Smart Media Journal / Vol.12, No.3 / ISSN:2287-1322
https://dx.doi.org/10.30693/SMJ.2023.12.3.68

Dynamic Computation Offloading Based on Q-Learning for UAV-Based Mobile Edge Computing

Shreya Khisa[1], Sangman Moh[2]

Abstract

Emerging mobile edge computing (MEC) can be used in battery-constrained Internet of things (IoT). The execution latency of IoT applications can be improved by offloading computation-intensive tasks to an MEC server. Recently, the popularity of unmanned aerial vehicles (UAVs) has increased rapidly, and UAV-based MEC systems are receiving considerable attention. In this paper, we propose a dynamic computation offloading paradigm for UAV-based MEC systems, in which a UAV flies over an urban environment and provides edge services to IoT devices on the ground. Since most IoT devices are energy-constrained, we formulate our problem as a Markov decision process considering the energy level of the battery of each IoT device. We also use model-free Q-learning for time-critical tasks to maximize the system utility. According to our performance study, the proposed scheme can achieve desirable convergence properties and make intelligent offloading decisions.

Keywords : Computation offloading | energy efficiency | mobile edge computing | Q-learning | reinforcement learning

I. INTRODUCTION

Internet of things (IoT) devices have limited computation capability and energy resources to support different computation-intensive applications, such as face recognition and virtual/augmented reality games. Mobile edge computing (MEC) techniques can effectively handle these challenges. Using MEC, IoT devices can offload computation-intensive tasks to the MEC server. IoT devices can utilize the MEC server resources of computation, energy, and memory, which can help enhance the energy efficiency of IoT devices. Two types of offloading techniques are being applied: binary and partial offloading[1,2]. In binary offloading, the entire task can be offloaded to the MEC server or computed locally. However, in partial offloading[3,4], some portion of the task can be offloaded to the MEC server, and the rest can be executed locally. Although the partial offloading mechanism can provide more benefits than the binary offloading scheme, it has complex hardware requirements.

Unmanned aerial vehicles (UAVs) are gaining much attention among the researchers and industries due to their flexibility, ease of deployment,

* This study was supported in part by research fund from Chosun University (2022).

[1] Dept. of Information and Systems Engineering, Concordia University, Canada
[2] Dept. of Computer Engineering, Chosun University

Manuscript: 2023.02.10
Confirmation of Publication: 2023.03.14

Corresponding Author: Sangman Moh, e-mail: smmoh@chosun.ac.kr

and flying capability[5-8]. Owing to the mobility of UAVs, in recent years, the integration of UAVs with MEC servers has received considerable attention because it can provide services in different urban and hostile environments. The UAV-MEC server can significantly enhance the computation performance[9,10]. For example, Yang et al. [11] proposed a UAV-based MEC system in which multiple UAVs serve as MEC servers to users on the ground. Recently, the authors in [12] proposed an algorithm for UAV-MEC to ensure the quality of service, as well as the optimization of the UAV trajectory.

In this study, we propose a dynamic computation offloading algorithm based on Q-learning to offload the tasks intelligently to the UAV-MEC server. We utilize a binary offloading mechanism, and each offloading decision is made based on the energy level and the deadline of the task in the current time slot. We consider a single UAV-MEC-based urban scenario. The UAV flies over a location where IoT devices are deployed. IoT devices can perform local execution and offload their tasks to the UAV-MEC server.

The contributions of this article can be summarized as follows: We formulate our problem as a Markov decision process (MDP)-based problem. Then, we develop a dynamic model-free Q-learning-based computation offloading algorithm, which can aid the offloading decision based on the deadline of the tasks and the energy level of the IoT devices to maximize the system utility. The proposed scheme achieves desirable convergence properties and makes intelligent offloading decisions, and performs better in terms of energy consumption and execution cost.

In Section II of this paper, relevant literature is reviewed. The system model is addressed in Section III. Our Q-learning-based offloading scheme is presented in Section IV. The simulation settings and results are discussed in Section V, followed by conclusions in Section VI.

II. RELATED WORKS

Recently, different offloading strategies have been proposed for UAV-based MEC systems. The authors in [13] proposed a UAV-based MEC system that optimized bit allocation and the UAV trajectory. They investigated the optimized bit allocation problem in both uplink and downlink communications. For uplink and downlink transmissions, frequency division duplex and non-orthogonal multiple access schemes were employed. The problem was formulated as a non-convex optimization problem and addressed using successive convex approximation-based approach. A partial offloading scheme in a UAV-MEC system was proposed in [14]. They investigated the minimization of the sum of the maximum delay in each time slot by jointly optimizing the offloading ratio, user−scheduling variables, and the UAV trajectory. In [15], the authors presented a UAV−MEC where UAV functioned as both an MEC server and relay to the access point.

An optimization problem was formulated to minimize the total energy consumption. A multi−UAV−based edge computing scenario was presented in [11]. Multi−UAVs were deployed for load balancing among the UAV and to enhance the performance of the entire system. A deep reinforcement learning−based task scheduling approach was proposed that could enhance the efficiency of task execution in each UAV. In [16], a Lyapunov function−based approach was used to minimize the energy consumption of the UAV−MEC system. Recently, a computation offloading algorithm based on multi−agent reinforcement learning was proposed in [17].

III. SYSTEM MODEL

We consider an urban scenario in which a UAV with an MEC server hovered over multiple IoT devices as shown in Fig. 1. The UAV provides edge services to ground IoT devices to assist them in completing their computation−intensive and time−critical tasks. IoT devices could offload their tasks to the UAV−MEC server to reduce energy consumption and improve the task execution latency. We assume that the UAV will return to its initial location at the end of each period after providing edge services to the ground IoT devices.

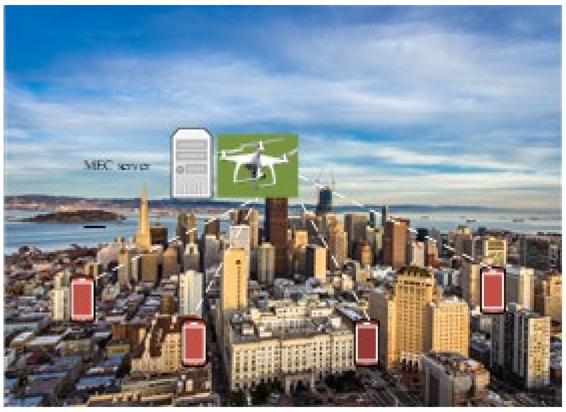

Fig. 1. An example of applications

1. Local Computation Model

When an IoT device decides to compute the task locally rather than offloading it to the UAV−MEC server, it employs a local computation model. If the IoT device decides to process its computation task i locally, the computation time depends on its own computing resources. The time required to execute a task depends on the CPU frequency and processing time. The execution latency of the task can be calculated as

$$t_i^{loc} = \frac{D_i C_i}{f_i^{loc}}, \tag{1}$$

where D_i, C_i, and f_i^{loc} represent the data size of task i, CPU cycles needed to process one bit of task, and computation capacity of the IoT device i, respectively.

As we have considered the latency constraint task, the execution latency must satisfy the following condition:

$$t_i^{loc} \le T_i^{\max} \tag{2}$$

Moreover, the computation capacity for the IoT device i is constrained by

$$f_i^{loc} \le f_i^{\max}, \tag{3}$$

where f_i^{max} is the maximum computational capability of an IoT device. The energy consumption for processing a task can be determined by

$$E_i^{loc} = \varphi_{loc} C_i D_i \left(f_i^{loc} \right)^2, \tag{4}$$

where φ_{loc} represents the effective switched capacitance of the processor of the IoT device.

2. UAV-MEC Computation Model

The UAV−MEC computation model is used when the IoT device decides to offload its task to the UAV−MEC server. When the size of the computation result is negligible compared to that of the computation task, the time required to receive the computation result can be omitted. For example, in applications such as face recognition or speech recognition, the task size is considerably larger than the size of the task result. In this case, the execution time depends on the computation execution time and uplink transmission time. The total execution time of the IoT device i is

$$t_{mec} = \frac{C_i}{f_{mec}} + \frac{D_i}{\gamma_i}, \tag{5}$$

where f_{mec} represents the computation capability of the UAV−MEC, and γ_i is the data transmission rate.

Eq. (5) must fulfill the task latency constraint requirement condition as

$$t_i^{mec} \le T_i^{\max}. \tag{6}$$

The energy consumption in the UAV for the execution of a single task can be represented as

$$E_{mec} = \varphi_{mec} C_i D_i \left(f_{mec} \right)^2, \tag{7}$$

where φ_{mec} represents the effective switched capacitance of the processor of the UAV.

Ⅳ. Q−LEARNING−BASED COMPUTTAION OFFLOADING

The offloading decision depends on the system state, which considers the energy of the IoT device and the deadline of the task. Notably, the current state depends only on the immediate previous state rather than the past states. Hence, we have formulated the computation offloading decision as an MDP.

We propose a model−free Q−learning−based scheme to make offloading decisions. In each time slot, each IoT device first observes the energy level E_k of their battery and then the deadline of each task $T_1^{(0)}, \dots \dots, T_M^{(0)}$. The state by considering the energy level and deadline can be formulated as $s_k = [T_1^{(0)}, \dots \dots, T_M^{(0)}, E_k]$. Based on the state s_k, the IoT devices decide to offload as $a_i = o_i$. These devices apply the ε −greedy policy with $0 < \varepsilon \le 1$ to avoid convergence in the local maxima.

The IoT devices either offload the entire task to the UAV−MEC server or compute it locally based on the state s_k. After processing the offloaded task to the UAV−MEC server, the result is sent back to the IoT device. As the size of the result after computation is very small compared to that of the offloaded task, the latency between the IoT device and UAV−MEC is negligible. The IoT device calculates the energy level, energy consumption, and computation latency. The tasks are latency−constrained. If the computation latency crosses the deadline bound, the task is automatically dropped. Moreover, the IoT devices are energy−constrained, and after the threshold level, they are unable to offload their tasks. Hence, the task proceeds for local computation. Moreover, after a certain threshold energy level, IoT devices are unable to compute the task locally. Hence, the task is dropped.

The utility of the IoT device at time slot k can be denoted by $U_i(k)$ depending on the energy consumption and execution cost, which can be written as:

$$\sum_{\max} U_i(k) = \delta t_k + \omega E_k \qquad (8)$$

where δ and ω represent the weights of the execution cost and energy consumption, respectively. To maximize the system utility $U_i(k)$, the Q−value is updated as:

$$Q(s_k,a_k) \leftarrow (1-\alpha)Q(s_k,a_k) + \alpha(U_k + \gamma \max Q(s_{k+1},a')) \qquad (9)$$

where the learning rate $\alpha \in [0,1]$ is the learning rate and $\gamma \in [0,1]$ denotes the discount factor, which is important for the future reward. The Q−learning−based computation offloading algorithm is presented as Algorithm 1. This algorithm can obtain an optimal policy for computation offloading using exploration and exploitation and has sufficient time slots.

Algorithm 1: Q-learning-based computation offloading

```
Input: Initialized $C, f, D, \alpha, \gamma, \varepsilon, T_1^{(0)}, \dots \dots, T_M^{(0)}$.
Output: Offloading decision and data offloading
1: for $k = 1,2,3, \dots$ do
2:     Check the energy level $E_k$.
3:     Observe the deadline $T_1^{(0)}, \dots \dots, T_M^{(0)}$.
4:     $s_k = [T_1^{(0)}, \dots \dots, T_M^{(0)}, E_k]$.
5:     Select $a_k = [o_i] \in A$ via $\varepsilon$-greedy policy.
6:     Offload data to the UAV-MEC.
7:     Calculate the energy level.
8:     Calculate the energy consumption, computation latency
9:     Calculate utility $U_k$ using Eq. (8).
10:    Update Q-value using Eq. (9).
11: end for
```

V. PERFORMANCE EVALUATION

1. Simulation Environment

In our simulation, we consider an urban $500\ m \times 500\ m$ area with $M = 10$ fixed IoT devices in this area. UAV height is 50 m. In each device, the task request is generated using a binomial distribution with a probability of 0.6. The simulation parameters are presented in Table 1.

Table 1. Simulation parameters

Parameter	Description	Value
M	Number of IoT devices	10
f_i^{loc}	Computation capability of IoT device	2 GHz
f_{mec}	Computation capability of UAV-MEC	8 GHz
C	Number of CPU cycles needed to process one bit of task	500
h	Height of UAV	50 m
a	Parameter of channel model for dense urban scenario	12.08
b	Parameter of channel model for dense urban scenario	0.11
α	Learning rate	0.9
γ	Discount factor	0.5
ε	Exploration probability of ε-greedy	0.1

k	Switch capacitance	10^{-28}
B	Bandwidth	1 MHz
D_i	Task input size	1.5–15 Mbits

We set the computation capability of the IoT device and UAV to 2 GHz and 8 GHz, respectively. We consider $B = 1$ MHz as the system bandwidth, $\sigma^2 = -169$ as the additive Gaussian noise, and $T = 50000$ as the number of time slots. As we consider a dense urban scenario, we set the line−of−sight $\eta_{LoS} = 1.6$, non−line−of−sight $\eta_{NLoS} = 23$, $a = 12.08$, and $b = 0.11$ [18]. We utilize a learning rate $\alpha = 0.9$, discount factor $\gamma = 0.5$, and exploration probability $\varepsilon = 0.1$ for our Q−learning−based algorithm. Each input task size varies between 1.5 Mb and 15 Mb. The number of CPU cycles required to process one bit of task is set to 500 for both the IoT device and the UAV−MEC server. The weights of the execution cost and energy consumption cost are set at $\delta = \omega = 0.5$. The initial battery energy level is set at 100 W.

2. Simulation Results and Discussion

Figs. 2 and 3 show the average execution cost and average energy consumption of our proposed scheme, respectively. The average execution cost represents the average execution time in seconds. We calculated the average execution cost by dividing the local and UAV−MEC server execution costs by the total number of task requests. The average energy consumption was also calculated in the same manner as the average execution cost.

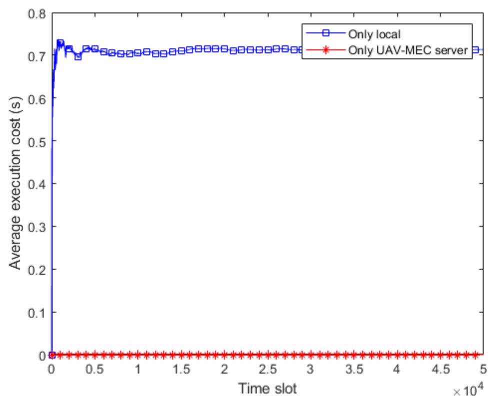

Fig. 2. Average execution cost

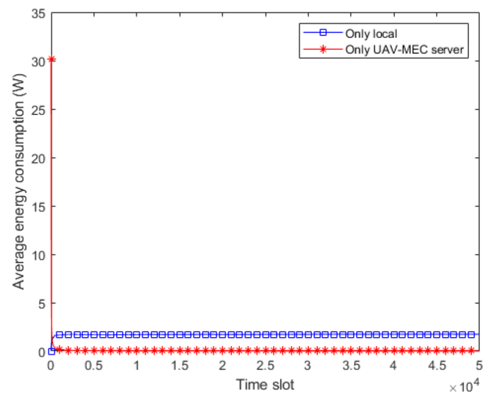

Fig. 3. Average energy consumption

It can be observed that local execution requires a longer execution time than UAV−MEC server execution. Moreover, Figs. 2 and 3 clearly show that, after the convergence of the algorithm, the average execution cost and average energy consumption remain the same in each time slot. After convergence, the average execution cost of the UAV−MEC server in each time slot is approximately 0.2 s, and for local execution, it is approximately 0.7 s. After convergence, the average energy consumption for both the local and UAV−MEC servers is very low.

Fig. 4 shows the average ratio for choosing whether to offload data or execute it locally. Our Q−learning−based algorithm makes an intelligent decision to maximize the system utility. Fig. 4 shows that, for approximately 70% of the time, the IoT device chooses to offload data to the UAV−MEC server; otherwise, it executes the task locally. The decision is based on the deadline of the task and the current energy level of the IoT device.

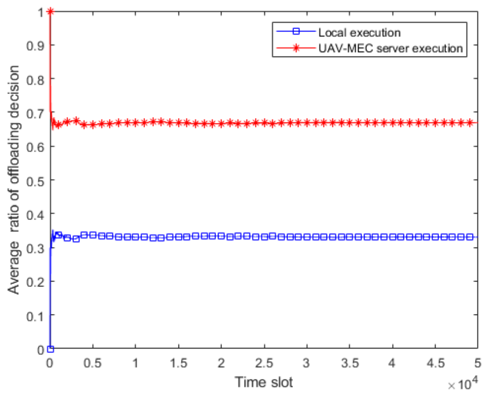

Fig. 4. Average ratio of offloading decision

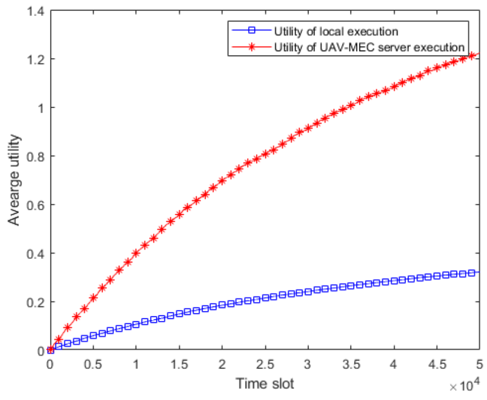

Fig. 5. Average utility

Fig. 5 is closely related to Fig. 4. As the offloading decision is made to maximize the system utility, Fig. 4 shows that the system utility for UAV−MEC server execution is higher than that for local execution. We calculated the system utility based on the execution cost and energy consumption. For simplicity, we considered the same weight for both the execution cost and energy consumption, i.e., $\delta = \omega = 0.5$. The UAV−MEC server execution cost depends on the data rate and offloading time, whereas the execution cost in IoT depends only on the device itself.

We adopted the ε−greedy policy to balance exploration and exploitation. The reason behind using this policy is that the algorithm is not stuck at the local optimum; rather, it converges to the global optimum. In an ε−greedy based algorithm, the value of ε plays a very important role in the exploration and exploitation techniques. Hence, we evaluated our algorithm by varying the ε value. The size of the input task varied between 10,000,000 and 50,000,000 bits. Fig. 6 shows the average execution cost for different values of ε. The figure that the average execution cost is highest when $\varepsilon = 0.9$ and lowest when $\varepsilon = 0.1$.

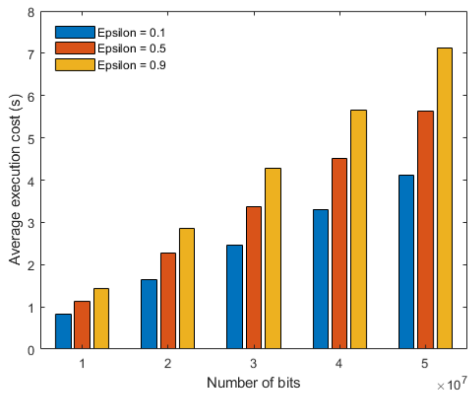

Fig. 6. Execution cost with different values of ε

Fig. 7 shows the average energy consumption for different values of ε. The energy consumption is the lowest with $\varepsilon = 0.9$ and highest when $\varepsilon = 0.1$. Finally, Fig. 8 shows the average utility of the system with various values of ε. The utility is maximum when $\varepsilon = 0.1$ and minimum when $\varepsilon = 0.9$.

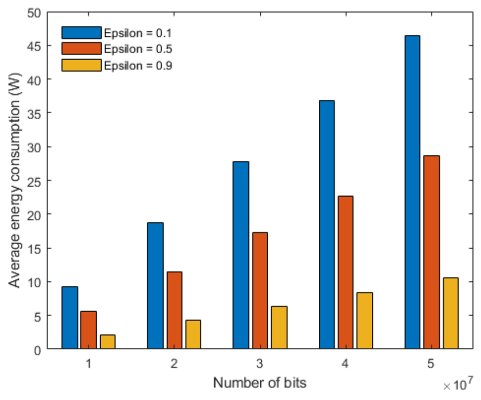

Fig. 7. Energy consumption with different values of ε

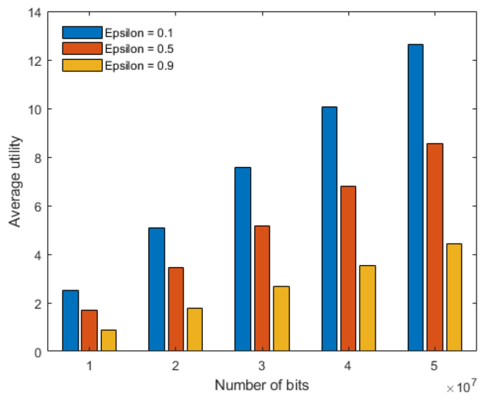

Fig. 8. Utility with different values of ε

VI. CONLCUSION

In this study, we proposed an energy-aware Q-learning-based computation offloading scheme for UAV-based MEC systems. To maximize the system utility, our offloading algorithm takes decisions intelligently based on the system state (the deadline of the tasks and the energy level of IoT devices). The decision involves choosing between offloading the data to UAV-MEC and executing it locally. Our proposed scheme can provide better performance in terms of energy consumption and execution costs. In the future, we intend to investigate deep Q-learning-based offloading techniques focusing on the minimized energy consumption.

REFERENCES

[1] S. M. A. Huda and S. Moh, "Survey on computation offloading in UAV-enabled mobile edge computing," *Journal of Network and Computer Applications,* vol. 201, article no. 103341, pp. 1-26, 2022

[2] S. Poudel and S. Moh, "Task assignment algorithms for unmanned aerial vehicle networks: A comprehensive survey," *Vehicular Communications,* vol. 35, article no. 100469, pp. 1-29, 2022

[3] S. Mao, S. Leng, K. Yang, X. Huang and Q. Zhao, "Fair energy-efficient scheduling in wireless powered full-duplex mobile-edge computing systems," *Proc. of 2017 IEEE Global Communications Conf.*, pp. 1-6, 2017

[4] C. You, K. Huang, H. Chae and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. on Wireless Communications,* vol. 16, no. 3, pp. 1397-1411, 2016

[5] Y. Zeng, R. Zhang and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Communications Magazine,* vol. 54, no. 5, pp. 36-42, 2016

[6] S. W. Kim, "Prototype design for unmanned aerial vehicle-based big data processing," *Smart Media Journal,* vol. 5, no. 2, pp. 51-58, 2016

[7] N. H. Kim, "Development of atmospheric environment information collection system using drone," *Smart Media Journal*, vol. 7, no. 4, pp. 44-51, 2018

[8] Y. Zhang, B. Kim, J. Sun and J. Lee, "Searching the damaged pine trees from wilt disease based on deep

learning," *Smart Media Journal*, vol. 9, no. 3, pp. 46-51, 2020

[9] S. Garg, A. Singh, S. Batra, N. Kumar and L. T. Yang, "UAV-empowered edge computing environment for cyber-threat detection in smart vehicles," *IEEE Network,* vol. 32, no. 3, pp. 42-51, 2018

[10] F. Zhou, Y. Wu, R. Q. Hu and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications,* vol. 36, no. 9, pp. 1927-1941, 2018

[11] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane and Y. Liu, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet of Things Journal,* vol. 7, no. 8, pp. 6898-6908, 2020

[12] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding and F. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Trans. on Vehicular Technology,* vol. 69, no. 5, pp. 5723-5728, 2020

[13] S. Jeong, O. Simeone and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. on Vehicular Technology,* vol. 67, no. 3, pp. 2049-2063, 2017

[14] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet of Things Journal,* vol. 6, no. 2, pp. 1879-1892, 2018

[15] T. Zhang, Y. Xu, J. Loo, D. Yang and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Trans. on Industrial Informatics,* vol. 16, no. 8, pp. 5505-5516, 2019

[16] J. Zhang, *et al.*, "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet of Things Journal,* vol. 6, no. 2, pp. 3688-3699, 2018

[17] S. Zhu, L. Gui, D. Zhao, N. Cheng, Q. Zhang and X. Lang, "Learning-based computation offloading approaches in UAVs-assisted edge computing," *IEEE Trans. on Vehicular Technology,* vol. 70, no. 1, pp. 928-944, 2021

[18] A. Al-Hourani, S. Kandeepan and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Communications Letters,* vol. 3, no. 6, pp. 569-572, 2014

Authors

Shreya Khisa received the B.S. degree in computer science and engineering from University of Chittagong, Bangladesh in 2017 and the M.S. degree in computer engineering from Chosun University, Korea in 2021, respectively. She is currently pursuing the Ph.D. degree with Concordia University, Canada. Her research interests include wireless communication, multiple access techniques, and optimization.

Sangman Moh received his Ph.D. degree in computer engineering from the Korea Advanced Institute of Science and Technology (KAIST), South Korea in 2002. Since late 2002, he has been a professor at the Dept. of Computer Engineering at Chosun University, Korea. Until 2002, he was with the Electronics and Telecommunications Research Institute (ETRI), Korea, where he served as a project leader since he received his M.S. degree in computer science from Yonsei University, Korea, in 1991. His research interests include mobile computing and networking, ad hoc and sensor networks, cognitive radio networks, unmanned aerial vehicle networks, and mobile edge computing.