# Learning-Based Task Offloading for Mobile Edge Computing

Rim Garaali *, Cirine Chaieb*, Wessam Ajib*, and Meriem Afif †

* Department of Computer Science, University of Quebec at Montreal, QC, Canada
† University of Carthage, National Institute of Applied Sciences and Technology, Tunis, Tunisia
*garaali.rim@courrier.uqam.ca, {chaieb.cirine, ajib.wessam}@uqam.ca
† meriem.afif@insat.ucar.tn

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838831

***Abstract*—Mobile edge computing (MEC) is an important technology for latency-sensitive applications. One of the biggest challenges in MEC is efficiently allocating resources under strict QoS requirements and resource constraints. The purpose of this paper is to study the joint problem of computation offloading and resource allocation in such networks. The problem is formulated as a mixed-integer non-convex optimization problem and is proved to be $\mathcal{NP}$-hard. In order to solve it efficiently, we propose a multi-agent deep reinforcement learning solution based on actor-critic method. To reduce system latency, each agent aims to learn interactively the best offloading policy independently of other agents. The simulation results illustrate the performance and advantages of the proposed solution compared to benchmark solutions.**

***Index Terms*—Mobile edge computing, task offloading, deep reinforcement learning, actor-critic algorithm.**

## I. Introduction

With the rapid enhancement of networking and the diverse new applications of wireless access technology, a massive number of smart wirelessly-connected devices are expected to emerge in the future generation of wireless networks. However, the computational resources of wireless devices are expected to be insufficient to meet the stringent requirements of latency-sensitive applications. Recently, mobile edge computing (MEC) and computation offloading techniques have received a significant attention to overcome the limitations of low-computation capability and low-power mobile devices [1], [2].

In [3], the authors studied the problem of offloading decision in a device-to-device (D2D) communication-enabled cellular network in order to maximize the sum of user's utility while minimizing energy consumption, task processing delay, task loss probability, and costs. They proposed a deep Q-learning (DQL) solution where the tasks of mobile users can be executed locally or offloaded to nearby mobile cloud-lets via D2D communications. In [4], a multi-agent reinforcement learning (RL) approach was developed to reduce the computation delay and to improve the channel access success rate in an industrial internet of things environment. The authors proved the effectiveness of their proposed solution compared to traditional single-agent RL approach. Moreover, the authors in [5] investigated deep RL (DRL) approach to jointly solve the problem of sub-carrier assignment and power allocation where the objective is to improve the spectrum efficiency and the system capacity in D2D networks. In [6], the authors studied the task scheduling problem and proposed an actor-critic algorithm to reduce the task scheduling delay in cloud/edge networks. In [7], the authors studied the caching problem while considering time-varying channels and D2D interference. They proposed a DRL solution to maximize the overall effective throughput of D2D pairs.

The non-orthogonal multiple access (NOMA) technique is facing many new challenges in MEC despite computation offloading. In [8], the authors investigated the joint power control and channel allocation problem with two different matching schemes (single NOMA and group matching NOMA) in NOMA-based MEC. The authors proved that NOMA technique outperforms traditional orthogonal multiple access (OMA) technique in terms of energy consumption and delay. In [9], the authors tackled the joint optimization problem of computation and radio resource allocation in NOMA-based MEC in order to minimize the overall delay. The authors proposed two efficient algorithms. The first one aims to find the optimum grouping among different servers and the second one aims to find the Nash equilibrium. In [10], the authors proposed a single-agent QL solution to reduce energy consumption by jointly optimizing task offloading, computation resources, and caching in D2D communication-enabled cellular networks. In [11], the authors proposed a task offloading decision based on DQL algorithm to reduce the executing computation delay while taking into account the time-varying channel conditions in OMA-based networks. Nevertheless, the authors in [12] studied the task offloading in NOMA-based networks. They tackled the joint optimization problem of computation offloading and resource allocation to minimize the system delay. They proposed a DRL approach that can fit with dynamic networks and time-varying channel conditions.

In this paper, we propose and evaluate an efficient distributed DRL-based computation offloading and resource allocation solution in MEC wireless networks. The objective of the proposed solution is to enhance computing performance in multi-user NOMA-based MEC networks.

Hereinafter, the notations $\underline{z}$, $\mathbf{Z}$ and $\mathcal{Z}$ are used to denote vectors, matrices and sets, respectively. By $\underline{z}^{\intercal}$ we refer to the



Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 08,2023 at 16:24:53 UTC from IEEE Xplore. Restrictions apply.

transpose of $\underline{z}$ and by $|z|$ we refer to the modulus of $z$.

The rest of this paper is organized as follows. Section II presents the system model. Section III formulates mathematically the delay minimization problem and proves its $\mathcal{NP}$-hardness. Section IV presents the proposed DRL-based computation offloading and resource allocation solution, and section V illustrates the performance of the proposed DRL solution compared to benchmark ones. Finally, section VI draws some conclusions.

## II. System Model

As shown in Fig. 1, we consider a MEC network composed of $U$ edge-users (EUs) and $C$ cellular-users (CUs). In this paper, we focus on data partitioned oriented applications. Indeed, the computation task of various applications such as file compression, video compression, face detection and speech recognition can be partitioned into independent sub-tasks [13]. The EUs can offload their computation tasks to the $E$ edge servers (ESs), whereas the CUs transmit wirelessly to one of the $B$ base stations (BSs). We assume that each EU has independent tasks that can be executed either locally or remotely in one or more ESs by offloading sub-tasks. Also, EUs can share frequency resources with only one CU and the ESs are far away from each other so the interference between ESs is negligible. Finally, we assume that the time is divided into slots and the channel gain conditions between EUs-ESs and between CUs-BSs are static during one time slot transmission.

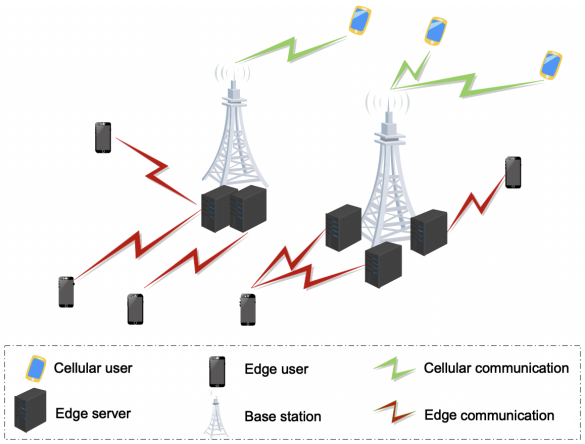


Fig. 1 – System Model.

### A. Communication Model

This paper investigates up-link NOMA and successive interference cancellation (SIC) techniques to sequentially decode the received signals to provide more efficient resource utilization. Assuming that the EUs are sorted in descending order according to their channel gain conditions, the received signal to interference plus noise ratio (SINR) from the $u$th EU to the $e$th ES is given as follows:

$$SINR_{u,e} = \frac{p_{u,e}|h_{u,e}|^2}{\sum_{i=u+1}^{U} p_{i,e}|h_{i,e}|^2 + p_c|h_{c,e}|^2 + \sigma_e^2}, \quad (1)$$

where $\sigma_e^2$ is the power of the additive white Gaussian noise, $p_{u,e}$ is the transmit power from the $u$th EU to the $e$th ES, $p_c$ is the transmit power of the interfering $c$th CU, and $h_{c,e}$ is the channel gain between the $c$th CU and the $e$th ES. Let $B$ denotes the available bandwidth, the up-link data rate $R_{u,e}$ is given by:

$$R_{u,e} = B \log_2(1 + SINR_{u,e}). \quad (2)$$

After processing sub-tasks remotely, each ES $e$ transmits the computation results to the EUs using OMA technique. The feedback delay $T_{e,u}^{\text{fb}}$ is given by:

$$T_{e,u}^{\text{fb}} = \frac{m_{e,u}^{p}}{R_{e,u}} x_{u,e}, \quad (3)$$

where $m_{e,u}^{p}$ is the output data size of user $u$ after being processed in the ES $e$ and $R_{e,u}$ is the down-link data rate which is expressed as follows:

$$R_{e,u} = \frac{B}{U} \log_2(1 + \frac{p_{e,u}|h_{e,u}|^2}{\sigma_e^2}). \quad (4)$$

### B. Computing Model

When the task is executed locally, it is denoted by a pair $\tau_u = (C_u, M_u)$, where $C_u$ is the number of central processing unit (CPU) cycles and $M_u$ is the task size. The local task computation delay is expressed by:

$$T_{\tau_u}^{\text{cp}} = \frac{C_u}{F_u}, \quad (5)$$

where $F_u$ is the available processing capacity at EU $u$.

Let $\underline{\lambda}_u = [\lambda_{u,1}, \lambda_{u,2}, \cdots, \lambda_{u,e}]^\intercal$ denotes the sub-task allocation vector of EU $u$ where $\lambda_{u,e} \in [0, 1]$ is the proportion of data size offloaded to ES $e$. The offloaded sub-task size, $m_{u,e}$, is expressed by [14]:

$$m_{u,e} = \beta_{u,e} \lambda_{u,e} M_u, \quad (6)$$

where $\beta_{u,e}$ is the transmitted data size ratio. The number of CPU cycles transmitted from EU $u$ to ES $e$ is $C_{u,e} = c m_{u,e}$, where $c$ is the required number of CPU cycles for one data bit [15] [16]. The transmission delay from EU $u$ to ES $e$ is given by:

$$T_{u,e}^{\text{tx}} = \frac{m_{u,e}}{R_{u,e}} x_{u,e}. \quad (7)$$

We denote by $\tau_{u,e} = (C_{u,e}, m_{u,e})$ the sub-task of EU $u$ executed at ES $e \in \mathcal{E}_u$, where $\mathcal{E}_u$ is the set of ESs to which EU $u$ offloads its sub-tasks. When a sub-task is offloaded to ES $e$, its computation delay at $e$ is expressed as in:

$$T_{u,e}^{\text{cp}} = \frac{C_{u,e}}{F_{u,e}} x_{u,e}, \quad (8)$$

where $F_{u,e}$ is the allocated computation resources by the $e$th ES, $x_{u,e}$ is a binary association variable to indicate whether


Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 08,2023 at 16:24:53 UTC from IEEE Xplore. Restrictions apply.

EU $u$ is associated to ES $e$ or not, and finally $\mathbf{X} = [x_{u,e}]$ is the association matrix.

Consequently, the execution delay can be calculated as the sum of transmission delay, computation delay, and feedback delay. The task execution delay of EU $u$ is calculated as follows:

$$T_u^{\text{ex}} = \max_{e \in \mathcal{E}_u \cup \{u\}} (T_{u,e}^{\text{tx}} + T_{u,e}^{cp} + T_{e,u}^{\text{fb}}). \tag{9}$$

It is to be noted that $T_{u,e}^{\text{tx}} = T_{e,u}^{\text{fb}} = 0$ if the task is completely executed locally, whereas $T_{\tau_u}^{\text{cp}} = 0$ if the task is totally offloaded to ESs.

## III. Problem Formulation

This section starts by formulating mathematically the problem and then proves its $\mathcal{NP}$-hardness.

### A. Problem formulation

Let $\mathbf{M} = [m_{u,e}]$ and $\mathbf{P} = [p_{u,e}]$ denote respectively the task size and the allocated power level matrices, the execution delay minimization problem is formulated as follows:

$$\underset{\mathbf{X},\mathbf{M},\mathbf{P}}{\text{minimize}} \frac{1}{U} \sum_{u=1}^{U} T_u^{ex} \tag{P1a}$$

$$\text{subject to } x_{\tau_u,e}, x_{u,e} \in \{0,1\}, \forall u \in \mathcal{U}, \forall e \in \mathcal{E}_u, \forall \tau_u, \tag{P1b}$$

$$\sum_{e \in E} x_{\tau_u,e} \leq 1, \forall u \in \mathcal{U}, \forall \tau_u, \forall \tau_u, \tag{P1c}$$

$$p_{u,e} > 0, \forall u \in \mathcal{U}, \forall e \in \mathcal{E}_u, \tag{P1d}$$

$$\sum_{e \in \mathcal{E}} p_{u,e} x_{\tau_u,e} \leq p_u^{max}, \forall u \in \mathcal{U}, \tag{P1e}$$

$$R_{u,e} \geq R_{th_e}, \forall u \in \mathcal{U}, \forall e \in \mathcal{E}_u, \tag{P1f}$$

$$\sum_{u=1}^{U} F_{u,e} x_{u,e} \leq F_e, \forall e \in \mathcal{E}, \tag{P1g}$$

$$\sum_{e=1}^{E} m_{u,e} x_{u,e} = M_u, \forall u \in \mathcal{U}. \tag{P1h}$$

Constraints (P1b) ensure that $x_{\tau_u,e}$ and $x_{u,e}$ are binary. Constraints (P1c) ensure that each EU's sub-task is executed by only one ES. Constraints (P1d) retain the positivity of the transmit power variables. Constraints (P1e) represent the power limit of each EU $u$. Constraints (P1f) guarantee a given data rate threshold $R_{th_e}$. Constraints (P1g) ensure that the total allocated computation resources by ES $e$ does not exceed its maximum computation resources. Finally, constraints (P1h) guarantee the integrity of the executed sub-tasks.

### B. Problem Complexity

In the following, the $\mathcal{NP}$-hardness of (P1) is demonstrated by restriction. That is, a special case of (P1) is proved to be $\mathcal{NP}$-hard in a polynomial-time.

*Lemma 1:* Problem (P1) is $\mathcal{NP}$-hard.

*Proof:* The $\mathcal{NP}$-hard proof reduces the Knapsack problem (KP), a known $\mathcal{NP}$-hard problem [17], to (P1).

In KP, we are given a set of items $i$ and a knapsack with capacity $c$ and for each item $i$, there is a weight $w_i$. The objective of KP is to maximize the sum of the values of the items in the knapsack so that the sum of the weights is less than or equal to $c$. Given an instance of KP, an instance of (P1) can be constructed as following: let $E = 1$, $p_{u,e} = p_u^{max}$, and $\forall u \in \mathcal{U}$ $M_u = 1$ task that can not be partitioned. We assume that, the available power in each EU is sufficient to guarantee the threshold data rate $R_{th_e}$.

Under this restriction, an instance of (P1) can be created in a polynomial-time as following. Let the ES be the knapsack and the EUs' tasks are equivalent to the items, we have $w_i = F_{u,e}$ and $c = F_e$. (P1) is equivalent to maximizing the $-\frac{1}{U}\sum_{u=1}^{U}(T_{u,e}^{\text{cp}} + T_{e,u}^{\text{fb}})x_{u,e}$ subject to (P1b) and (P1g) constraints. As KP is $\mathcal{NP}$-hard, (P1) is $\mathcal{NP}$-hard too. This completes the proof of lemma 1. ■

The formulated problem (P1) is a mixed-integer non-convex $\mathcal{NP}$-hard problem, and it is challenging to solve it using traditional approaches. Hence, an efficient distributed DRL-based solution is proposed in the following.

## IV. Deep Reinforcement Learning Solution

This section models problem (P1) as a Markov decision process (MDP) and then details the proposed DRL solution.

### A. MDP-based computation offloading model

The joint computation offloading and resource allocation process can be represented as a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}_a(t), r_a(t))$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathbb{P}_a(t)$ is the probability of choosing action $a(t) \in \mathcal{A}$ at time slot $t$, and finally $r_a(t)$ is the immediate reward due to action $a(t) \in \mathcal{A}$.

Each EU acts as an independent learning agent to find the optimum offloading policy. At the beginning of each time slot, each EU obtains the state information through local observations and the ESs broadcast their available computing resources to all EUs. The state $s \in \mathcal{S}$ at time slot $t$ is defined by $s(t)$=$\{\underline{x}_u(t), M_u(t), \underline{F}(t), \underline{h}_u(t)\}$ where:

- $\underline{x}_u(t)$ is the binary association vector at time slot $t$.
- $M_u$ is the task size of the $u$th EU.
- $\underline{F}(t)$ is the processing capacity of ESs at time slot $t$.
- $\underline{h}_u(t)$ is the channel gain vector between EU $u$ and all ESs.

When an EU launches an application, it has to decide (i.e., takes an action) to execute the task locally or remotely. Based on the execution time prediction and on the computation resources of ESs, the EU chooses ESs and divides the task into sub-tasks. Then, it chooses the power level to upload its data size to ESs. Here, we suppose $N$ discrete power level values $\underline{P} = [p_e^1, \cdots, p_e^N]$. The reward function helps to find the optimum action policy and value function. After performing the ES selection, the EU calculates the immediate reward. The latter is given by:

$$r_a(t) = -(T_{u,e}^{\text{tx}} + T_{u,e}^{cp} + T_{e,u}^{\text{fb}}). \tag{10}$$

Note that a positive reword is considered if the agent minimizes its execution delay and a negative reward otherwise.


Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 08,2023 at 16:24:53 UTC from IEEE Xplore. Restrictions apply.

The cumulative reward is given by $D = \sum_{t=0}^{T} \xi^t r_a(t)$ where $\xi \in [0, 1]$ is the discount factor. Each agent aims to adjust the offload task strategy by maximizing the long-term accumulated reward through iterations. Since the tackled problem (P1) aims to minimize the system execution delay, the overall reward is equivalent to the negative of the objective function.

### *B. DRL-based computation offloading model*

In the proposed DRL solution, parallel training is investigated by using the asynchronous advantage actor-critic (A3C) approach. A3C is a temporal difference version of the policy gradient composed of two networks: actor and critic. The proposed A3C solution is represented in Algorithm 1. Each agent has its own local actor-critic for training and a global actor-critic to share parameters between other agents. At each time slot $t$ and based on the current state, the agent uses the past experiences from the replay buffer to train its actor-critic network and to perform an action. The action taken in line 8 is the selection of ESs. Then, the sub-task size is determined. If the chosen ES is connected to the EU, in line 10, the power is allocated by the EU and the sub-task is offloaded. However, when the ES is not connected to the EU, the EU does not perform power allocation and the sub-task is executed locally in line 11. Then, the EU calculates $r(t)$ using equation (10) and moves to a new state $s(t+1)$. At each time slot, each EU stores its experience in the local replay buffer, calculates the loss and updates the actor-critic network parameters. Finally, in lines $18-19$, algorithm 1 stores the actor-critic experiences in the global replay buffer and updates the global parameters of each actor-critic network.

**Algorithm 1** A3C algorithm

```
1:  Initialize the actor-critic network with random weights.
2:  Initialize the local replay buffer for each actor-critic.
3:  Initialize the global replay buffer.
4:  for each time slot t do
5:    for each actor-critic do
6:      Set the initial state s_0.
7:      while tasks are not executed do
8:        Perform an action a_s(t) from the actor network.
9:        if the agent selects an ES then
10:         Determinate the corresponding sub-task and allocate the transmit power level.
11:         Execute the task locally.
12:         Calculate r_s(t) and move to s(t+1).
13:         Store the experience into the local replay buffer.
14:         Update the actor-critic network parameters.
15:       end if
16:     end while
17:   end for
18:   Store experiences into the global replay buffer.
19:   Update the weights of each actor-critic network.
20: end for
```

When using A3C approach, the EU improves both the policy and the value function. The actor is the policy-based function that controls the EU's behaviour and takes decisions. It determines the most suitable action given a state. The critic is the value function that criticizes the actions taken by the actor. Knowing the value function helps improve the policy gradient. Each EU learns from the environment and shares its memory, with other EUs to accelerate the learning process. The policy and the value loss functions of state $s \in \mathcal{S}$ at time slot $t$ are denoted respectively by $PL_s(t)$ and $VL_s(t)$, and are calculated as follows:

$$PL_s(t) = -\log_2(\pi_s(t)) \times D_s(t) - \beta \times H_\pi(t), \tag{11}$$

$$VL_s(t) = (r_a(t) - V_s(t))^2, \tag{12}$$

where $H_\pi(t)$ is the entropy function and $D_s(t)$ is the advantage function which is given by:

$$D_s(t) = Q(s(t), a(t)) - V_s(t). \tag{13}$$

$V(s)$ is the output of the critic network and $Q(s(t), a(t))$ is the output of the *softmax* activation function of the actor network. Note that, $D_s(t)$ is used to improve the agent's decision and $H_\pi(t)$ is used in the loss function to guarantee the network exploration.

## V. Simulation Results

In this section, we evaluate the performance of the proposed DRL solution and we compare it with the following approaches. The *local* approach is where all tasks are executed locally in EUs. The *best channel* approach is where all tasks are offloaded to the ES with the best channel gain condition. The *nearest server* approach is where the selected ES is one of the nearest ESs. Finally, the *best server* approach is where the selected ES is one of the ESs with the highest computing resources. We consider Rayleigh fading channels where the coefficients $h_{u,e}$ follow a complex normal distribution with 0 mean and $d_{u,e}^{\alpha}$ variance [18]. $d_{u,e}$ is the distance between the $u$th EU and the $e$th ES, and $\alpha$ is the path-loss exponent [18].

As mentioned before, the action taken at each time slot depends strictly on the number of ESs, the channel gain conditions, and the available computing resources at ESs. To evaluate the impact of each parameter, we start by fixing the offloading server vector $\underline{x}(t)$ and the available computing resources at ESs $\underline{F}(t)$.

The learning rate is set to $0.001$. Then, we train the A3C algorithm while varying the channel gain conditions from one-time slot to another.

Fig. 2 shows the fast convergence of the proposed DRL solution. It is clear that, the average rewards improves as the number of iterations increases and the training converges from $200$ iterations.

Fig. 3 plots the average delay versus different task sizes. The evaluation of the DRL solution is performed under the same number of computation tasks. We set the number of users to 1, the number of ESs to 3, and the available computing resources in ESs to $[5.5, 3.5, 2.5]$ GHz. It can be seen that, as the task size increases, the average delay becomes more important. The gap between the DRL solution and the benchmark ones widens


Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 08,2023 at 16:24:53 UTC from IEEE Xplore. Restrictions apply.

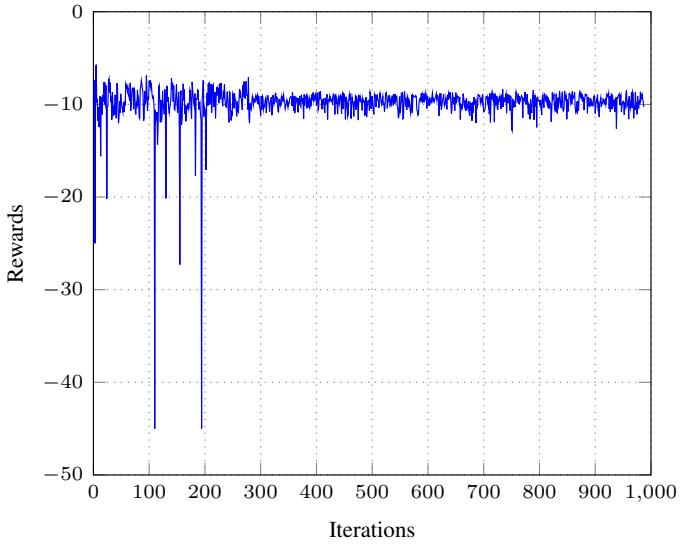


Fig. 2 – Training rewards.

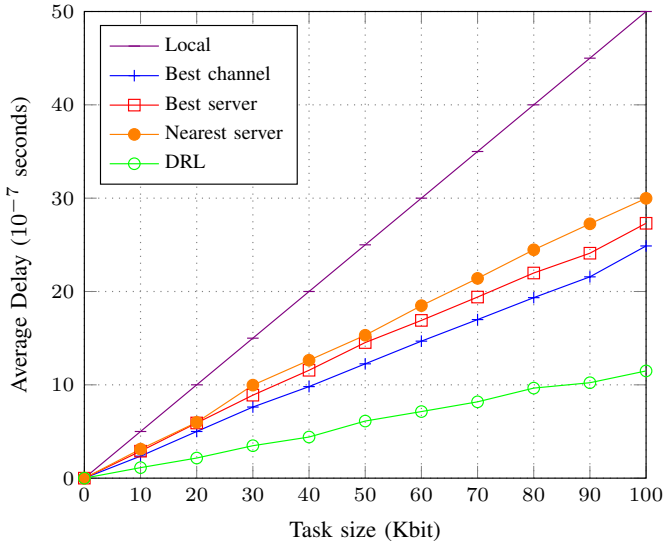


Fig. 3 – Average delay for different task sizes.

further as the task size increases. We can also observe that when the task size is less than 10 Kbits, the average delay of the *best channel*, the *nearest server*, and the *best server* are close. Whereas, the performance gap between the *local* and DRL solutions is more than 70% when the task size is greater than 30 Kbits. Hence, this figure proves the effectiveness of the proposed DRL solution.

Fig. 4 illustrates the impact of the ES number on the average delay. As expected, as the number of available ESs increases, the average delay decreases. We can observe that, the DRL solution outperforms always the other benchmark ones. It can be also seen that, when the EU is connected to 2 ESs, the performance gap between the *nearest server* and the *best channel* solutions is more than 23%. Whereas, the performance gap between the *nearest server* and the DRL solutions is more

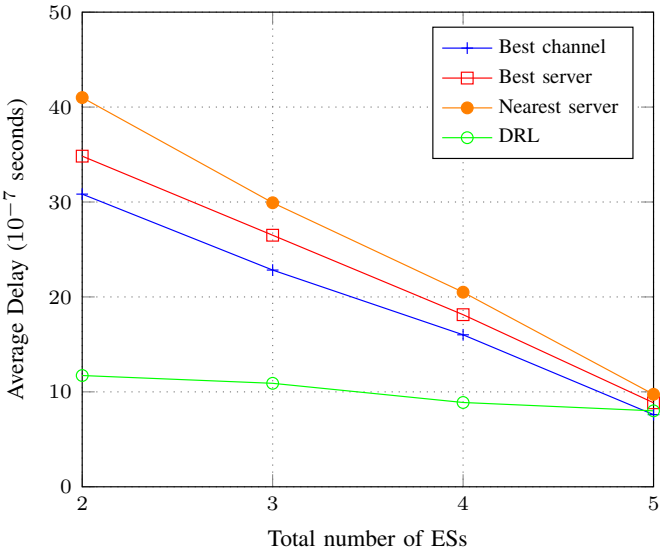


Fig. 4 – Average delay for different number of ESs.

than 70%. Further, when the number of ESs is larger than 4, all the task offloading solutions converge approximately to the same average delay. This is because when the EU is connected to all ESs, more computing resources are available and hence the task is totally offloaded and executed in ESs.

## VI. Conclusion

This paper investigated the utilization of deep reinforcement learning approach to solve the joint computation offloading and resource allocation problem in NOMA-based MEC networks. The formulated optimization problem is a non-linear and non-convex optimization problem under resource constraints where the objective is to minimize the latency of computation for edge wireless users. To provide an efficient practical solution to the formulated problem, an asynchronous advantage actor-critic algorithm is proposed. Simulation results showed that the proposed solution outperforms the basic benchmark solutions.

## References

[1] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Computer Networks*, vol. 182, p. 107496, 2020.

[2] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A Survey on Mobile Augmented Reality With 5G Mobile Edge Computing: Architectures, Applications, and Technical Aspects," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1160–1192, 2021.

[3] D. Van Le and C.-K. Tham, "A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds," in *PROC. IEEE INFOCOM WKSHPS*, 2018, pp. 760–765.

[4] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, 2020.

[5] X. Zhang, Z. Lin, B. Ding, B. Gu, and Y. Han, "Deep Multi-Agent Reinforcement Learning for Resource Allocation in D2D Communication Underlaying Cellular Networks," in *21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2020, pp. 55–60.



Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 08,2023 at 16:24:53 UTC from IEEE Xplore. Restrictions apply.

[6] F. Qi, L. Zhuo, and C. Xin, "Deep Reinforcement Learning Based Task Scheduling in Edge Computing Networks," in *PROC. IEEE ICCC*, 2020, pp. 835–840.
[7] A. Moussaid, W. Jaafar, W. Ajib, and H. Elbiaze, "Deep Reinforcement Learning-based Data Transmission for D2D Communications," in *PROC. of IEEE WiMob*, 2018, pp. 1–7.
[8] X. Diao, J. Zheng, Y. Wu, and Y. Cai, "Joint Computing Resource, Power, and Channel Allocations for D2D-Assisted and NOMA-Based Mobile Edge Computing," *IEEE Access*, vol. 7, pp. 9243–9257, 2019.
[9] L. P. Qian, B. Shi, Y. Wu, B. Sun, and D. H. K. Tsang, "NOMA-Enabled Mobile Edge Computing for Internet of Things via Joint Communication and Computation Resource Allocations," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 718–733, Jan 2020.
[10] Z. Yang, Y. Liu, Y. Chen, and N. Al-Dhahir, "Cache-Aided NOMA Mobile Edge Computing: A Reinforcement Learning Approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6899–6915, october 2020.
[11] J. Jeong, I.-M. Kim, and D. Hong, "Deep Reinforcement Learning-based Task Offloading Decision in the Time Varying Channel," in *PROC. IEEE ICEIC*, March 2021, pp. 1–4.
[12] V. Dat Tuong, T. Phung Truong, A.-T. Tran, A. Masood, D. Shumeye Lakew, C. Lee, Y. Lee, and S. Cho, "Delay-Sensitive Task Offloading for Internet of Things in Nonorthogonal Multiple Access MEC Networks," in *PROC. IEEE ICTC*, 2020, pp. 597–599.
[13] J. Liu and Q. Zhang, "Adaptive Task Partitioning at Local Device or Remote Edge Server for Offloading in MEC," in *PROC. IEEE WCNC*, 2020, pp. 1–6.
[14] L. Jianhui and Z. Qi, "Offloading Schemes in Mobile Edge Computing for Ultra-Reliable Low Latency Communications," *IEEE Access*, vol. 6, pp. 12 825–12 837, 2018.
[15] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, 2016.
[16] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, 2015.
[17] S. Khuri, T. Bäck, and J. Heitkötter, "The zero/one multiple knapsack problem and genetic algorithms," in *ACM symposium on Applied computing*, 1994, pp. 188–193.
[18] W. Jaafar, W. Ajib, and D. Haccoun, "Opportunistic Adaptive Relaying in Cognitive Radio Networks," in *PROC. IEEE ICC*, 2012.


Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 08,2023 at 16:24:53 UTC from IEEE Xplore. Restrictions apply.