# Efficient Task Offloading in Multi-User Edge Computing

Subjects: Computer Science, Information Systems

Contributor: Chang Shu , Yinhui Luo , Fang Liu

Task offloading is one of the most important issues in edge computing and has attracted continuous research attention in recent years. With task offloading, end devices can offload the entire task or only subtasks to the edge servers to meet the delay and energy requirements.

task offloading urban Internet of Things

## 1. Introduction

The proliferation of IoT applications, e.g., object recognition, vehicular systems and self-driving [1][2][3], have pushed the horizon of a novel computing paradigm—Multi-access Edge Computing (MEC) [4]. In MEC, a number of edge servers are deployed close to the user, and the computation-intensive tasks from end users are uploaded to and processed in the network of edge servers. After that, the results are returned to end user devices [5][6]. As a result, the completion time of services and the energy consumption of end user devices can be reduced.

Traditional offloading schemes [7] upload the entire task to edge servers. These approaches fail to utilize the parallelism between end devices and edge servers. To address this issue, the recent studies [8][9] decompose the task into a number of subtasks and upload some of them to edge servers. In this way, end devices and edge servers can execute their subtasks in parallel, such that the overall completion time is reduced. For example, the researchers in [8] synthetically considered the computation workload of subtasks, dependency among subtasks and wireless transmission time. In doing this, the subtasks can be rationally scheduled in the edge server or local devices, and the makespan of applications can be reduced.

However, in the approaches based on task decomposition, more communication delay is introduced as the information exchange among subtasks that are not located in the same processor needs to be done through the wireless links. To better understand the limitations of the existing works and reveal further optimization space, researchers study an example of task offloading for the object recognition applications [10], as shown in **Figure 1**.

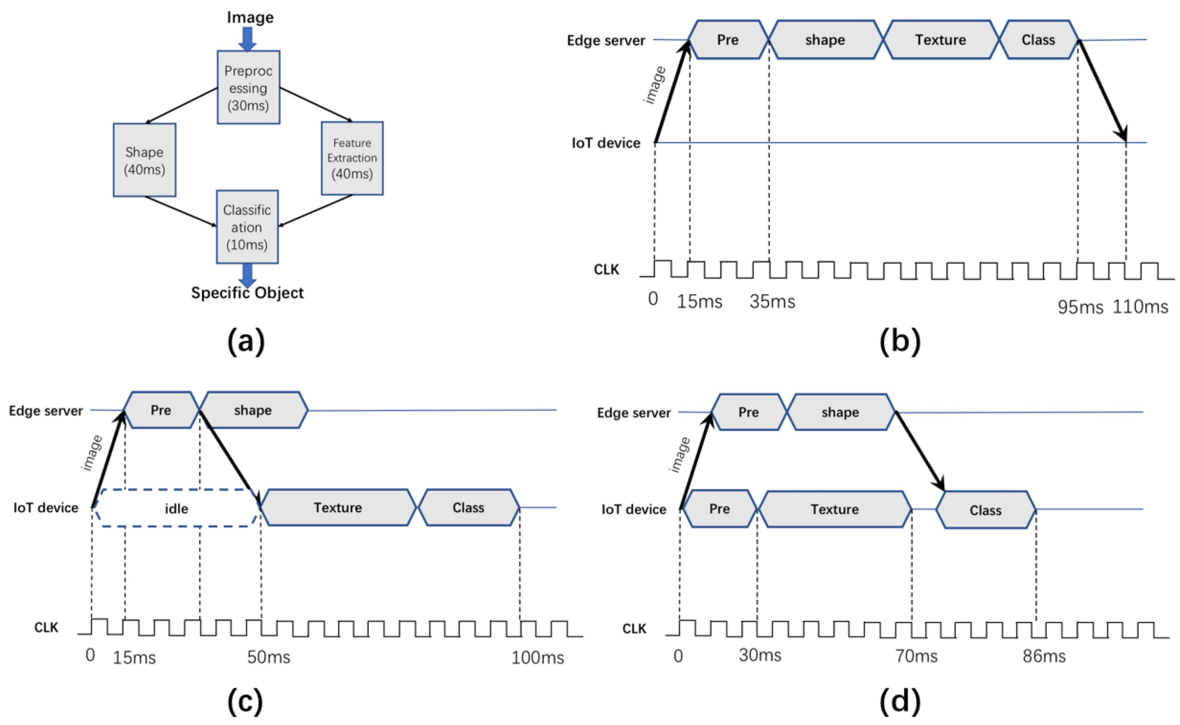

**Figure 1.** Case study. (**a**) the DAG of object recognition. (**b**) entire task offloading strategy (**c**) subtask offloading strategy. (**d**) Duplication subtasks offloading strategy.

The task topology is depicted as a directed acyclic graph (DAG) as shown in **Figure 1**a. Each rectangle denotes a subtask with the execution delay labeled. An arrowed line from rectangle A to rectangle B denotes that subtask B depends on the results of subtask A. It can be seen that the object recognition first inputs the target image to the “Pre-processing” subtask. Based on the results of “Pre-processing”, “Shape” and “Texture” will extract the object shapes and texture, respectively. Based on the extractions, the “Classification” subtask can recognize the object as a specific class of objects.

**Figure 1**b shows the basic idea of traditional offloading works [7][11][12], which treat the four subtasks as a whole and offload them to edge servers. This approach overlooks the parallelism between users and edge servers. **Figure 1**c shows the basic idea of the recent works based on task decomposition [8][13], which chooses some of the subtasks to offload, and thus the overall delay is reduced. From **Table 1**, through the offloading strategies based on task decomposition, the latency is reduced from 110 to 100 ms due to parallelism.

**Table 1.** The delay comparison.

| | Energy Cost (Local Execution Time) | Overall Delay |
|---|---|---|
| **Figure 1b** | **0 ms** | **110 ms** |
| **Figure 1c** | **50 ms** | **100 ms** |
| **Figure 1d** | **80 ms** | **86 ms** |

However, from **Figure 1**c, it can be seen that considerable time (from 0 to 50 ms) is wasted in the “idle” slot because the subtask “Texture” needs to wait for the computational results from the subtask “Pre”. The reason is that uploading subtasks requires the information exchange among them to be done through wireless links (an additional communication round is needed from “Pre” to “Texture”). To further reduce the execution delay and fully exploit the parallelism between local devices and edge servers, an intuitive yet reasonable solution is to duplicate the Pre-processing module in the local devices (**Figure 1**d), and the overall execution delay can be further reduced to 86 ms.

The key idea behind the duplication-based approach is trading the computation resources (CPU cycles required to accomplish task) for reducing communication delay. Considering that wireless interference widely exists in multi-user edge computing networks [14][15] and has a high impact on the overall performance [16][17], trading computation resources for communication efficiency is likely to achieve significant performance gains. However, implementing such an idea in real-world systems is a non-trivial task due to the following challenges:

- **How to reveal the duplication and offloading opportunities from complex task topologies.** Before trading resources for communication efficiency, firstly, to identify which subtasks are the most appropriate to be duplicated and offloaded is needed. Different combinations of the two kinds of subtasks can lead to largely different performances. Considering the diversity and complexity of potential edge applications [10], the searching space can be surprisingly large.

- **How to coordinate the duplication and offloading strategies among multiple users.** Apart from the "actual" execution time, the overall execution time consists of communication delay and task queuing delay. Each user's duplication and offloading decisions will affect wireless contentions and the subtask scheduling at both local users and edge servers and further affect the communication delay and task queuing delay. As a result, coordinate the duplication and offloading strategies among multiple users to improve the overall system performance is needed.

## 2. Related Work of Efficient Task Offloading in Multi-User Edge Computing

Recently, multi-access edge computing was recognized by the European 5G PPP (5G infrastructure public, private partnership) research body as one of the key emerging technologies for 5G networks. There have been numerous works on the computation offloading problem in the literature, and various offloading policies have been proposed, which can be classified into two stages.

For the first stage, the researchers do not distinguish subtasks inside the application and upload the entire task as a whole to the edge servers. In [18], they study the multi-user offloading problem for edge computing in a multi-channel wireless interference environment and prove that this is NP-hard to obtain the optimal solution, and then they adopted a heuristic approach for achieving efficient computation offloading in a distributed manner.

Simulation results show that the proposed algorithm achieves superior computation offloading performance and scales well as the user size increases. Furthermore, the researchers [19] studied a distributed offloading scheme for a multi-user and multi-server based on orthogonal frequency-division multiple access in small-cell networks. They formulated a distributed overhead minimization problem and then adopted the potential game theory to prove that their proposed decision is a potential game problem.

Finally, compared with other existing computation algorithms, the simulation results show that this new algorithm can guarantee saving overheads. The other works in the first stage, such as [20][21][22], also consider a multi-user and multi-server scenario and propose a heuristic offloading decision algorithm to assign different tasks to the corresponding channel and server to minimize the average completion time.

With the increasing complexity of applications, the offloading related work comes into the second stage. This task has been considered as DAGs in order to identify a fine-grained offloading opportunity. In [23], the researchers considered that the application consists of a series of sequential modules and then decide which module to be offloaded and how these modules should be executed (cloud computing or edge computing). Furthermore, the researchers in [8] investigated the problem of fine-grained task offloading in edge computing for low-power IoT systems.

## References

1. Liang, M.; Hu, X. Recurrent convolutional neural network for object recognition. In Proceedings o f the IEEE Conference On ComputerVision And Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3367–3375.
2. Mallik, A.; Ding, W.; Khaligh, A. A comprehensive design approach to an EMI filter for a 6-kW thre e-phase boost power factor correction rectifier in avionics vehicular systems. IEEE Trans. Veh. Te chnol. 2016, 66, 2942–2951.
3. Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; et al. End to end learning for self-driving cars. arXiv 2016, arXiv:1604.07316.
4. Luo, Q.; Hu, S.; Li, C.; Li, G.; Shi, W. Resource scheduling in edge computing: A survey. IEEE Com mun. Surv. Tutor. 2021, 23, 2131–2165.
5. Miao, W.; Min, G.; Zhang, X.; Zhao, Z.; Hu, J. Performance modelling and quantitative analysis of vehicular edge computing with bursty task arrivals. IEEE Trans. Mob. Comput. 2021, 13, 1357–1 368.
6. Cong, R.; Zhao, Z.; Min, G.; Feng, C.; Jiang, Y. EdgeGO: A mobile resource-sharing framework for 6g edge computing in massive IoT systems. IEEE Internet Things J. 2021.
7. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloadin g. IEEE Commun. Surv. Tutor. 2017, 19, 1628–1656.
8. Shu, C.; Zhao, Z.; Han, Y. Dependency-Aware and Latency-Optimal Computation Offloading for M ulti-User Edge Computing Networks. In Proceedings of the IEEE Conference on Sensing, Commu nication and Networking, Boston, MA, USA, 10–13 June 2019.
9. Xiao, L.; Lu, X.; Xu, T.; Wan, X.; Ji, W.; Zhang, Y. Reinforcement learning-based mobile offloading f or edge computing against jamming and interference. IEEE Trans. Commun. 2020, 68, 6114–6126.
10. Ra, M.R.; Sheth, A.; Mummert, L.; Pillai, P.; Wetherall, D.; Govindan, R. Odessa: Enabling interacti ve perception applications on mobile devices. In Proceedings of the Ninth International Conferen ce on Mobile Systems, Applications, and Services, Portland, OR, USA, 27 June–1 July 2022; ACM: New York, NY, USA, 2011; pp. 43–56.
11. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The com munication perspective. IEEE Commun. Surv. Tutor. 2017, 19, 2322–2358.
12. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile edge computing: A survey. IEEE Internet Thi ngs J. 2017, 5, 450–465.
13. Kao, Y.H.; Krishnamachari, B.; Ra, M.R.; Bai, F. Hermes: Latency optimal task assignment for reso urce-constrained mobile computing. IEEE Trans. Mob. Comput. 2017, 16, 3056–3069.
14. Zhao, Z.; Min, G.; Gao, W.; Wu, Y.; Duan, H.; Ni, Q. Deploying edge computing nodes for large-sc ale IoT: A diversity aware approach. IEEE Internet Things J. 2018, 5, 3606–3614.
15. Zhao, Z.; Min, G.; Dong, W.; Liu, X.; Gao, W.; Gu, T.; Yang, M. Exploiting Link Diversity for Perform ance-Aware and Repeatable Simulation in Low-Power Wireless Networks. IEEE/ACM Trans. Netw.

2020, 28, 2545–2558.

16. Wang, F.; Xu, J.; Wang, X.; Cui, S. Joint offloading and computing optimization in wireless powere d mobile-edge computing systems. IEEE Trans. Wirel. Commun. 2017, 17, 1784–1797.
17. Wang, C.; Liang, C.; Yu, F.R.; Chen, Q.; Tang, L. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. IEEE Trans. Wirel. Commun. 2017, 16, 4 924–4938.
18. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. 2015, 24, 2795–2808.
19. Yang, L.; Zhang, H.; Li, X.; Ji, H.; Leung, V. A Distributed Computation Offloading Strategy in Sma ll-Cell Networks Integrated With Mobile Edge Computing. IEEE/ACM Trans. Netw. (TON) 2018, 26, 2762–2773.
20. You, C.; Huang, K.; Chae, H.; Kim, B.H. Energy-efficient resource allocation for mobile-edge com putation offloading. IEEE Trans. Wirel. Commun. 2016, 16, 1397–1411.
21. Lyu, X.; Tian, H.; Sengul, C.; Zhang, P. Multiuser joint task offloading and resource optimization i n proximate clouds. IEEE Trans. Veh. Technol. 2016, 66, 3435–3447.
22. Chen, X. Decentralized computation offloading game for mobile cloud computing. IEEE Trans. Pa rallel Distrib. Syst. 2014, 26, 974–983.
23. Ning, Z.; Dong, P.; Kong, X.; Xia, F. A cooperative partial computation offloading scheme for mob ile edge computing enabled Internet of Things. IEEE Internet Things J. 2018, 6, 4804–4814.