TASK OFFLOADING IN A SECURE MOBILE EDGE COMPUTING ENVIRONMENT FOR INDUSTRIAL APPLICATIONS

Branka Mikavica [a,*], Aleksandra Kostić-Ljubisavljević [a]

[a] University of Belgrade - Faculty of Transport and Traffic Engineering, Serbia

Abstract: *The emerging intelligent applications in an industrial environment are highly computation-consuming. Smart devices without sufficient computing resources cannot support these applications. Task offloading to the Mobile Edge Computing (MEC) servers can be an appropriate solution to improve the performances in terms of delay sensitivity and security. To maximize the utility of the system, resource allocation should be considered simultaneously. In this paper, we propose a novel Deadline-Aware Task Offloading (DATO) scheme based on the double Vickrey-Clarke-Grove (VCG) auction in a secure MEC environment for industrial applications. The effectiveness of the proposed scheme is compared to the classic Greedy Task Offloading (GTO) scheme. The extensive simulations show that the proposed task offloading scheme is equally efficient in terms of utility, but concurrently improves performances in terms of time-out failures. Furthermore, the applied auction mechanism in the observed task offloading problem assures individual rationality and truthfulness.*

Keywords*: industrial applications, mobile edge computing, task offloading, auction, security.*

1. INTRODUCTION

Industrial applications require the use of smart objects capable of monitoring, collecting, analyzing and making intelligent decisions in an industrial setting. With the emergence of Industry 4.0, manufacturing industry scenarios are focused on the digitization and integration of all physical processes across the entire organization, according to Sengupta et al. (2021). Thus, smart factories and logistic centers can effectively handle the growing complexities. Innovations should make industrial ecosystems more robust and scalable. However, it is necessary to extend the features of cloud services to the end devices and enable local processing of tasks in a timely manner. Smart industrial devices have limited computing capabilities. Moreover, processing on the cloud causes a large response time, disruptions in the underlying communication networks and security issues. One of the effective solutions is to introduce a middleware computing resource, such as Mobile Edge

* b.mikavica@sf.bg.ac.rs

Computing (MEC) to support local task processing instead of processing in the devices or remote cloud server. MEC servers are usually owned by different authorities and profit-driven, according to Luo et al. (2019). Therefore, the offloading cost is imposed on smart devices for task execution on MEC servers.

Auctions are often used as promising solutions for fair distribution in various edge-computing domains, including MEC, Fog Computing and Cloud Computing. Double auctions can efficiently balance the incentives of all relevant participants, including buyers and sellers. Buyers place their bids (their willingness to pay for used resources), while sellers place asks (the minimum required price per resource provided). Specifically, the double Vickrey-Clarke-Grove (VCG) auction is a convenient tool to provide incentives for truthful bidding in the task offloading process in an industrial MEC environment.

In this paper, we observe a general industrial scenario, with several MEC servers in a smart factory or logistics center. Smart devices, including mobile robots, Automated-Guided Vehicles (AGV), smart cameras etc., generate multiple computation-sensitive tasks with heterogeneous constraints on their delay tolerance. These tasks should be offloaded to the MEC servers to improve the system's performance. However, task offloading introduces several issues, including security, heterogeneity of demands and incentives for resource provision. The main contributions of this paper are the following: (i) we propose a novel deadline-aware task offloading scheme to reduce the number of time-out failures in task execution on MEC servers; (ii) we set truthful VCG-based double auction that satisfies individual rationality and provides incentives for task offloading; (iii) simulation results show the effectiveness of the proposed model and improved performances compared to the classic greedy algorithm.

The remainder of the paper is organized as follows. The related work is discussed in Section 2. The proposed system model and problem formulation for the two various double VCG auction-based task offloading schemes are presented in Section 3. In Section 4, we present performance evaluation and discuss simulation results. Finally, Section 5 provides concluding remarks and future research directions.

2. RELATED WORK

Task offloading provides low latency, high bandwidth and stability for the mobile computing scenarios, according to Wang et al. (2018), Cao et al. (2019) and Feng et al. (2019). A price-based distributed method is used by Liu and Liu (2018), to schedule offloaded computation tasks. The Stackelberg game is used to obtain equilibrium and maximize the utilities of both MEC servers and users. The communication cost for task offloading along with the optimization of computing resources is addressed by Al-Shuwaili and Simeone (2017). A multi-queue model is proposed by Zhang et al. (2019), to observe the effects of offloading schemes on the performance of IoT devices combined with their assigned edge computing server. Industrial applications usually pose strict requirements on task offloading, especially in terms of security and latency. However, security requirements are often neglected, according to Zhang et al. (2018), Zhou et al. (2019), and Tran and Pompili (2019), while deadline considerations are often decoupled with resource sharing for MEC servers, according to Zhang (2017) and Alameddine et al. (2019).

Auctions can be effectively used to provide incentives for task offloading in the MEC environment. An adaptive task offloading auction model is proposed by Luo et al. (2019), where the offloading problem is addressed under the access capabilities, latency and security constraints. To allocate edge servers to mobile devices, an auction executed by a pair of deep neural networks is proposed by Mashhadi et al (2020). The results show that the auction mechanism maximizes the profit of the edge servers and satisfies the task processing delay and energy consumption constraints of the mobile devices. Due to the use of deep neural networks, mobile devices are unable to unfairly affect the results of the auctions. A multi-part collaborative task offloading with multiple servers in MEC systems is analyzed by Zhang et al. (2022), to optimize server overload and long-term performance. In this paper, resource allocation on MEC servers is jointly observed with deadline and security considerations in task offloading problem. The truthful double VCG-based auction mechanism is established between the participants, thus providing incentives for MEC servers to enable task offloading.

3. SYSTEM MODEL AND PROBLEM FORMULATION

The proposed architecture for task offloading in the MEC environment for industrial applications is shown in Figure 1. The framework supports a system with heterogeneous MEC servers and IoT devices (mobile robots, AGVs, smart cameras, etc.) that offload computation tasks to MEC servers.

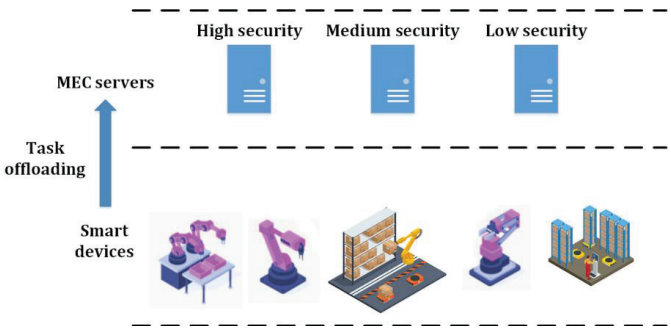

Figure 1. The architecture of the secure industrial MEC environment

The proposed framework can be used in realistic scenarios, smart factories, logistics centers etc., where, for instance, smart (IoT) devices equipped within the assemble lines in a factory offload computation tasks to its neighboring MEC servers. The user, i.e., the IoT device chooses the appropriate MEC server to offload tasks based on the required security level. The selected MEC server decides how to schedule the offloading tasks with various deadline constraints. The determination of the tasks to be provisioned by the MEC server is performed using VCG-based double auction. The MEC server acts as the seller, sharing its computation resources. Each IoT device acts as the buyer that offers the monetary payment to offloaded MEC server.

The proposed architecture comprises three MEC servers, $j \in \{1,2,3\}$, and each server provides a low, medium or high-security level, respectively. At the beginning of each time slot, an active IoT device generates a computation task to be offloaded to the corresponding MEC server. The number of active IoT devices in the time slot $i \in [1, N]$

generating the request for task offloading to the selected MEC server j is denoted by $E_{i,j}$. The arrival rate of tasks to be provisioned follows a Poisson distribution with the average input rate λ. All requests generated by active IoT devices are computationally independent. To initiate task offloading, an IoT device submits a bid to the MEC server. We assume that each active IoT device in a given time slot generates a single request for task offloading, and each task can be executed within a single time slot. The task can be expressed as the following tuple:

$$e_{i,j} = \left(\gamma_{e_{i,j}}, \tau_{e_{i,j}}, b_{e_{i,j}} \right) \tag{1}$$

The parameter $\gamma_{e_{i,j}}$ in (1) describes the computation resources, expressed in capacity units, that task $e_{i,j}$ occupies on the selected MEC server. $\tau_{e_{i,j}}$ denotes the delay tolerance, expressed in time slots, while $b_{e_{i,j}}$ denotes the value of the bid (in monetary units) per required capacity unit of the MEC server j. The bid is a nonnegative value and represents the true willingness to pay for task execution. It is important to emphasize that an IoT device is not being charged by the value of the bid, but by the value that is less or equal to the bid. The payoff is defined in an auction process. It is assumed that bids are independent, and there is no information on other bid values in the auction.

At the beginning of each time slot, the values of asks per capacity unit for each MEC server, denoted by $\alpha_{i,j} \in \left(\alpha_{i,j}^{\min}, \alpha_{i,j}^{\max} \right)$, are defined. The minimum ask, $\alpha_{i,j}^{\min}$, represents the cost per capacity unit for task provisioning. To prevent predatory bidding, the maximum ask per capacity unit is limited to the $\alpha_{i,j}^{\max}$. Once bids of all active bidders are placed and corresponding asks of MEC servers are defined, the determination of candidates for winning bidders is performed. The set of winning bidders is determined depending on the chosen task offloading scheme. We assume that the queue length, denoted by L, represents the number of tasks waiting for the execution.

3.1 Greedy task offloading scheme (GTO)

Set of all requests for task offloading to the chosen MEC server j consists of requests generated in the current time slot and the tasks from the queue determined in the previous time slot. Candidates for task execution are all requests with the bid value that is greater than or equal to the value of the ask for the given MEC server, i.e., $b_{e_{i,j}} \geq \alpha_{i,j}$. Afterwards, the candidate tasks are sorted in the non-increasing order by the values of the bids. Tasks from this sorted set are added to the winning set of the tasks until there is enough capacity for the given MEC server. Thus, for each winning task in the greedy task offloading scheme $\omega_{i,j}^{GTO} \in W_{i,j}^{GTO}$, applies:

$$\omega_{i,j,m}^{GTO} \geq \omega_{i,j,m+1}^{GTO}, \quad \sum \gamma_{\omega_{i,j,m}^{GTO}} \leq \Gamma_j \tag{2}$$

In (2), m denotes the position of the winning task in the set $W_{i,j}^{GTO}$, while Γ_j denotes the capacity of the MEC server j expressed in capacity units. All other tasks, apart from the winning tasks, are candidates for the queue. These tasks are firstly sorted in the non-increasing order by the value of their bids per capacity unit. Afterwards, the tasks from this ordered set are placed in the queue, until the queue length is exceeded. The queue of tasks waiting for task provisioning $q_{i,j,l}^{GTO} \in Q_{i,j}^{GTO}, l \in [0, L]$ can be expressed as follows:

$$q_{i,j,0}^{GTO} \geq \cdots \geq q_{i,j,l}^{GTO} \geq \cdots q_{i,j,L}^{GTO} \tag{3}$$

For each task in the queue $Q_{i,j}^{GTO}$, the parameter depicting delay sensitivity is decremented by one. Furthermore, all tasks whose delay sensitivity is decremented to zero ($\tau_{q_{i,j}^{GTO}} = 0$) are removed from the queue, i.e., time-out failure occurs. The number of candidate tasks removed from the queue due to the time-out failure is denoted by $f_{i,j}^{GTO}$. Thus, the total number of time-out failures for the MEC server j can be expressed as follows:

$$F_j^{GTO} = \sum_i f_{i,j}^{GTO} \tag{4}$$

The price to be paid per offloaded task is determined using the VCG-based double auction. Each winning task is charged depending on the bid value of the next winning task in the ordered set of winning tasks and the resources occupied on the corresponding MEC server. Hence, the price for each offloaded task $\omega_{i,j}^{GTO} \in W_{i,j}^{GTO}$ can be expressed as:

$$p_{\omega_{i,j,m}^{GTO}} = \begin{cases} b_{\omega_{i,j,m+1}^{GTO}} \cdot \gamma_{\omega_{i,j,m}^{GTO}}, & if\ \exists \omega_{i,j,m+1}^{GTO} \\ b_{\omega_{i,j,m}^{GTO}} \cdot \gamma_{\omega_{i,j,m}^{GTO}}, & otherwise \end{cases} \tag{5}$$

As shown in (5), the price for task offloading is always less than or equal to the value of the bid for the given task.

The utility function for each MEC server j represents the difference between the offloading price and the ask determined in the current time slot. Thus, the utility can be expressed as follows:

$$U_j^{GTO} = \sum_i \sum_m p_{\omega_{i,j,m}^{GTO}} - \alpha_{i,j} \cdot \gamma_{\omega_{i,j,m}^{GTO}} \tag{6}$$

3.2 Deadline-aware task offloading scheme (DATO)

Deadline-aware task offloading scheme is proposed to improve user experience in the task offloading process and to reduce the number of time-out failures. Similar to the GTO scheme, the set of the candidate tasks for task offloading are all tasks generated in the current time slot and all tasks waiting in the queue from the previous time slot whose bid

values per capacity unit of the selected MEC server is greater than or equal to the value of the ask per capacity unit for the given MEC server. Afterwards, the candidate tasks are classified depending on their delay sensitivity, and the values of the bids. Thus, the highest priority in task offloading is assigned to the tasks with the most critical value for the delay tolerance. The candidate tasks with the same priority in terms of delay tolerance are further sorted in the non-increasing order of their bids. Hence, the DATO scheme concurrently addresses both generated utility and deadline awareness.

The candidate tasks from the sorted set are assigned to the winning set until the capacity of the corresponding MEC server is not exceeded. Thus, for each winning task $\omega_{i,j}^{DATO} \in W_{i,j}^{DATO}$ applies:

$$\omega_{i,j,m}^{DATO} \geq \omega_{i,j,m+1}^{DATO}, \quad \tau_{\omega_{i,j,m}^{DATO}} \leq \tau_{\omega_{i,j,m+1}^{DATO}}, \quad \sum \gamma_{\omega_{i,j,m}^{DATO}} \leq \Gamma_j \tag{7}$$

Similar to the GTO scheme, the parameter m in (7) denotes the position of the winning task in the set $W_{i,j}^{DATO}$, while Γ_j denotes the capacity of the MEC server j expressed in capacity units. The candidate tasks for queue placement (all tasks apart from the winning tasks in the current time slot) fill in the queue until the queue length is exceeded, i.e.,

$$q_{i,j,0}^{DATO} \geq \cdots \geq q_{i,j,l}^{DATO} \geq \cdots q_{i,j,L}^{DATO} \tag{8}$$

The delay tolerance parameter for each task in the queue is decremented by one, while all tasks whose delay tolerance has expired ($\tau_{q_{i,j}^{DATO}} = 0$) are removed from the queue, thus increasing the number of time-out failures, denoted by $f_{i,j}^{DATO}$. Therefore, the total number of time-out failures for the MEC server j can be expressed as follows:

$$F_j^{DATO} = \sum_i f_{i,j}^{DATO} \tag{9}$$

The price to be paid per offloaded task is set using a VCG-based double auction mechanism, i.e., each winning task is charged depending on the bid value of the next highest winning task and the occupied resources on the selected MEC server. Thus, the price can be expressed as follows:

$$p_{\omega_{i,j,m}^{DATO}} = \begin{cases} b_{\omega_{i,j,m+1}^{DATO}} \cdot \gamma_{\omega_{i,j,m}^{DTO}}, & if\ \exists \omega_{i,j,m+1}^{DATO} \\ b_{\omega_{i,j,m}^{DATO}} \cdot \gamma_{\omega_{i,j,m}^{DATO}}, & otherwise \end{cases} \tag{10}$$

The utility function for each MEC server j can be represented as the difference between the offloading price and the MEC server's ask in the current time slot. Hence, the utility can be expressed as follows:

$$U_j^{DATO} = \sum_i \sum_m p_{\omega_{i,j,m}^{DATO}} - \alpha_{i,j} \cdot \gamma_{\omega_{i,j,m}^{DATO}} \tag{11}$$

4. PERFORMANCE EVALUATION

To analyze the performances of the proposed auction-based task offloading scheme, we performed a set of simulation experiments in the open-source programming language Python 3.7 in 1000 iterations. The proposed DATO scheme is compared with the GTO scheme in the VCG double auction-based process in terms of the utility of MEC servers and the number of time-out failures.

4.1 Simulation setup

In this evaluation study, we consider a network composed of three MEC servers ($j \in \{1,2,3\}$); each MEC sever has 100 capacity units available at the beginning of each time slot. We assume that MEC server $j=1$ provides a low-security level, MEC server $j=2$ provides a medium security level, and finally, the MEC server $j=3$ provides a high-security level. The assumed computation capacity for each MEC server is 100 capacity units. The analysis is performed in $N=27$ time slots of one-hour duration for 30 days. The first three-time slots represent a warm-up period and these slots are not included in the simulation results. The average number of IoT devices in the observed industrial environment takes values from the set (100, 200, 300, 400, 500), while the number of the active IoT devices that generate requests for task offloading is modeled by the Poisson distribution with the average input rate $\lambda=2$. Selection of the appropriate MEC server, i.e., the security level for task offloading is performed randomly. The queue length for each MEC server takes values from the set (50, 100, 150, 200, 250).

4.2 Number of the time-out failures

We observe the number of time-out failures depending on the selected task offloading scheme, the number of IoT devices and the length of the task queue, as shown in Figure 2.

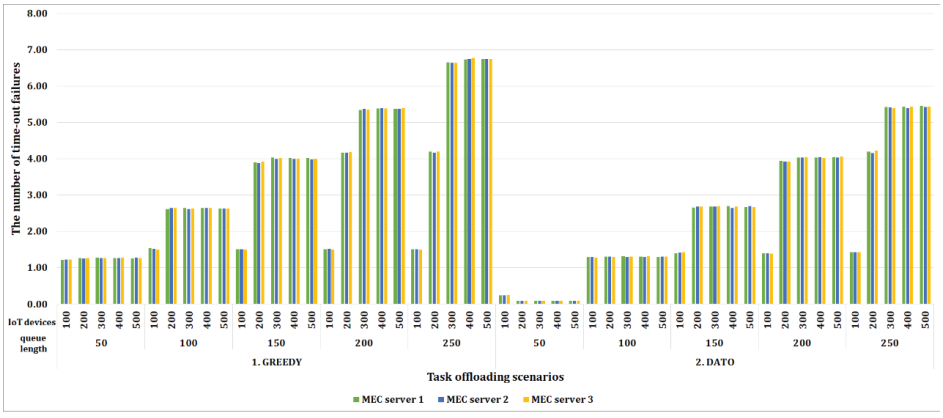

Figure 2. The number of time-out failures

These results apply to each MEC server. It should be emphasized that the proposed DATO scheme provides a lower rate of time-out failures in all scenarios. When the number of active IoT devices in an industrial environment increases, the number of time-out failures also increases, regardless of the length of the task queue. As the length of the queue

increases, the number of failures increases, as well. The greatest rise in the number of failures occurs when the number of active IoT devices increases from 100 to 300 on average, for the largest observed queue length and both task offloading schemes. Afterwards, the number of time-out failures follows the saturation trend and remains on the same level. For shorter queue lengths, the saturation level is achieved even for the lowest number of active IoT devices.

4.3 The utility of MEC servers

The utility of MEC servers (expressed in monetary units) mainly depends on the selected security level. A higher security level imposes higher asks per server's capacity unit, thus generating greater overall utility. In both analyzed task offloading schemes, the queue for task offloading is formed in a non-increasing order by the value of the bids. The GTO scheme is purely greedy and assures the greatest utility regardless of the tasks' delay sensitivity. The proposed DATO scheme considers delay tolerance of each task, thus improving users' experience in the task execution process. It should be emphasized that the DATO scheme maintains nearly the same overall utility as the GTO scheme for each security level, as shown in Figure 3.

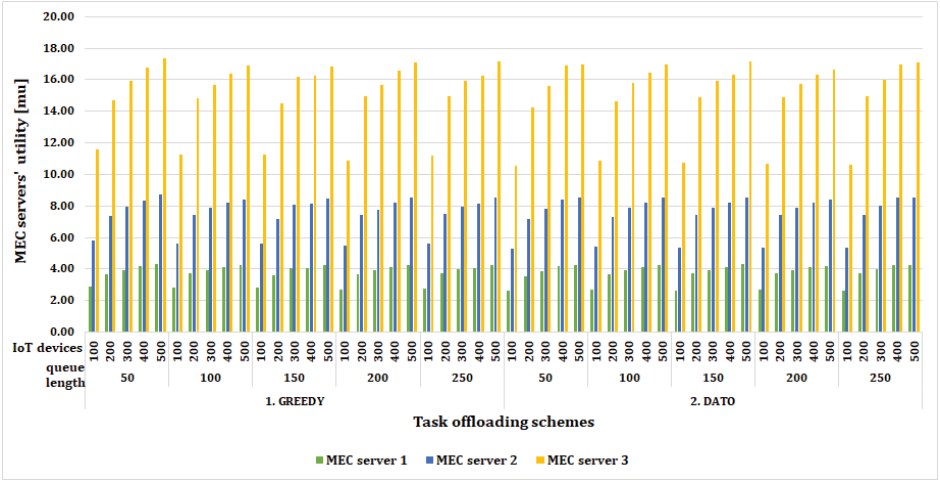

Figure 3. The MEC servers' utility

The utility increases as the number of active IoT devices increases. The queue length has a minor impact on the overall utility. It slightly reduces the utility for the greatest queue length and both task offloading schemes.

5. CONCLUSION

In this paper, the problem of task offloading in the industrial MEC environment is addressed. The novel auction-based task offloading scheme is proposed to meet the requirements in terms of utility and deadline constraints. The truthful double VCG-based auction mechanism provides incentives for MEC servers to perform task offloading and determines payment for each allocated task with deadline constraints. Extensive simulations are conducted to evaluate the performance of the proposed auction-based

task offloading scheme. The results show superiority and improved efficiency compared to the classic greedy algorithm.

There are several potential future research directions. The benefits of a deadline-aware task offloading scheme can be observed from the users' perspective, as well. The proposed scheme can be extended to address potential penalties for failures. Task offloading among MEC servers can be introduced to improve resource utilization. Moreover, the energy consumption of the task offloading in the industrial MEC environment can also be a subject for future research.

ACKNOWLEDGMENT

This work was supported by the Ministry of Education, Science and Technological Development of Serbia.

REFERENCES

[1] Alameddine, H.A., Sharafeddine, S., Sebbah, S., Ayoubi, S., Assi, C. (2019). Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing. IEEE Journal on Selected Areas in Communications, 37(3), 668-682.

[2] Al-Shuwaili, A., Simeone, O. (2017). Energy-Efficient Resource Allocation for Mobile Edge Computing-Based Augmented Reality Applications. IEEE Wireless Communications Letters, 6(3), 398-401.

[3] Cao, X., Wang, F., Xu, J., Zhang, R., Cui, S. (2019). Joint Computation and Communication Cooperation for Energy-Efficient Mobile Edge Computing. IEEE Internet of Things Journal, 6(3), 4188-4200.

[4] Feng, W., Xu, J., Ding, Z. (2019). Multi-Antenna NOMA for Computation Offloading in Multiuser Mobile Edge Computing Systems. IEEE Transactions on Communications, 67(3), 2450-2463.

[5] Li, L., Zhou, H., Xiong, S.X., Yang, J., Mao, Y. (2019). Compound Model of Task Arrivals and Load-Aware Offloading for Vehicular Mobile Edge Computing Networks. IEEE Access, 7, 26631-26640.

[6] Liu, W., Liu, Y. (2018). Price-Based Distributed Offloading for Mobile-Edge Computing with Computation Capacity Constraints. IEEE Wireless Communications Letters, 7(3), 420-423.

[7] Luo, S., Wen, Y., Xu, W., Puthal, D. (2019). Adaptive Task Offloading Auction for Industrial CPS in Mobile Edge Computing, IEEE Access, 7, 169055-169065.

[8] Mashhadi, F., Salinas Monroy, S.A., Bozorgchenani, A., Tarchi, D. (2020). Optimal auction for delay and energy constrained task offloading in mobile edge computing, Computer Networks, 183, 107527.

[9] Sengupta, J., Ruj, S., Bit, S.D. (2021). A Secure Fog-based Architecture for Industrial Internet of Things and Industry 4.0, IEEE Transactions on Industrial Informatics, 17(4), 2316-2324.

[10] Tran, T.X., Pompili, D. (2019). Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks. IEEE Transactions on Vehicular Technology, 68(1), 856-868.

[11] Wang, F., Xu, J., Wang, X., Cui, S. (2018). Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems. IEEE Transactions on Wireless Communications, 17(3), 1784-1797.

[12] Zhang, C., Zhao, H., Deng, S. (2018). A Density-Based Offloading Strategy for IoT Devices in Edge Computing Systems. IEEE Access, 6, 73520-73530.
[13] Zhang, H., Yongjin, Y., Shang, B., Zhang, P. (2022). Joint Resource Allocation and Multi-Part Collaborative Task Offloading in MEC Systems. IEEE Transactions on Vehicular Technology, Early Access.
[14] Zhang, T. (2017). Data Offloading in Mobile Edge Computing: A Coalition and Pricing Based Approach. IEEE Access, 6, 2760-2767.
[15] Zhang, Y., Feng, B., Quan, W., Li, G., Zhou, H., Zhang, H. (2019). Theoretical Analysis on Edge Computation Offloading Policies for IoT Devices. IEEE Internet of Things Journal, 6(3), 4228-4241.