Edge-IoT Computational offloading using Deep Learning to Enhance Energy Efficiency

Shubham Kumar[1], Prof. Deepak Pathak[2]

[1]MTech Scholar, [2]Assistant Professor

Sri Satya Sai College of Engineering

RKDF University, Bhopal, Madhya Pradesh, India

ABSTRACT: In this research work, resource allocation with advanced computing in Internet of Things (IoT) networks is achieved using machine learning approaches. Edge Computing plays a promising role in IoT networks by providing compute capabilities close to users. However, the huge number of users on IoT networks require enough spectral resources to delegate their IT tasks to an edge server, while recently IoT users have evolved to have more powerful IT skills that allow them to perform certain tasks at local level. Hence, writing IT outsourcing guidelines for such advanced IoT IT systems remains a challenge. In this thesis, centralized user clustering is examined to group users into different clusters based on user priorities. The cluster with the highest priority is assigned to offload computing and runs on the Edge Server. To address the curse of high dimensionality, deep gain is used using a Q network algorithm. Furthermore, the proposed computation algorithm outperforms other basic schemes with the same system costs.

Keywords: Edge Computing, Internet of Things (IoT), Offloading, Deep Learning, Energy Efficiency.

I. Introduction

The term Internet of Things (IoT), which refers to clearly identifiable objects, things and things in an Internet-like structure, was first proposed in 1998 [1]. In recent years, the IoT concept has become particularly popular thanks to some representative applications (for example, intelligent monitoring of greenhouses while reading electricity meters, monitoring of telemedicine and intelligent transport). Typically, the IoT has four main components- acquisition heterogeneous access information processing, applications and services. The Internet of Things (IoT) has brought about a new era of smart applications, such as smart city, smart industry, as well as smart farming by using a lot of IoT devices, like sensors, actuators, and gateways to collect and process a large amount of data generated by the IoT networks [1]. Innovations in hardware and software in recent years have contributed to the expansion of IoT networks with a large number of connected devices, which increases the requirements for data processing, storage, and communications [2]. For example, in smart farming, many sensors are deployed to monitor the environmental factors and animal welfare, like temperature, humidity, level of lighting, noise, and gas levels.

Generally, the generated data by sensors are collected by the gateway and then sent to the cloud server to perform further processing [3]. However, few challenges need to be addressed in this situation. First, many farms are located in remote areas where sensors might not be able to connect to the cloud services, edge computing, defined as providing computing capacities close to users, is a promising solution to complete the computation chain [3], [4]. Second, a large number of sensors requests for computation resource from the edge server, which burdens on the spectrum resource and the computation capacity.

Moreover, the IoT devices have been developed to have more powerful computation ability, which makes it possible for them to perform some simple data processing. Therefore, the design of the optimal computation task offloading scheme is necessary and urgent in the IoT edge computing networks.

II. Computational Offloading

The task offloading process is used to reduce the power consumption and workload of a smart device as tasks are moved to the cloud for execution. Refers to user equipment that processes some compute-intensive applications and loads the data processed by those applications into the edge server via wireless transmission, provided that continuous or other indicators are weighed.

1. *Benefit of task offloading*

- External processing units can offer more processing power, storage capacity, and flexibility than a local computer.
- The computational limitations of a device can be overcome by shifting the workload to other systems with

better performance and resources. Other aspects of the device could be improved, in particular: Power consumption, cost and portability.

- Companies that need hardware to run their business do not have to devote resources to information technology and infrastructure development.
- Cluster computing is cheaper than a single computer and much more flexible in terms of adding processing units.

2. *Limitations*

- Cloud computing services have downtime problems because outages can occur when service providers are overloaded with business.
- External processing requires a network dependency, which can cause downtime if a network connection fails.
- Network processing introduces unwanted latencies for latency-sensitive applications, including autonomous driving and video analytics. [10]
- Cloud service providers are not always secure and critical information can be leaked in the event of a security breach.
- Modern computers incorporate various technologies, including; Graphics, audio, and networking hardware inside the motherboard, making many hardware accelerators unnecessary.
- Use of cloud services requires individual control of hardware and forces users to trust cloud providers to maintain infrastructure and adequately protect their data.

III. Related Work

Liu et al. [1] studied resource allocation with advanced computing in Internet of Things (IoT) networks using machine learning approaches. Edge Computing plays a promising role in IoT networks by providing compute capabilities close to users. However, the huge number of users on IoT networks require enough spectral resources to delegate their IT tasks to an edge server, while recently IoT users have evolved to have more powerful IT skills that allow them to perform certain tasks at local level. Hence, writing IT outsourcing guidelines for such advanced IoT IT systems remains a challenge. This article explores centralized user aggregation to group IoT users into different clusters based on user priorities. The cluster with the highest priority is outsourced and runs on the edge server, while the cluster with the lowest priority performs the computation locally. For the other clusters, the development of distributed relocation policies for IoT users is modeled through a Markov decision-making process in which each IoT user is seen as an agent making a series of relocation decisions in calculating system costs based on environmental dynamics. To address the curse of high dimensionality, we use a deep Q network to learn the optimal guideline where a deep neural network is used to approximate the Q function in Q learning.
Alfakih et al. [2] proposed a reinforcement-learning-based state-action-reward-state-action (RL-SARSA) algorithm to resolve the resource management problem in the edge server, and make the optimal offloading decision for minimizing system cost, including energy consumption and computing time delay. We call this method OD-SARSA (offloading decision-based SARSA).
Wang et al. [3] proposed a fog-cloud computational offloading algorithm in Internet of Vehicles (IoV) to both minimize the power consumption of vehicles and that of the computational facilities. First, we establish the system model, and then formulate the offloading problem as an optimization problem, which is NP-hard. After that, author proposed a heuristic algorithm to solve the offloading problem gradually. Specifically, we design a predictive combination transmission mode for vehicles, and establish a deep learning model for computational facilities to obtain the optimal workload allocation.
Alelaiwi et al. [4] proposed the use of a deep-learning-based response-time-prediction framework to determine whether to offload in the nearby fog/edge node or neighbor fog/edge node, or cloud node. Furthermore, a restricted Boltzmann machines learning is applied to tackle the randomness in the availability of resources.
Chen et al. [5] proposed an intelligent task download algorithm (iTOA) for UAV on-board computer networks. Compared to existing methods, iTOA is able to intelligently perceive the network environment to decide the outsourcing action based on Deep Monte Calor Tree Search (MCTS), the core algorithm of Alpha Go. MCTS simulates the trajectories outsourcing decision makers to get the best decision while maximizing the reward, for example: B. Lower latency or power consumption. To accelerate the research convergence of MCTS, we also proposed a fractional deep neural network (sDNN) to provide the a priori probability of MCTS.
Wei et al. [6] studied the scenario where multiple mobiles upload tasks to a MEC server in a sing cell, and allocating the limited server resources and wireless channels between mobiles becomes a challenge. We formulate the optimization problem for the energy saved on mobiles with the tasks being dividable, and utilize a

greedy choice to solve the problem. A Select Maximum Saved Energy First (SMSEF) algorithm is proposed to realize the solving process.
Higuchi et al. [7] investigated the potential of such a virtual edge server consisting of multiple vehicles, the feasibility in its early deployment stage is still to be thoroughly analyzed. The simulation results based on a realistic vehicle trajectory data set show that such horizontal task offloading among vehicles can reduce the peak load on edge computing infrastructure by 53% even under a low penetration rate of V2V communication technology.

IV. Methodology

Deep edge computing (DEEPEC) consisting of a set of mobile users U, a set of micro-BSs (base station) with edge server N, and a macro-BS with a deep cloud C is presented in figure 1. It is assumed that each BS covers a local area called a zone, and a mobile user should be associated with only one zone. Edge server may be a physical server or a virtual machine with computing capacities, that its associated BS is interconnected by backhaul links, allowing a mobile user to be served by a nonlocal BS. Each mobile user can offload computing request to a BS in its zone.we assume that the macro-BS is used as the central controller, which is responsible for collecting task information, computing resource information of edge clouds in BSs, and the network status.The set of mobile users and BSs are denoted by U = {1, 2, …, u}and N = {1, 2, …, n}, respectively.

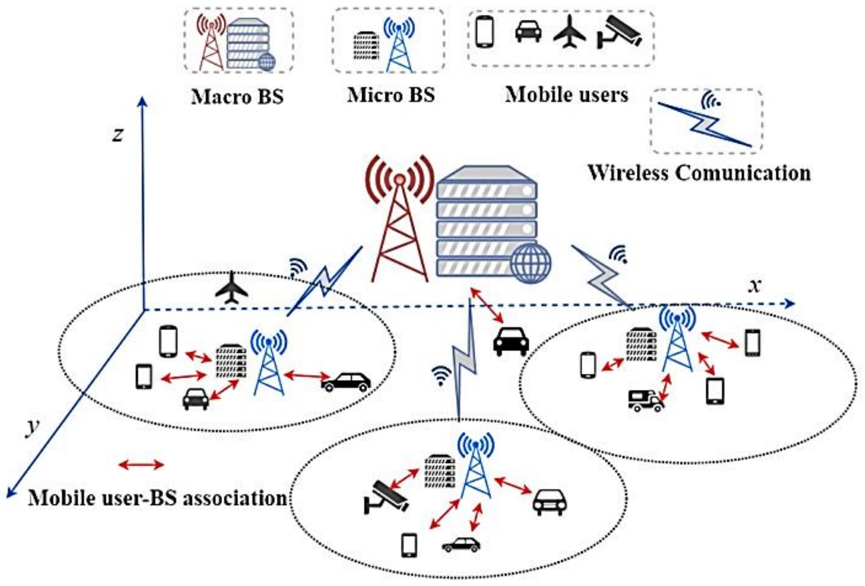

Figure 1: System Model

We assume each mobile user u ε U generate one computing request at a time, given as q_u = < w_q, s_q, pr_q, Tg_q, Tb_q>. w_q denotes the workload of request q, i.e., the required computing to accomplish the request, and s_q denotes the request input data size. We use pr_q to denote the request priority representing the importance of different requests. Tg_q and Tb_q are ideal delay and tolerable delay thresholds. Considering the position of mobile user varies over time, we use $P'_u = (x_u, y_u, 0)$ to denote the location of mobile user u at time t. All BSs are fixed and the location of BS n is given as $P'_u = (x_u, y_u, H)$ with the same attitude h.
We apply Non-orthogonal multiple access (NOMA) protocol, as a promising radio access technology for 5G systemas the communication scheme, between mobile users and BSs.
Therefore, mobile users in the same zone can transmit data to BS simultaneously at the expense of the interference. In this case, the interference may cause performance degradation, i.e., the decrease of uplink rate.

1. *User Priority Initialization*

As mentioned above, an ultradense IoT network is considered with a large number of IoT users and many IoT users are requesting for computing services from the resource-restricted edge server, but it is noted that those users have different priorities of task execution. In this section, a centralized user clustering method is investigated at the gateway to group the IoT users according to user priorities which are then assumed similar across users in the same cluster. Here, the user priority of each IoT user is defined by a tuple, including its distance d_i to the gateway and its computation offloading probability Pri. We define the computation offloading probability of each IoT user as:
Considering a typical IoT user u_i as an example, each task is generated with its task execution latency requirement as L_{th}^j and the task size as m_j. The task execution latency at the IoT user is calculated as $L_d^j = m_j/D$.

Algorithm:

```
Input:
number of observed tasks J, clusters H and clustering feature vector, user priority $x_i$
Initialization
Randomly initialize H cluster centroids
Repeat
for i = 1 : U do
for h = 1 : H do
Calculate cluster index $h_i$ for $x_i$
end for
end for
for h = 1 : H
do
Update the cluster centroids by ch
end for
Until: No change of the cluster centroids
```

2. Computatıon Offloadıng Scheme vıa Deep Reınforcement Learnıng

After centralized user clustering, the IoT users are grouped into different clusters by the gateway according to their unique features, user priorities. In each cluster, each IoT user is assumed to have the same user priorities. Here, the cluster with the highest priority is designated as edge computing while the cluster with the lowest priority is assigned as local computing. For the other clusters, the optimal distributed computation offloading scheme solved by DRL is proposed in this section. First, the basic formulations of reinforcement learning are shown and then a computation offloading scheme is designed with modeling the computation offloading process as an MDP. Therefore, a deep Q-network-based computation offloading algorithm is proposed to learn the optimal computation offloading policy, which can deal with the MDP problem with large-state space.

Reinforcement Learning is the process that the agents continuously learn optimal strategies by interacting with the environment. In this scenario, each IoT user is considered as an agent and everything else in the IoT network is made up of the environment. Here, due to performing user clustering, the scale of the computation offloading problem in the IoT network is reduced. The computation offloading scheme is assumed to have time slots structure.

From Fig. 3, each agent, the IoT user ui observes a state sk from state space S at time epoch k, and then, an action ak is taken from the action space A, that is, decides which computing mode is taken by selecting different transmit powers based on the policy π. Therefore, the environment changes, new state sk+1 is observed and a reward Rk e is obtained. In our scenario, the reward is designated as a negative reward, the system cost Ck, the weighted sum of energy consumption, and task execution latency. A reinforcement learning problem usually contains few key elements, including {Action, State, Reward, Environment}, and the environment is typically formulated as an MDP.

Action: At time step k, the action ak for each IoT user in cluster Ch is the discrete transmit power, in which Pk t = 0 indicates choosing local computing while others are actual transmit power in edge computing.1 In our scenario, the action set is denoted by A = {0,P}.

State: The states of the agent represent the exploration information from the environment. Here, we use the channel gain gk, task queue t k stored at the IoT user, and its remaining computation capacity ratio rk to present the exploration information. Hence, the state of the user in cluster Ch at the time step k,

Reward: The function of the reward signal is to encourage the learning algorithm to reach the goal of the optimization problem. In this article, the negative reward is adopted to minimize the system cost while making the right decisions on computation offloading and power allocation. Here, the system cost is measured by the weighted sum of energy consumption and task execution latency in the IoT network. To make it clear, we use reward shaping to make the reward more informative and accelerate the training.

V. Result Analysis

To evaluate the effectiveness of the proposed algorithms, the proposed algorithm is implemented using Matlab R-2020a. The simulations were conducted on an Intel i5, 3.7Ghz PC with 8 GB RAM.

1. Energy consumption versus the number of Iteration

In this case, the maximum power p_{max} of mobile users is set to 5w. Figure 2 shows the performance of the proposed Deep reinforcement learning with variable number of iterations. It is observed that the energy consumption of proposed methodology decreases with increasing iteration of reinforcement learning.

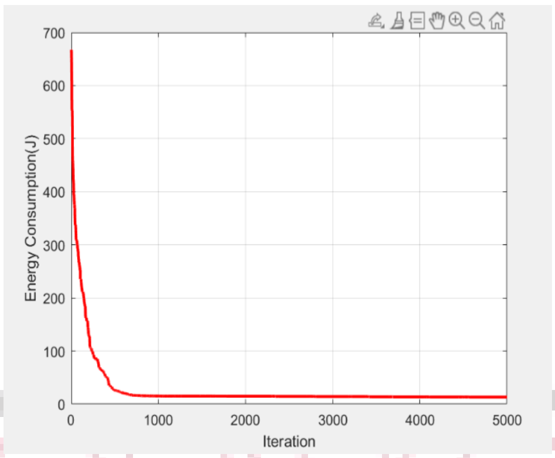

Figure 2: Energy Consumption with respect to number of iterations

2. Energy consumption versus the number of users

In this case, the maximum power p_{max} of mobile users is set to 5w. Figure 3 shows the performance of the proposed Deep reinforcement learning as compared to the Simulated annealing power allocation (SAPA). Obviously, as the number of mobile users increases, the total transmitting energy consumption increases. It is also observed that the energy consumption of proposed methodology is always smaller than that of SAPA under different number of mobile users, which implies that proposed methodology can give a better power allocation result compared to SAPA in energy saving. In proposed methodology, each mobile user gradually decreases from p_{max} to the sub-gradient direction, to find its optimal transmitting power and achieve the Nash equilibrium. However, in SAPA, the stable solution obtained in the process of random optimization may be a local optimal solution. If there is no special statement, the following experiments are based on the power policy P* obtained by proposed methodology.

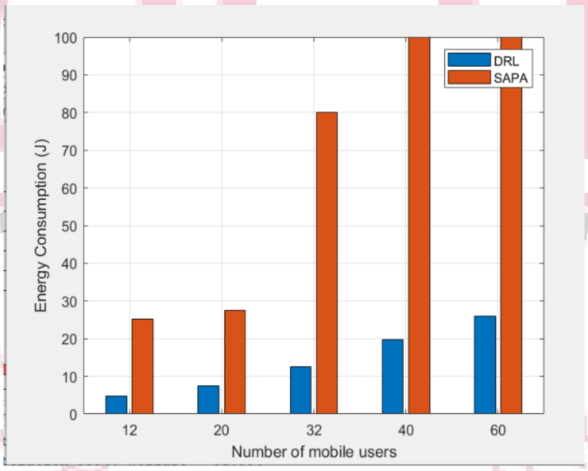

Figure 3: Energy Consumption with respect to number of mobile users

3. Energy consumption versus the maximum power

As shown in Figure 4, under different maximum power p_{max}, i.e., p_{max} = 4w, 5w, 6w, we evaluate the energy consumption under different number of mobile users. It is noticed that the energy consumption is related to the maximum power pmax, i.e., the bigger the pmax is, the higher the energy consumption is. This is because the average transmitting power of mobile users is higher under the bigger maximum power pmax, which leads to more energy consumption.

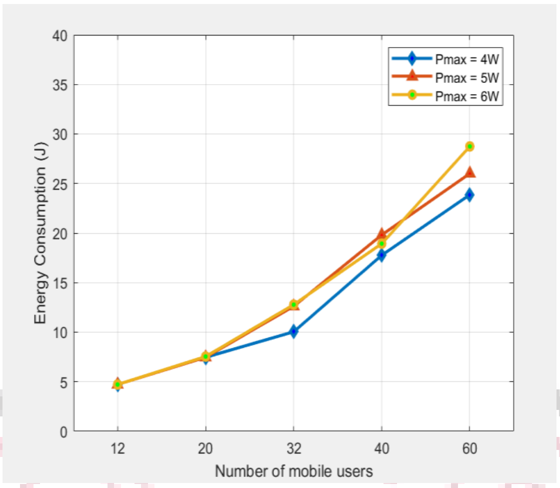

Figure 4: Energy Consumption with respect to maximum power

4. Welfare versus number of mobile users

In this case, the maximum power p_{max} of mobile users is set to 5w. Figure 5 shows the performance of the proposed Deep reinforcement learning with variable number of users. It is observed that the welfare of proposed methodology increases with increasing number of users of reinforcement learning.

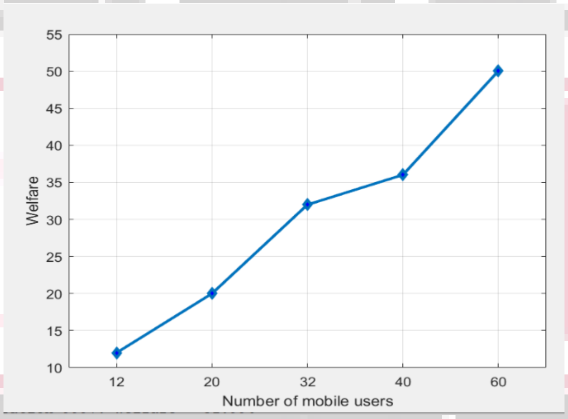

Figure 5: Welfare with respect to number of mobile users

5. Welfare versus different request workload

As shown in Figure 6, under different request workload W_q, i.e., W_q = 1500, 2000, 2500, we evaluate the welfare under different number of mobile users. It is noticed that the welfare is related to the request workload W_q, i.e., the bigger the W_q is, the higher the welfare is. This is because the average request workload of mobile users is higher under the bigger request workload which leads to more welfare.

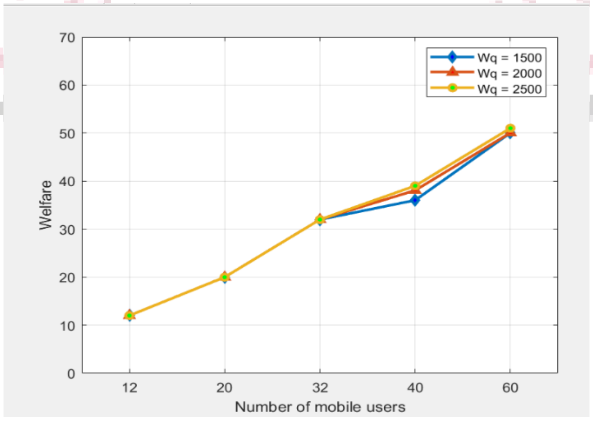

Figure 6: Welfare with respect to different request workload of mobile users

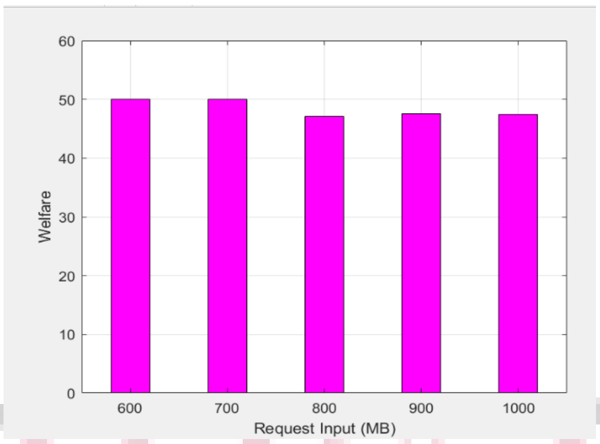

Figure 7: Welfare with respect to different request workload

VI. Conclusion

In this research, the request scheduling problem is studied in the ultra-dense edge computing network. The algorithm considers a dense edge network consisting of a macro-BS, many micro-BSs and a large number of mobile users under the 5G architecture. The NOMA protocol is used as the multiple access scheme between users and BSs. The power allocation problem among IoT users is solved using priority based deep reinforcement learning algorithm that first of all prioritize users. Then deep reinforcement learning algorithm optimizes the request offloading for mobile users and the computing resource scheduling at the micro-BSs, by forming a mixed-integer non-linear program. Simulation results verify that proposed algorithm can effectively save energy consumption and maintains a good performance in a dynamic scenario system. In the future, we will work on the design of edge computing resource scheduling algorithms for systems based on realistic applications, so as to solve the bottlenecks of practical problems.

References

[1] X. Liu, J. Yu, J. Wang and Y. Gao, "Resource Allocation With Edge Computing in IoT Networks via Machine Learning," in IEEE Internet of Things Journal, vol. 7, no. 4, pp. 3415-3426, April 2020.
[2] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio and G. Fortino, "Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA," in IEEE Access, vol. 8, pp. 54074-54084, 2020.
[3] X. Wang, X. Wei and L. Wang, "A deep learning-based energy-efficient computational offloading method in Internet of vehicles," in China Communications, vol. 16, no. 3, pp. 81-91, March 2019.
[4] Abdulhameed Alelaiwi, “An efficient method of computation offloading in an edge cloud platform”, Journal of Parallel and Distributed Computing, Volume 127, May 2019, Pages 58-64.
[5] Jienan Chen, Siyu Chen, Siyu Luo, Qi Wang, Bin Cao, Xiaoqian Li, “An intelligent task offloading algorithm (iTOA) for UAV edge computing network”, Digital Communications and Networks, 2020.
[6] F. Wei, S. Chen and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," in China Communications, vol. 15, no. 11, pp. 149-157, Nov. 2018.
[7] T. Higuchi, S. Ucar and O. Altintas, "Offloading Tasks to Vehicular Virtual Edge Servers," 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW), Monterey, CA, USA, 2019, pp. 162-163.
[8] M. Khayyat, I. A. Elgendy, A. Muthanna, A. Alshahrani, S. Alharbi and A. Koucheryavy, "Advanced Deep Learning-based Computational Offloading for Multilevel Vehicular Edge-Cloud Computing Networks," in IEEE Access.
[9] Y. Li, F. Qi, Z. Wang, X. Yu and S. Shao, "Distributed Edge Computing Offloading Algorithm Based on Deep Reinforcement Learning," in IEEE Access, vol. 8, pp. 85204-85215, 2020.
[10] H. Guo, J. Lv and J. Liu, "Smart Resource Configuration and Task Offloading with Deep edge computing," 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, Spain, 2019, pp. 1-6.