2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC) | 978-1-6654-5818-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICAIIC54071.2022.9722676

DDPG-Edge-Cloud: A Deep-Deterministic Policy Gradient based Multi-Resource Allocation in Edge-Cloud System

Arslan Qadeer, and Myung J. Lee
Department of Electrical Engineering
The City College of New York of CUNY
New York, USA
{aqadeer000@citymail, mlee@ccny}.cuny.edu

***Abstract*—5G and beyond is the key enabler for extreme mobile-broadband (xMBB), Massive and Ultra-reliable machine-type communication (mMTC, uMTC). To handle such large-scale and real-time traffics, Edge Cloud (EC) plays a critical role to minimize the latency and provide compute power at the edge of the network for Internet of Things (IoTs). However, an EC endures limited compute capacity in contrast with the back-end cloud (BC). Intelligent resource management techniques become imperative in such resource constrained environment. This paper studies the problem of compute and wireless resource allocation in an integrated EC and BC environment. Machine learning-based techniques are emerging to solve such optimization problems. However, it is challenging to adopt traditional discrete action space-based methods since they do not scale well in large-scale environments. To this end, to overcome the bottlenecks of wireless bandwidth and compute capacity in resource constrained EC and BC, we propose continuous action space-based DDPG-Edge-Cloud, a deep deterministic policy gradient-based multi-resource allocation (MRA) framework with a pruning principle. The proposed agent is equipped with a Conv1D residual block, gated recurrent unit (GRU) layer and an attention layer for local and long-term temporal feature extraction. We validate the proposed framework by comparing with two alternative agents. Experimental results demonstrate that our proposed agent converges fast and achieves up to 55% and 86.5% reduction in operational cost and rejection rate, and achieves up to 115% gain in the quality of experience on average.**

***Index Terms*—Edge cloud computing, deep deterministic policy gradient, resource allocation, smart city, IoT.**

I. Introduction

Next generation mobile applications (e.g. Augmented Reality (AR), Virtual Reality (VR)) and streetscape applications (e.g. swift control-response for emergency vehicles and situation-aware traffic/pedestrian signaling) possess resource-hungry and real-time constraints [1]. Edge-cloud (EC) architecture is a stepping stone to meet the above compute and real-time constraints by reducing the network latency and providing the computational resources at the edge of the network [2]. Furthermore, A three-tier hierarchical EC system integrated with the back-end cloud (BC) provides support for a broad-range of applications with varying QoS requirements in greater extent [3].

Edge clouds possess a limited amount of computational resources [3] and back-end clouds experience the same in the case of pay-as-you go model [4]. 5G supports dynamic Radio Access Network (RAN) and a wider frequency spectrum landscape [5]. However, in the presence of excessive amount of connected devices in the EC environment, a large amount of concurrent traffic can be anticipated. Thus, communication resources, which connect the devices with EC and BC, also become a bottleneck for the system. This multi-resource allocation (MRA) and system cost reduction challenge is manifold: First, handling the user requests from a wide range of applications at large scale with different QoS requirements. Second, the computational complexity pertaining to optimal resource allocation in a large system particularly in dynamic traffic patterns requires scalable solution techniques.

The proven success of Machine Learning (ML) based techniques has spurred the adoption of ML algorithms to solve control and management problems for IoTs in clouds and 5G wireless networks [6]–[8]. Cheng *et al.* [9] utilized the Deep Reinforcement Learning (DRL) to train the Deep Q-Network in order to solve a resource provisioning and task scheduling problem in a cloud-based environment under the strict QoS requirement. Wei *et al.* [10] proposed a natural actor-critic reinforcement learning framework to jointly solve the problem of content caching, computation offloading and radio resource management with the goal of minimizing the end-to-end delay. Deep deterministic policy gradient (DDPG) based methods, which provide superior state representation in high-dimensional space, are also adopted to solve the resource allocation problems. Peng *et al.* [11] leveraged the DDPG and hierarchical learning architectures to jointly solve the spectrum, computation and storage allocation problem in an EC based system. Recent studies [12], [13] solved the computation offloading and resource allocation problem for multiple mobile users in EC based systems by utilizing the DDPG-based framework and proposing the sate-of-the-art algorithms. Nevertheless, all of the above consider either EC or BC based environments separately, and lack the three-tier hierarchical architecture integrated with the BC.

Motivated from the above discussion, we aim at presenting a scalable solution which can also be easily implemented in real-world scenarios like COSMOS testbed. The COSMOS is a National Science Foundation (NSF) sponsored project with many academia partners including The City College of New York [14]. Our recent work [15] proposed a novel resource allocation framework to solve the bandwidth allocation problem in COSMOS based environment. The presented results are encouraging, which stimulated to extend the current framework [15] and comprehensively solve the multi-resource allocation (MRA) problem with continuous action-space. This is the rationale behind the introduction of DDPG-Edge-Cloud.

Identify applicable funding agency here. If none, delete this.

The proposed framework takes into account the wireless and computation resource allocation problem equitably at the Edge-cloud (EC) and back-end cloud (BC). To the best of our knowledge, none of the existing works applied DDPG with local and temporal feature learning networks, and pruning principle to solve the MRA problem jointly in EC and BC with the goal of minimizing overall system cost for providers, meeting the strict QoS requirements of applications and fast self-learning capability.

The main contribution of our work is summarized as follows:

- We present a simple user job model which takes into consideration both the deadline of the job and data to be processed at the same time. Thereafter, to aptly process these jobs, we present a multi-resource allocation (MRA) model under an integrated wireless communication, EC and BC environment to handle a large-scale of user requests under constrained resources.
- We formulate the MRA problem for user requests into DDPG-based Actor-Critic framework. The reward maximization objective for resource allocation with wireless bandwidth, EC and BC compute resources considers to optimize three fundamental bench-marking points; 1) Minimize system cost; 2) Minimize rejection rate for enhanced reliability; and 3) Minimize round-trip time of a user request for better Quality of Experience (QoE).
- Instead of using fully-connected networks (FCNs), both the actor and critic networks in the DDPG-based framework consist of convolution (Conv1D) residual block, gated recurrent unit (GRU) and an attention layer. Conv1D residual block aims at learning the correlations among local features of each input state, and GRU and attention layers capture the temporal features. Further, our proposed pruning principle [15] helps to minimize the rejection rate by efficiently offloading the service requests to servers and base stations.
- The performance of DDPG-Edge-Cloud is evaluated in terms of convergence efficiency, average operational cost, rejection rate and QoE. Compared with other DDPG-based agents, our proposed method achieves better convergence and loss results during training. In addition, our proposed approach outperforms two DDPG-based methods in all of the test scenarios. Overall our proposed approach achieves better performance for the MRA problem.

The remainder of this paper is organised as follows: The multi-resource allocation (MRA) in EC and BC based smart streetscape system is modeled in Section II. Section III introduces our core DDPG-based framework and pruning principle for our MRA system. Section IV presents the performance evaluation and discusses the experimental results, followed by the conclusion and future directions in Section V.

II. System Model and Problem Formulation

A. System Description

We consider the Edge-cloud (EC) based streetscape system as shown in Fig. 1. Our proposed system provides 5G based wireless radio access (including sub-6 GHz and mmWave) to sensors, street signals, vehicles, security cameras, mobile devices and other Internet of Things (IoTs) via software defined radios (SDRs) herein called virtual base stations (BSs). In 5G architecture, one SDR/BS covers a small cell; therefore, multiple BSs can connect to one nearby edge-computing infrastructure via high speed and low-latency software defined fiber links [14]. For the beyond 5G deployments, mobile network operators (MNO) rely on connected EC and BC for

Fig. 1: System model and Structure of Edge-cloud based Streetscape System.

scalability and enhanced services [16]. Therefore, it is vital to consider EC along with BC in order to propose a realistic system. In our proposed system, EC is interconnected with the BC via high speed fiber links to leverage abundant and always available resources at public clouds (AWS, Google Compute Engine, Microsoft Azure) in order to offload delay-tolerant multimedia tasks or to save data for future uses.

Context aware control of resources is main ingredient of the smart streetscape environment such as smart control of pedestrian signal for elderly people and traffic signal control for emergency situations (e.g. fire on a building) or in a logistics unloading case, traffic can automatically be diverted to a safer and smoother street with the help of data sent by the IoTs [17].

Multi-media (AR, VR, Video) applications are bandwidth hungry and resource intensive, at the same time require low end-to-end latency [1]. In such scenarios, proposed EC infrastructure plays an important role to supply uninterrupted services to both streetscape and multi-media applications. Upon arrival of a service request, based on it's exigency, the system decides whether to run it on EC or BC, and based on the availability of resources how much bandwidth and compute resources have to be allocated in order to execute the task within the deadline.

B. Users and Jobs Model

In the proposed framework, we consider all the devices which are connected to the system as EC users. Each user offloads its computational task in the form of a distinct service request. The entire workload of the system is a set of J jobs from U users i.e. $J = \{j_1(dl_1, d_1), j_2(dl_2, d_2), ..., j_U(dl_U, d_U)\}$. A job $j_u(dl_u, d_u)$ is a tuple of two variables where dl_u means the hard deadline in milliseconds and d_u represents the data in bytes to be processed in job j_u offloaded by user u. The user job can demand both CPU and Memory for a successful execution but processing vitally involves CPU; therefore, we only consider CPU cycles as a job processing source as proposed by Cheng *et al.* [9]. Suppose $\mathbb{C}$ represents the number of CPU cycles required to process 1 Byte of data, then L_u is given as the total CPU cycles required to compute data d_u ($L_u = d_u \times \mathbb{C}$). A similar task computation model (with CPU cycles) is proposed in [12], [13] as well.

C. Wireless Bandwidth Model

As shown in Fig. 1, an EC system owns W base stations (BSs) $\{1, 2, 3, ..., W\}$, each of which makes meshed wireless connections with actuators/relays to provide wireless access capacity, load-balancing and failover. The total wireless bandwidth that is available on all BSs is represented as B. Each BS w can support H wireless bandwidth channels $\{ch_1^w, ch_2^w, ch_3^w, ..., ch_H^w\}$, and each wireless bandwidth channel ch_h^w ($h \in [1, H]$) provides a variable amount of bandwidth (data rate in bps) and costs c_h^w. The directional antenna array in mmWave cellular networks of the COSMOS testbed [14] is capable of exploiting beamforming, which compensates the increased path-loss at mmWave frequencies and overcomes the additional noise due to the large transmission bandwidth [5]. Interference isolation is also achieved in directional beamforming, which, as a result, reduces the adjacent-cell interference. Therefore, in our case, we ignore path-loss, channel noise and interference factors, and manage resources at the application layer through well-defined APIs [3], which provides a fine-grained control of the wireless bandwidth.

D. Computational Model

1) Edge Cloud: An Edge-cloud (EC) owns M servers $\{1, 2, 3, ..., M\}$. Each server m processes an offloaded job via a set of virtual machines (VMs). Let $K = \{vm_1^m, vm_2^m, vm_3^m, ..., vm_K^m\}$ be the set of VMs that can be assigned by server m and each VM vm_k^m provides a variable amount of compute capacity (CPU cycles in Hz) to process a job and costs c_k^m. The total compute capacity on all EC servers is given by C_{ec} CPU cycles. Chen *et al.* [13] proposes a similar computational model; however, they consider EC with unlimited compute resources, which may not be valid for practical scenarios.

In practice, a fine-grained control of the compute resources can be achieved at the application layer by processing user jobs in a Docker container, which uses cgroups to limit the system resources. However, for the simplicity of the model, we will use the term VM in this study.

2) Back-end Cloud: Previous work [18] considered back-end cloud (BC) as a source of unlimited compute capacity. However, we take into account a realistic model to minimize the overall cost of the system by adopting pay-as-you go model. Therefore, just like an EC, we consider N BC servers $\{1, 2, 3, ..., N\}$, which execute a job via a set of $K = \{vm_1^n, vm_2^n, vm_3^n, ..., vm_K^n\}$ VMs, each with a variable amount of compute capacity (CPU cycles) and costs c_k^n. C_{bc} denotes the total number of CPU cycles available on all BC servers. This model can be easily extended to infinite resource model by relaxing C_{bc} and K sufficiently large.

E. Delay Model

We define round-trip time (RTT) as the total time that it takes for a job to upload to the EC or BC via a wireless channel, process the job and then send the result back. This involves propagation, transmission (2-way) and processing time. A similar delay model is also used in [10].

1) Propagation Time: Propagation time through fiber or air media is negligible and assumed to be constant. Therefore, we consider a constant delay $t_u(prop_{ec}) = 5ms$ for EC and $t_u(prop_{bc}) = 50ms$ for BC depending on where the resources are allocated for job execution.

2) Transmission Time: This includes the time that a job takes to upload to the EC or BC and the time to send the result back successfully. It depends upon the amount of data and wireless channel that is allocated. The transmission time of a job to the EC can be calculated as: $t_u(trans_{ec}) = \frac{d_u}{ch_h^w} + \frac{R_u}{ch_h^w}$, where R_u is the result which is generated after the job execution and sent back. In general, the result is a control signal and contains only a few kilobytes of data [13]. Nonetheless, the result for AR/VR applications can be substantially large, therefore, we incorporate it in our framework.

According to Fig. 1, EC is connected with BC via a high capacity fiber link and considered to guarantee b bandwidth [14]. Thus, the transmission time between a device and BC can be given as: $t_u(trans_{bc}) = t_u(trans_{ec}) + \frac{d_u}{b} + \frac{R_u}{b}$.

3) Processing Time: This depends upon the number of required CPU cycles L_u and the allocated VM vm_k^m, vm_k^n at EC or BC, respectively, and given as: $t_u(proc_{ec}) = \frac{L_u}{vm_k^m}$ for the EC, and $t_u(proc_{bc}) = \frac{L_u}{vm_k^n}$ for the BC.

To summarize, when a job j_u is offloaded to the EC, the total round-trip time is given as: $rtt_{ec}(j_u) = t_u(prop_{ec}) + t_u(trans_{ec}) + t_u(proc_{ec})$, similarly, when the job is offloaded to the BC then: $rtt_{bc}(j_u) = t_u(prop_{bc}) + t_u(trans_{bc}) + t_u(proc_{bc})$.

F. Utility Model

The total usage of the system resources at any given time t is the sum of the occupied resources by all the jobs which are being processed. Therefore, the bandwidth utility rate of all base stations W is given as:

$$Ur_W(t) = \frac{\sum_{w=1}^{W}(\sum_{h=1}^{H} ch_h^w.\mu_h^w(t))}{B}, \tag{1}$$

where $\mu_h^w(t)$ is the total number of ch_h^w channels which are serving the jobs at time t. Similarly, the utility rate of a server at EC and BC can be measured as:

$$Ur_M(t) = \frac{\sum_{m=1}^{M}(\sum_{k=1}^{K} vm_k^m.\mu_k^m(t))}{C_{ec}}, \tag{2}$$

and

$$Ur_N(t) = \frac{\sum_{n=1}^{N}(\sum_{k=1}^{K} vm_k^n.\mu_k^n(t))}{C_{bc}}, \tag{3}$$

respectively.

III. The Proposed DDPG-Edge-Cloud Agent

In this section, we present DDPG-Edge-Cloud to solve the MRA problem for mobile and streetscape based applications. Like RL agent in [15], the proposed DDPG-Edge-Cloud agent also runs on the EC for better accessibility of the environment and faster training. In our DDPG-based framework, the agent contains actor and critic networks whose architectures are same. The actor, i.e., a policy function, observes the state s_t by interacting with the environment and takes a continuous action a_t via a deterministic policy and receives an immediate reward r_t. On the other hand, the critic uses the action-value function $Q(s_t, a_t)$ to update the policy parameters. At each state, different resource allocation actions yield different rewards. The goal of the agent is to maximize the long-term reward by finding an optimal resource allocation policy. We define state and action space followed by our unique reward model below.

1) State Space: The state of the system at any time t is the observation of utility rates of all the base stations, EC and BC servers, and the current job j_u which has to be scheduled either at EC or BC. The state contains five parameters, i.e. $s_t = \{Ur_W, Ur_M, Ur_N, j_u(dl_u, d_u)\}$. Based on these sequences, the agent learns optimal resource allocation strategies in each decision epoch.

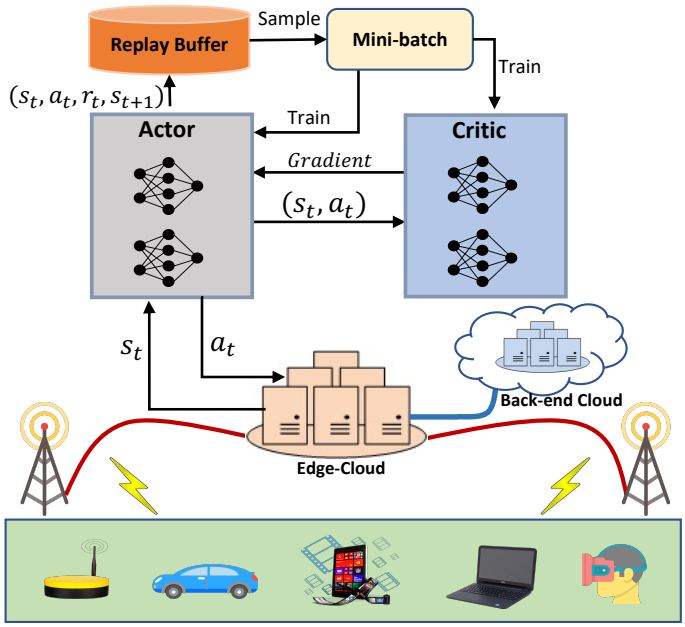

Fig. 2: The architecture of the proposed DDPG-Edge-Cloud framework.

2) Action Space: In each state s_t, the agent decides how much bandwidth and VM resources (on EC or BC) have to be allocated in order to successfully execute the job within the deadline. The continuous action space, $a_t = \{ch_h, vm_k, cloud\}$, has three parameters. ch_h is the amount of bandwidth, vm_k is the amount of CPU cycles and $cloud$ has value either 0 or 1 which means task is offloaded to EC or BC, respectively.

3) Reward: In our model, the reward can be calculated as the sum of lump income and cost of resource procurement in the MRA system:

$$r_t(s_t, a_t) = I_u(s_t, a_t) - cost_u(s_t, a_t) \times rtt(j_u), \quad (4)$$

where $I_u(s_t, a_t)$ is the net income earned by executing the job j_u and $cost_u(s_t, a_t)$ is the cost of resource procurement for $rtt(j_u)$ time (time it takes to process the job) for selecting action a_t at state s_t. In the definitions of $I_u(s_t, a_t)$ and $cost_u(s_t, a_t)$, we take into account the completion time of the job and the utility of wireless and computation resources.

$I_u(s_t, a_t)$ is defined as,

$$I_u(s_t, a_t) = \begin{cases} (dl_u - rtt_{ec}(j_u)).\delta, \; if \; cloud_t = 0 \\ (dl_u - rtt_{bc}(j_u)).\delta, \; if \; cloud_t = 1. \end{cases} \quad (5)$$

In Eq. (5), δ represents the revenue that the service provider generates by successfully executing the user job. It can be set on-the-fly which can differ depending on the job completion time. Note that, if the completion time of job ($rtt(j_u)$) exceeds the deadline (dl_u), then the income becomes negative which impacts the overall reward. This effect encourages the system to take the allocation decisions that can complete the jobs before deadlines.

In contrast to the income, $cost_u(s_t, a_t)$ describes the cost of resource procurement per unit time by allocating wireless bandwidth channel and a VM at EC or BC and is given as,

$$cost_u(s_t, a_t) = \begin{cases} Ur_W(t).c_h + Ur_M(t).c_k, \; if \; cloud_t = 0 \\ Ur_W(t).c_h + Ur_N(t).c_k, \; if \; cloud_t = 1, \end{cases} \quad (6)$$

where c_h and c_k represent the cost of wireless bandwidth channel and VM allocation per unit time, respectively.

As compared to [18], we calculate the reward for each individual job, so as the cost and the income. This approach is more meaningful in a way that the QoS requirement in each job may vary and calculating reward for every individual job can better assist the agent to derive an optimal resource allocation policy.

The optimization problem of wireless bandwidth and VM allocation for the user jobs at different base stations and servers, while considering the varying QoS requirements and constrained resources is formulated as below:

$$maximize \sum_{t=1}^{T} r_t(s_t, a_t) \quad (7)$$

subject to the constraints $Ur_W(t) \leq 1$, $Ur_M(t) \leq 1$ and $Ur_N(t) \leq 1, \forall t \in T$, which describes that the utility of bandwidth and EC and BC servers does not exceed from it's total available capacity, respectively. The constraint $rtt_{ec}(j_u) \leq dl_u, rtt_{bc}(j_u) \leq dl_u, \forall u \in U$ guarantees the hard deadline requirement of the job offloaded at EC or BC, respectively.

A. Actor-Critic Framework

In traditional DDPG-based framework, fully-connected networks (FCNs) are mostly used as both actor and critic networks [19], which have huge trainable weights and capture only global descriminative features of the task sequences. However, the computation-intensive tasks have complex temporal variations in nature. For high-quality state representation and better function approximation of MRA system, we propose a network to capture the local and temporal features in sequential data. Inspired from [13], the first part of our proposed agent contains Conv1D residual block structure to learn the correlations among local features of each input state. The second part contains GRU to learn the temporal dependencies and an attention mechanism to capture meaningful information at certain moments that has decisive effect on prediction. GRU model has fewer parameters and controls the flow of the information without using a memory unit, resulting in less complicated structure with the performance on par with LSTM [20]. This is why we prefer GRU over LSTM.

To break the undesired temporal correlations of training samples and reduce variance, an experience replay (ER) is used to store all the experience $((s_t, a_t, r_t, s_{t+1}))$ to train the agent on more independent samples. Uniform sampling is used to randomly select a mini-batch of transitions from the ER buffer to train the actor and critic networks.

1) Pruning Principle: Our proposed agent is responsible to select appropriate amount of bandwidth, VM units and cloud to execute a job. We introduce pruning principle to further select the base station and server with the least utility. The BS is given as $w_t = \arg\min(Ur_w(t)), \forall t \in T, \forall w \in W$, the EC server is given as $m_t = \arg\min(Ur_m(t)), \forall t \in T, \forall m \in M$, and the BC server (if task offloaded to BC) $n_t = \arg\min(Ur_n(t)), \forall t \in T, \forall n \in N$. The major contribution of our proposed pruning principle is the reduction of action space at every state by a significant amount. It also helps to balance the load among all the base stations and servers; which means, it does not lead to overload a single base station or server which could potentially result in higher rejection rate for future jobs.

The training process of DDPG-Edge-Cloud agent with pruning principle to obtain an optimal MRA policy is summarized in Algorithm 1. The target networks of actor and critic are clone of their respective online networks.

Algorithm 1: Training process of DDPG-Edge-cloud agent with Pruning Principle

Input : User jobs with varying QoS requirements
Initialize replay memory Δ to capacity Ω;
Initialize the actor and critic online and target networks with random weights;
for *episode = 1 to E* **do**
  Reset the environment;
  for *t = 1 to T* **do**
    Predict an action a_t using the actor network;
    Apply pruning principle to select base station and server
    Execute action a_t, observe next state s_{t+1} and receive reward r_t
    Store transition sample (s_t, a_t, r_t, s_{t+1}) in Δ
    Sample random mini-batch of transitions from Δ
    Train the critic and actor on sampled mini-batch
    Update the weight vectors of online networks in actor and critic
    Update the weight vectors of target networks in actor and critic
    $s_t \leftarrow s_{t+1}$
  end
end
Output: Optimal Multi-Resource Allocation policy

IV. Experimental Results

We develop our MRA framework and proposed DDPG-Edge-Cloud agent with pruning principle in the Python 3.8.10 to simulate a near real-world environment. The simulation code will be made public after the community release of COSMOS testbed [14]. We run all the experiments on Dell Desktop Machine with 2.9 GHz Intel Core i7 processor, 128GB memory and Windows 10 Pro 64-bit OS, and discuss the advantages of our proposed algorithm over the alternative methods.

A. Experiment Setup

The two baselines we use to compare with the DDPG-Edge-Cloud agent are described below:

- **DDPG-NN:** Existing DDPG [19] with two fully-connected network layers used in the actor and critic networks. Same uniform sampling replay buffer is used for a fair comparison.
- **DDPG-CNN:** In a DDPG-based actor and critic networks, the fully-connected layers are replaced by identical Conv1D residual blocks introduced in Section III-A. As opposed to pruning principle in our proposed method, base stations and servers are randomly selected in the case of these agents.

We perform the experiment on three different sizes of environments, small, medium and large. Small-scale environment contains 4 wireless base stations, 10 EC and 10 BC servers. Medium-scale environment contains 12 base stations, 30 servers at the EC and 30 at the BC. The large-scale environment consists of 20 base stations, 50 EC and 50 BC servers. We set each base station to provide $1Gbps$ of total bandwidth, each server with 18 cores and each core with 2GHz (total 36GHz) both in the EC and BC servers. The bandwidth (b) between EC and BC set to $1Gbps$. The configurations of the proposed system can be customized; however, here our objective is to compare the performance of three algorithms.

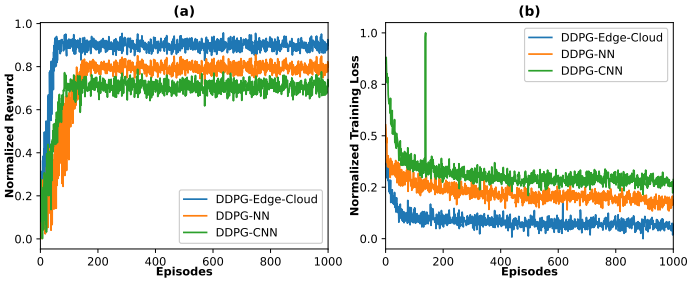

Fig. 3: Convergence comparison of three agents in small-scale environment. (a) Normalized Reward. (b) Normalized training loss.

We conduct experiments on small-scale, medium-scale and large-scale environments with 10,000, 100,000 and 1,000,000 jobs, respectively. The wireless bandwidth and VM per unit rental/usage cost is normalized to 1 cent per second. The value of revenue δ is set to 10 cents. Mobile users and other IoT devices generate tasks of different sizes with varying QoS demands. We group these user tasks/jobs into two groups; 1) Type-1 tasks are critical tasks with comparably small size (bytes) and deadline. We set the deadline of such tasks to be in the range of [200ms, 800ms], and the size to be in the range of [100KB, 500KB]; 2) Type-2 tasks are multi-media tasks with large size and relaxed deadline as compared to the Type-1 tasks. The deadline of such tasks is set between [1000ms, 2000ms], and the size between [1MB, 2MB]. CPU cycles required to process 1-byte of data is randomly generated between [1000, 4000] and the resultant data is also randomly generated between [500, 1000] in KBs.

The default learning rate is set to 0.0001 and 0.001 for actor and critic networks, respectively, and the discount factor γ = 0.99. The Adam optimizer is used to optimize the loss function during training. The learning iterations (T) per episode are set to 1000.

B. Convergence

We compare the convergence rate of DDPG-Edge-Cloud with DDPG-NN and DDPG-CNN in a small-scale environment to get the intuition of the performance of the agents. Fig. 3(a) and Fig. 3(b) depict the convergence rate in terms of reward and training loss which are normalized using max-min normalization method. Initially, DDPG-CNN grows up quickly as compared to DDPG-NN; however, due to the nature of local feature extraction only, the DDPG-CNN agent is trapped in local optima. Our proposed agent learns both local and long-term features, which results in efficient training and superiority in convergence.

C. Performance Analysis

Performance comparison is based upon three key performance indicators (KPIs) in the MRA system: operational cost, rejection rate and QoE. The operational cost is calculated using Eq. (6). A job is considered rejected if its completion time exceeds the given deadline or no more resources are available for allocation. Both the cost and rejection rate are averaged over the total number of accepted jobs in the respective scale. In our environment, QoE is inversely proportional to the round-trip time (RTT) of the job. This means, smaller RTT of a job will induce better QoE for the users. Fig. 4(a) illustrates the average operational cost, our proposed agent, on average, achieves 30.5%, 42% and 55% reduction in cost in small, medium and large-scale environments, respectively. Fig. 4(b) shows that our proposed strategy consistently gives rejection rate a multiple of 10^{-3} even in the large scale environment.

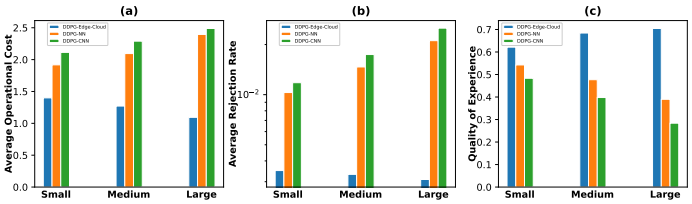

Fig. 4: Performance comparisons of three agents in small, medium and large-scale environments. (a) Average operational cost, (b) Average rejection rate, (c) Quality of Experience.

Moreover, our proposed agent, on average, achieves 68%, 79% and 86.5% reduction in rejection rate in small, medium and large-scale environments, respectively. The improvement of this magnitude is due to the virtue of our proposed pruning principle, which evenly distributes the load among all base stations and servers, and minimizes the probability of rejection for the future jobs. In view of Fig. 4(c), our proposed agent achieves nearly 22%, 60% and 115% gain in QoE in small, medium and large-scale environments, respectively.

To summarize, our proposed DDPG-Edge-Cloud adaptively determines the dynamic selection of VMs, wireless bandwidth channels and cloud for joint optimization of resources in the EC and BC. Moreover, our proposed pruning principle helps reduce the rejection rate significantly. Overall, our proposed framework outperforms the alternative agents in all three environments.

V. Conclusion and Future Work

We present DDPG-Edge-Cloud, a deep deterministic policy gradient-based multi-resource allocation (MRA) framework. The MRA system is utilized to optimize the problem of compute and wireless resources for the IoTs and mobile users in a smart streetscape based edge-cloud (EC) and back-end cloud environment. The proposed DDPG-Edge-Cloud agent is equipped with Conv1D residual block, gated recurrent unit (GRU) layer and an attention layer. The agent runs in the EC to make dynamic resource allocation decisions for the user tasks. The presented algorithm learns the local and long-term temporal features from the sequential data and outperforms the alternative methods in convergence speed. DDPG-Edge-Cloud achieves up to 55% reduction in operational cost on average. The proposed pruning principle helps our agent to achieve up to 86.5% reduction in rejection rate on average. Further, the proposed agent achieves up to 115% gain in the QoE of the users.

In future, we plan to explore priority-based replay buffer techniques where priority will be calculated using a heuristic function, which will help further boost up the convergence rate. Moreover, to cope with the overwhelming volume of accumulated data from numerous IoTs, we plan to investigate on data parallelism techniques for training in a distributed fashion to accelerate the learning process.

Acknowledgment

This work is supported by NSF PAWR (#1827923) and NSF IRNC (#2029295).

References

[1] Y. Liu *et al.*, "Toward Edge Intelligence: Multiaccess Edge Computing for 5G and Internet of Things," in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6722-6747, Aug. 2020, doi: 10.1109/JIOT.2020.3004500.

[2] Z. Ning *et al.*, "Green and Sustainable Cloud of Things: Enabling Collaborative Edge Computing," in IEEE Communications Magazine, vol. 57, no. 1, pp. 72-78, January 2019, doi: 10.1109/MCOM.2018.1700895.

[3] O. -M. Ungureanu *et al.*, "Collaborative Cloud - Edge: A Declarative API orchestration model for the NextGen 5G Core," 2021 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2021, pp. 124-133, doi: 10.1109/SOSE52839.2021.00019.

[4] S. Gong *et al.*, "Adaptive Resource Allocation of Multiple Servers for Service-Based Systems in Cloud Computing," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, 2017, pp. 603-608.

[5] M. A. Habibi *et al.*, "A Comprehensive Survey of RAN Architectures Toward 5G Mobile Communication System," in IEEE Access, vol. 7, pp. 70371-70421, 2019, doi: 10.1109/ACCESS.2019.2919657.

[6] L. Lei *et al.*, "Deep Reinforcement Learning for Autonomous Internet of Things: Model, Applications and Challenges," in IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1722-1760, thirdquarter 2020, doi: 10.1109/COMST.2020.2988367.

[7] X. Wang *et al.*, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," in IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 869-904, Secondquarter 2020, doi: 10.1109/COMST.2020.2970550.

[8] A. Feriani *et al.*, "Single and Multi-Agent Deep Reinforcement Learning for AI-Enabled Wireless Networks: A Tutorial," in IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 1226-1252, Secondquarter 2021, doi: 10.1109/COMST.2021.3063822.

[9] M. Cheng *et al.*, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 2018, pp. 129-134, doi: 10.1109/ASPDAC.2018.8297294.

[10] Y. Wei *et al.*, "Joint Optimization of Caching, Computing, and Radio Resources for Fog-Enabled IoT Using Natural Actor–Critic Deep Reinforcement Learning," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2061-2073, April 2019, doi: 10.1109/JIOT.2018.2878435.

[11] H. Peng *et al.*, "Deep Reinforcement Learning Based Resource Management for Multi-Access Edge Computing in Vehicular Networks," in IEEE Transactions on Network Science and Engineering, vol. 7, no. 4, pp. 2416-2428, 1 Oct.-Dec. 2020, doi: 10.1109/TNSE.2020.2978856.

[12] S. Nath *et al.*, "Dynamic Computation Offloading and Resource Allocation for Multi-user Mobile Edge Computing," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9348161.

[13] J. Chen *et al.*, "A DRL Agent for Jointly Optimizing Computation Offloading and Resource Allocation in MEC," in IEEE Internet of Things Journal, vol. 8, no. 24, pp. 17508-17524, 15 Dec.15, 2021, doi: 10.1109/JIOT.2021.3081694.

[14] D. Raychaudhuri *et al.*, 2020. Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless. Proceedings of the 26th Annual International Conference on Mobile Computing and Networking. Association for Computing Machinery, New York, NY, USA, Article 14, 1–13. DOI:https://doi.org/10.1145/3372224.3380891

[15] A. Qadeer *et al.*, "Flow-Level Dynamic Bandwidth Allocation in SDN-Enabled Edge Cloud using Heuristic Reinforcement Learning," 2021 8th International Conference on Future Internet of Things and Cloud (FiCloud), 2021, pp. 1-10, doi: 10.1109/FiCloud49777.2021.00009.

[16] Arno, H. Van, Mazur, M.: Telecom infrastructure the open source way.Technical report, Ubuntu, Canonical, United Kingdom (October 2021)

[17] J. Yang *et al.*, "Regional Smart City Development Focus: The South Korean National Strategic Smart City Program," in IEEE Access, vol. 9, pp. 7193-7210, 2021, doi: 10.1109/ACCESS.2020.3047139.

[18] Y. Liu *et al.*, "Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System," in IEEE Transactions on Mobile Computing, vol. 15, no. 10, pp. 2398-2410, 1 Oct. 2016, doi: 10.1109/TMC.2015.2504091.

[19] Chen, Z. *et al.*, Decentralized computation offloading for multi-user mobile edge computing: a deep reinforcement learning approach. J Wireless Com Network 2020, 188 (2020). https://doi.org/10.1186/s13638-020-01801-6

[20] S. Gao *et al.*, "Short-term runoff prediction with GRU and LSTM networks without requiring time step optimization during sample generation," Journal of Hydrology, Volume 589, 2020, 125188, ISSN 0022-1694, https://doi.org/10.1016/j.jhydrol.2020.125188.