Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach

Pavitra Basappa Gudimani

Summer term 2020

Abstract

The amount of data generated by all the mobile devices in the future can cause congestion at the cloud for data processing and cause a delay in service time. Mobile Edge Computing (MEC) architecture is a promising solution for a given challenge. MEC improves the response time by moving network and computing resources closer to the customer. MEC does not send the data to the cloud for further processing rather the edge network itself will process the data. The burst request at MEC faces an issue in allocating computing and network resources. At this stage, Deep Reinforcement Learning based Resource Allocation (DRLRA) scheme adapts to a mutative nature and allocates computing and network resources. Thus, it reduces the service time to users and offers fairness in the use of resources at MEC. DRLRA achieved better results in varying MEC environments than the traditional Open Shortest Path First (OSPF) algorithm.

1 Introduction

The phenomena of embedding the sensors in smart devices to sense the environment have changed the way of living life. The data sent by IoT devices from smart cities, smart healthcare, smart vehicles [1], etc. results in a high volume of data. And it is produced at a high pace (the European Commission has predicted that there will be 50 to 100 billion smart devices connected to the Internet by 2020) [2]. The generated data is further transferred to the cloud. The volume and speed of the data generated create congestion at the cloud. Thus, efficient data processing architecture is required.

At this stage, MEC architecture aid in reducing the network congestion and response delay. It deploys the applications in a distributed form in mobile edge servers to deliver the computational and critical application services. It appears to be an encouraging solution to the problem faced by several mobile devices. On the other hand, it has following limitations. Primarily, the initial cost of investment to deploy MEC Servers (MECS) at the base stations and their maintenance is expensive. Secondly, only a restricted number of applications can be deployed to supply services to innumerable edge devices [2]. Lastly, the dynamic nature of human life brings new requirements to the devices. Continuously changing the location of requests generation, and inconsistent quantity of data cause serious burst over MECS. This leads to an increase in the computational load, and congestion in the network links of certain regions. MEC fails to provide flexibility in granting computing and network resources.

Therefore, the author has presented DRLRA algorithm to assign computing and network resources accordingly under the dynamic circumstances of MECS. DRLRA uses Deep Reinforcement Learning (DRL) technology to understand the environment and improve its behavior to perform favorable action to allocate resources. Software Defined Network (SDN) with MEC architecture centralizes the distributed network. The main goal is to minimize service time by utilizing DRLRA.

2 SDN-enabled MEC architecture

Deploying all the applications on a single MECS is expensive. MECS solves the problem by virtualizing the resources. When mobile requests for an application, it will be sent to MECS in its vicinity. If there is no required application, the request will be routed to other MECSs with the needed application deployed. Hence, the total service time includes the access time to nearby MECS, routing time among MECSs, and processing time at corresponding MECS.

To have the request delivered to destined MECS in the mutative environment the routing should happen correctly. That's where the emerging SDN technology comes in handy to understand the state of the network. In this context SDN uses below planes [3].

1. Data Plane - Forwarding devices switches, routers, and gateways reside in this plane and work according to decisions made by the controller in the control plane.
2. Control Plane - The core of SDN architecture and all forwarding decisions made upon the control logic provided here.

DRL unit with the DRLRA algorithm is installed in the SDN controller plane and it takes dynamic states of the network as input from the data plane and it trains the agent to make decisions, based on which SDN controller can perform forwarding.

3 Problem Definition

MEC architecture without resource allocation algorithm faces below problems.

Edge network routing delay: The routing delay for the aggregated requests of mobile devices at nearby MECS to the MECS that is hosting the required application [4].

Data Processing delay: Number of CPU Cycles required for computation at the destined MECS [4].

Variance of network resource allocation: Several requests will be forwarded to destined MECS. Allocation of network resources on the data links should be stabilized to reduce network traffic and to transfer the request [4].

4 Deep Reinforcement Learning Based Resource Allocation Algorithm

A smart resource algorithm understands the varying environment of the MECS and makes a feasible sequence of decision on resource allocation using Deep Q-Network (DQN) framework.

4.1 Reinforcement Learning

Reinforcement Learning (RL) is a branch of Machine Learning, which trains the agent without labelled data [5]. The agent continuously interacts with the environment and understand it. This way the agent strengthen it's decision making ability [6]. Then agent acts in the state and gets a reward. The essential terms for RL procedure includes Episode, Policy, Cumulative reward and Action value function. Episode is the sequence of states, actions, and rewards that occurs between initial state and termination state. Policy is mapping from states of the environment to actions to be taken in those states [7]. Cumulative reward is R_t, Sum of reward obtained by an agent at the end of each episode. $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i, \gamma \in [0,1]$, γ is discount factor to indicate the importance of future reward to the current state. T is the final step of an episode [4]. Action value function is to quantify the quality of the action made at the state.

4.2 Deep Q Network

DQN is a powerful framework of DRL. It takes the current state as an input and gives action value for every action that can be taken in that state i.e Q(s,a;ω), where ω is a parameter of network. Action with the highest Q - value $a = argmax_{a_t} Q(s_t, a_t; \omega)$ is selected using ϵ-greedy strategy. To make the generated Q - value closer to the target Q - value, Mean Square Error, a loss optimizing function is used [4].

$$L(\omega) = E[(r + \gamma max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}; \omega) - Q(s_t, a_t; \omega))^2] \quad (1)$$

where $r + \gamma max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}; \omega)$ is the target Q - value and $Q(s_t, a_t; \omega)$ is evaluation network Q - value.

4.3 Training

DRL agent in a MEC environment is trained using DQN framework. DQN framework overcomes the issues faced by traditional Q-learning. Q-learning uses a table to store the generated Q - values. When the network becomes complex, the table will not be sufficient to store the values. DQN framework generates Q - values using a neural network. Therefore, DRLRA adapts DQN framework which is suitable for a mutative MECS environment. It takes the MECS environment as an input state and generates Q - value for each possible action like allocating network and computing resources in that state. Three following factors of RL used are in DRLRA.

1. **State :** Consist of the location of generated requests for application m and aggregated requests on MECS v within the service region of generated request. Vector for the state is given as $s = \{p_m^v, \forall v \in V, \forall m \in M\}$ [4]. p_m^v indicates MEC at the base station where request generated [4].
2. **Action :** Performed action to allocate network and computing resources. Action vector is given as $a = \{\alpha_v^m, \forall v \in V, \forall m \in M\}$. Where α_v^m action taken by MECS to process the aggregated requests to required application installed on another MECS [4].
3. **Reward :** After acting a_t at every time step t, the agent gets a reward. The main objectives, reducing service time and balancing utilization of network and computing resources are considered for achieving reward from the environment. The reward is given as $r = w_1 \cdot T + w_2 \cdot b^{net} + w_3 \cdot b^{cp}$, where T is service time for requests, b^{net} and b^{cp} are calculated by variance of transmission load on network links and computing load on each MECS respectively. $w_i, i \in 1, 2, 3$ are weight of the elements T, b^{net}, b^{cp} [4].

Figure 1 explains the training process. Experience Replay memory is installed to store environment response for taken action and parameters ω and ω^- are set for target and evaluation network respectively. Episode k is initialized with state s_t at time t. s_t is taken as input to evaluation network and using the ϵ-greedy method action a_t with maximum Q - value is chosen. The agent gets a reward r_t and next state s_{t+1}. $Q(s_t, a_t, r_t, s_{t+1})$ is stored in experience

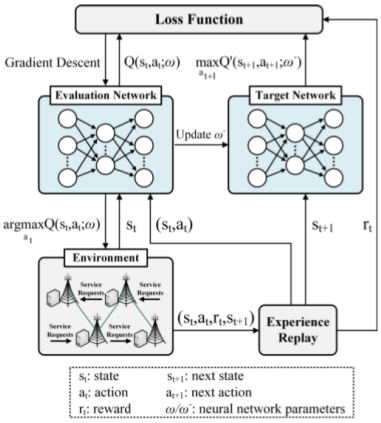

Figure 1: Detailed DQN Framework [4]

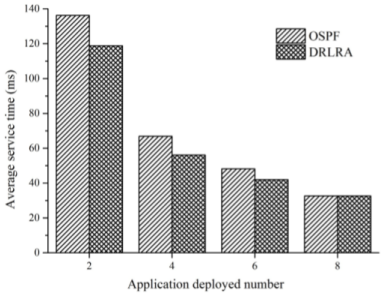

Figure 2: OSPF and DRLRA comparison under different application deployed number [4]

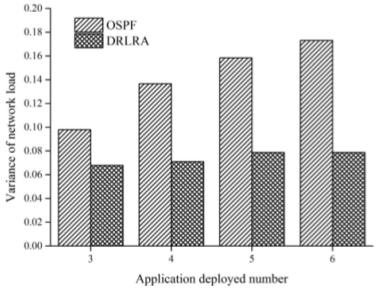

Figure 3: OSPF and DRLRA comparison under variance of network load [4]

replay. To train the network, sample state s_t and action a_t applied to evaluation network from memory. $Q(s_j, a_j; \omega)$ and $\gamma max_{a_{j+1}} Q^{'}(s_{j+1}, a_{j+1}; \omega)$ calculated from evaluation and target network respectively. The loss function is used to bring the Q - value of evaluation network closer to the target network through gradient descent and backpropagation. Finally, the trained evaluation network is obtained after performing resource allocation of t steps in episode k [4].

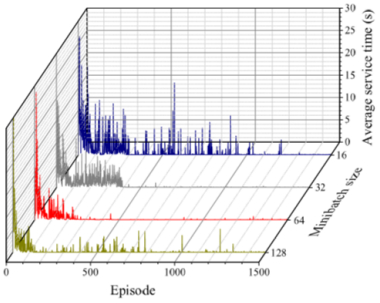

Figure 4: Convergence performance of DRLRA [4]

5 Performance Evaluation

Average service time minimization : Fig. 2 shows that the average service time decreases with an increase in the number of deployed applications. As the number of deployed applications increases, each MECS will almost have the requested application. Therefore, the distance between the location of requests generated for a specific application and MECS hosting the required application will reduce. Thus, resulting in a reduction in average service time. At a certain point, when the number of deployed applications reaches a maximum, the service time is minimized.

Resource Allocation Balancing Analysis : The complexity of the MECS network increases with an increase in deployed number of applications which degrades the performance of OSPF is shown in Fig. 3. On the other hand, DRLRA, while making decisions on path selection for routing it learns the situation and adapts to it. Therefore, DRLRA allocates resources by using it evenly and prevents network congestion.

Convergence Performance Analysis : Fig. 4 shows as the batch of samples provided increases, the speed required to stable the average service time is reduced. The agent will not get a chance to learn enough with the small number of samples. At sample size of 16, proposed DRLRA displays a high degree of randomness and slow convergence speed. With the huge batch size of 128, DRLRA agent might converge to the poor average service time. Therefore, the size of samples should be chosen carefully. Experimental result in the graph shows a better result at minibatch size of 32 or 64 [4].

References

[1] Wen Sun, JiaJia Liu, and Haibin ZHang. *When Smart WearableS meet IntellIgent VehIcleS: challengeS and Future dIrectIonS*. IEEE Wireless Communications, 2017.

[2] Xiang Sun and Nirwan Ansari. *EdgeIoT: Mobile Edge Computing for the Internet of Things*. IEEE Communications Magazine, 2016.

[3] Kuljeet Kaur et al. *Edge Computing in the Industrial Internet of Things Environment: Software-DefinedNetworks-Based Edge-Cloud Interplay*. IEEE Communications Magazine, 2018.

[4] Jiadai Wang et al. *Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach*. IEEE Transactions on Emerging Topics in Computing, 2019.

[5] *Reinforcement Learning*. URL: `https://en.wikipedia.org/wiki/Reinforcement_learning`.

[6] *Deep Q Networks*. URL: `https://towardsdatascience.com/qrash-course-deep-q-networks-from-the-ground-up-1bbda41d3677`.

[7] *Policy*. URL: `https://stackoverflow.com/questions/46260775/what-is-a-policy-in-reinforcement-learning`.