, Bandwidth and noise PSD has been taken into consideration. We analyzed the number of UEs that are now in the task queue. If there is only one UE, we can decide whether to offload the job and compute the transmission energy using the epsilon greedy model. We checked if there are multiple UEs in the task queue and the amount of transmission needed for the task before it in the queue is greater than the amount needed for the task after it. If so, we simply sort the task queue based on the amount of transmission needed for processing, offload the task in question, and execute it on the edge server from the queue after sorting, using less power in the process. According to the prior state and maximum, the Q value function produced this transmission quantity.

Fig.2 shows the number of tasks in queue with respect to time on the basis of provided number of nodes, environment variables, CPU requested and processing time. The Deep Q learning algorithm assigns the requested task to any 3 of the servers based on the reward. Fig.3 analyses the three server utilization taken into consideration with respect to time.

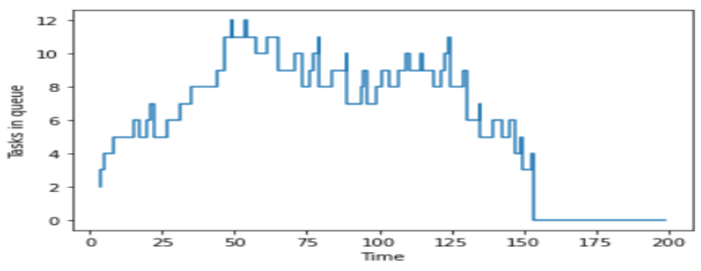

Fig.2: No. of tasks in queue with respect to time

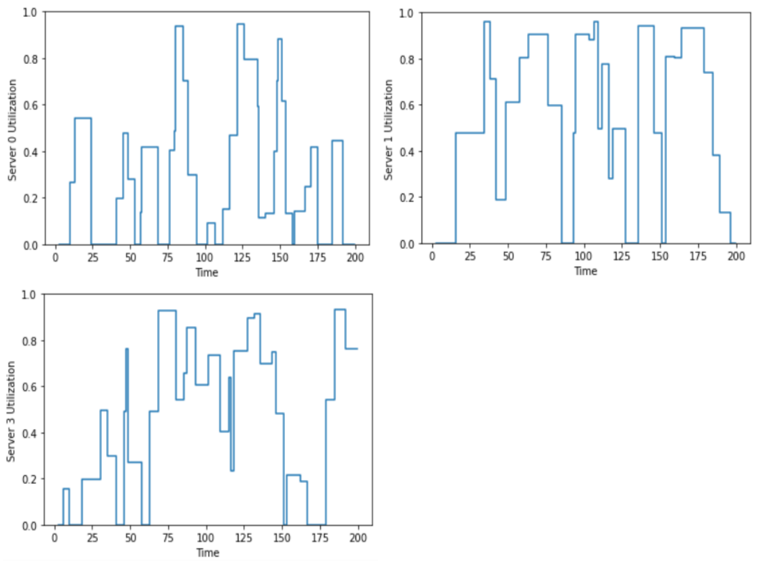

Fig.3: Server Utilization with respect to time.

CPU utilization of different CPU's and number of tasks in queue during a single window execution has been shown in Fig.4

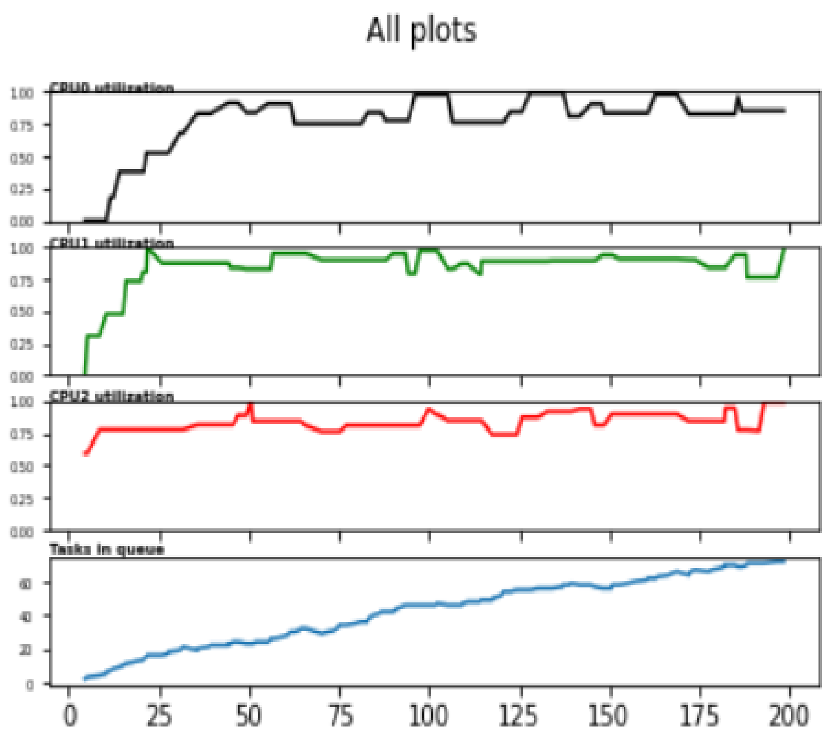

Fig.4: CPU utilization of different CPU's and number of tasks in queue during a single window execution

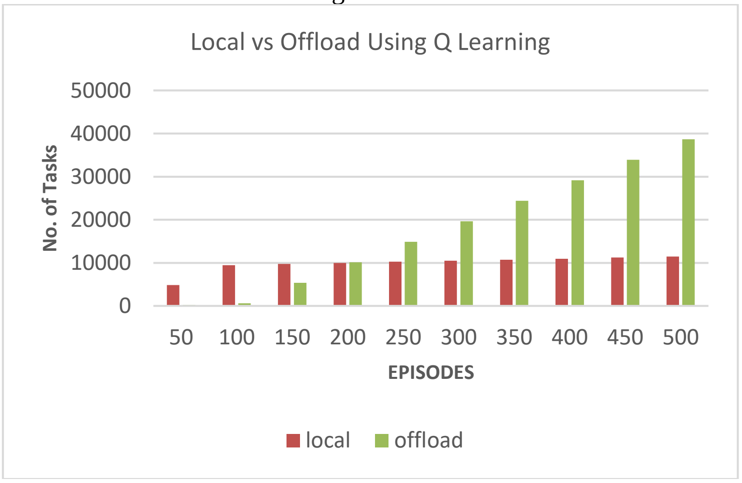

Fig.5: Episodes Vs No.of Tasks for Local computation and Edge Server Offloading using Q-Learning Algorithm

Fig 5 reflects the tasks in number which can be offloaded with respect to the edge server and tasks that can be computed locally based on Deep Q learning algorithm.

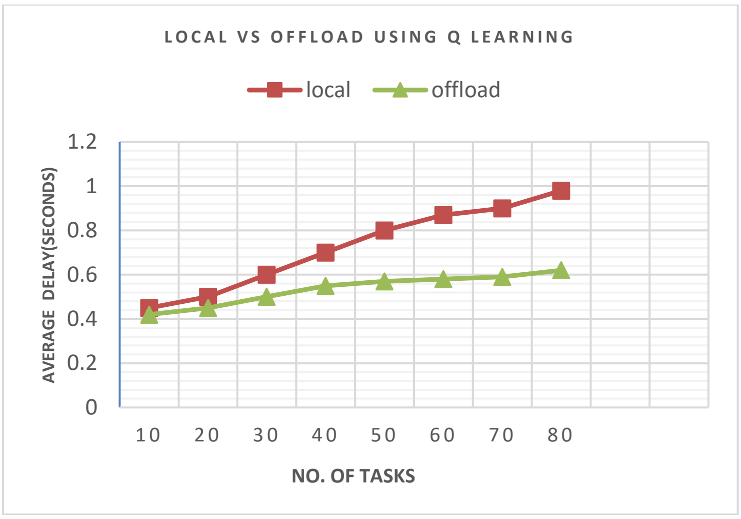

Fig.6 No.of Tasks Vs Average Delay (Sec) for Local and Offloading using Q-Learning Algorithm

As the number of tasks rises, the average time taken to complete each job also rises, as seen in Fig. 6. When there are more tasks running simultaneously, the Q-learning method requires less time for task execution than local computing.

5.Conclusion

In MEC, deep Q learning algorithms play a crucial role in the offloading of computational tasks. We make the assumption in this study that numerous tasks are running concurrently on various user devices, and that the jobs are both delay- and power-insensitive. We use the TCP/IP or UDP based on application to link user equipment to the server. The work queue on the server adjusts based on the amount of transmission needed to complete jobs, and the reward value is updated in the Q learning process. The Q learning algorithm, which is based on reinforcement learning, considers both rewards and penalties to minimise power usage. Offloading workload to a node server instead of a remote server increases power efficiency, lowers processing delay, and lowers total infrastructure costs.

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, Applied Computing and Informatics, ACI 58,pages 16 ,2016, http://dx.doi.org/10.1016/j.aci.2016.11.002
3. Yeongjin Kim, Hyang-Won Lee, and Song Chong,Mobile Computation Offloading for ApplicationThroughput Fairness and Energy Efficiency, IEEE Transactions in wireless Communication, : DOI 10.1109/TWC.2018.2868679, pp 16,2018
4. 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).
5. 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)
6. 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).
7. Liang Huang, Xu Feng, Cheng Zhang, Liping Qian, Yuan Wu, Deep reinforcement learningbased joint task offloading and bandwidth allocation for multi-user MEC. Digital Communications and Networks 5 (2019) 10–17
8. Gagandeep Kaur, Ranbir Singh Batth, Edge Computing: Classification, Applications, and Challenges, 2nd International Conference on Intelligent Engineering and Management 1-6 (2021).
9. Ahmed Alalawi, Alauddin Al-Omary , Cloud Computing Resources: Survey of Advantage, Disadvantages and Pricing, International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy. 1-6 (2020).
10. Khadija Akherfi, Micheal Gerndt, Hamid Harroud, Mobile cloud computing for computation offloading:Issues and challenges. 2-14 (2016)
11. Ke Zhang, Yuming Mao, Supeng Leng,Quanxin Zhao, Longjiang Li,Xin Peng, Li Pan, Sabita Maharjan, Yan Zhang, Energy-efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks.1-10 (2016)
12. Zhaohui Luo, Minghui LiWang, Zhijian Lin, Lianfen Huang, Xiaojiang Du, Mohsen Guizani, Energy-Efficient Caching for Mobile Edge Computing in 5G Networks. 1-13 (2017)
13. Ibrahim Alghamdi, Christos Anagnostopoulos, Dimitrios P. Pezaros, On the Optimality of Task Offloading in Mobile Edge Computing Environments. 1-6 (2019)
14. Kai Peng, Maosheng Zhu, Yiwen Zhang, Lingxia Liu, Jie Zhang, Victor C.M. Leung , Lixin Zheng, An energy- and cost-aware computation offloading method for workflow applications in mobile edge computing.1-15 (2019)
15. Nahida Kiran, Chunyu Pan,Yin Changchuan, Reinforcement Learning for Task Offloading in Mobile Edge Computing for SDN based Wireless Networks.1-6 (2020)
16. Ming Tang, Member, Vincent W.S. Wong, Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems. 1-12 (2020)
17. Nanliang Shan , Yu Li , and Xiaolong Cu, "A Multilevel Optimization Framework for Computation Offloading in Mobile Edge Computing", Hindawi ,Mathematical Problems in Engineering Volume 2020, 17 pages https://doi.org/10.1155/2020/412479.