Distributed Task Scheduling for Wireless Powered Mobile Edge Computing: A Federated-Learning-Enabled Framework

Xiaojie Wang, Shupeng Wang, Yongjian Wang, Zhaolong Ning, and Lei Guo

Abstract

Sixth generation (6G) wireless communication networks are expected to provide ultra-low latency and energy-efficient services for global users. The paradigm of wireless powered mobile edge computing (WP-MEC) provides a feasible framework to realize the above purposes by leveraging the advantage of both wireless power transfer and MEC. However, massive network traffic and control information overburden the centralized server. In this article, we integrate federated learning and deep reinforcement learning with WP-MEC to jointly optimize computing and communication resources. Specifically, we design an online learning framework to schedule computation tasks of mobile clients, where the learning model can be distributed over edge servers, and parameter synchronization happens on the cloud. The superiority of the designed framework is evaluated and compared with benchmarks in terms of average task completion delay and task completion ratio. Finally, we discuss some research challenges and opportunities for future WP-MEC.

Introduction

Although the standardization and deployment of the fifth generation (5G) wireless communication networks began in 2020, it will not meet all requirements beyond 2030, such as high accuracy of time and phase synchronization. In addition, mobile communications cover merely 20 percent of land area nowadays. To meet the requirements, such as enhanced network performance and global coverage, the investigation of the sixth generation (6G) has attracted much attention. In particular, 6G will provide services with near 100 percent geographical coverage, where remote areas, including motorways and villages, can be covered. However, current wireless nodes in rural areas are restricted to energy supply. Thus, wireless powered mobile edge computing (WP-MEC) has become a promising paradigm to support the mobile wireless network for seamless, cost-effective, and ubiquitous service availability in 6G.

Generally, WP-MEC is formed by the integration of wireless power transfer (WPT) and MEC. The WPT technology enables radio frequency (RF) signals to be transmitted from edge servers to mobile devices, making charging over the air possible. In addition, MEC allows mobile clients to offload their computation-intensive tasks to edge servers for delay minimization. However, due to the characteristics of half-duplex, the charging and task offloading processes cannot be conducted simultaneously. Here arises a vital issue: How can the above two processes be reasonably handled for system performance optimization?

Existing studies always investigate computation offloading in WP-MEC networks by optimization techniques, such as approximation and convex optimization, since time division and resource allocation are complicated with each other, and the formulated problems are always NP-hard. Currently, researchers mainly focus on computation offloading with one edge server in WP-MEC networks. For example, the authors of [1] proposed a binary offloading algorithm for jointly optimizing individual task offloading scheduling and system time allocation, with the purpose of maximizing the sum computation rate of mobile clients. The authors of [2] designed an online resource allocation strategy to maximize the fairness among mobile clients by Lyapunov optimization.

However, there is little attention paid to computation offloading with multiple edge servers in WP-MEC networks, due to its complexity in action coordination among different edge servers. The authors of [3] proposed a centralized scheduling algorithm to maximize the task computation ratio for computation offloading with multiple edge servers in WP-MEC networks. Nevertheless, it cannot be executed distributedly, and there are several challenges for designing a distributed framework for WP-MEC:

Multiple Coupling of Parameters: For the optimization problem in computation offloading with multiple edge servers in WP-MEC networks, parameters are always coupled and hard to determine. The time division of energy transmission and task offloading periods impacts the amount of harvested energy of mobile clients, which further impacts task scheduling decisions, since transmitting tasks to edge servers depends on the harvested energy of mobile clients. The task scheduling decision affects the computation and communication resource allocations, and the resource allocation result has impacts on system performance, which inversely impacts the future time division. Thus, these tightly coupled parameters make the optimization problem rather hard to resolve.

Xiaojie Wang and Lei Guo are with Chongqing University of Posts and Telecommunications; Shupeng Wang (corresponding author) is with the Institute of Information Engineering of the Chinese Academy of Sciences; Yongjian Wang is with the National Computer Network Emergency Response Technical Team/Coordination Center of China; Zhaolong Ning (corresponding author) is with Dalian University of Technology and also with Chongqing University of Posts and Telecommunications.

Digital Object Identifier: 10.1109/MNET.201.2100179

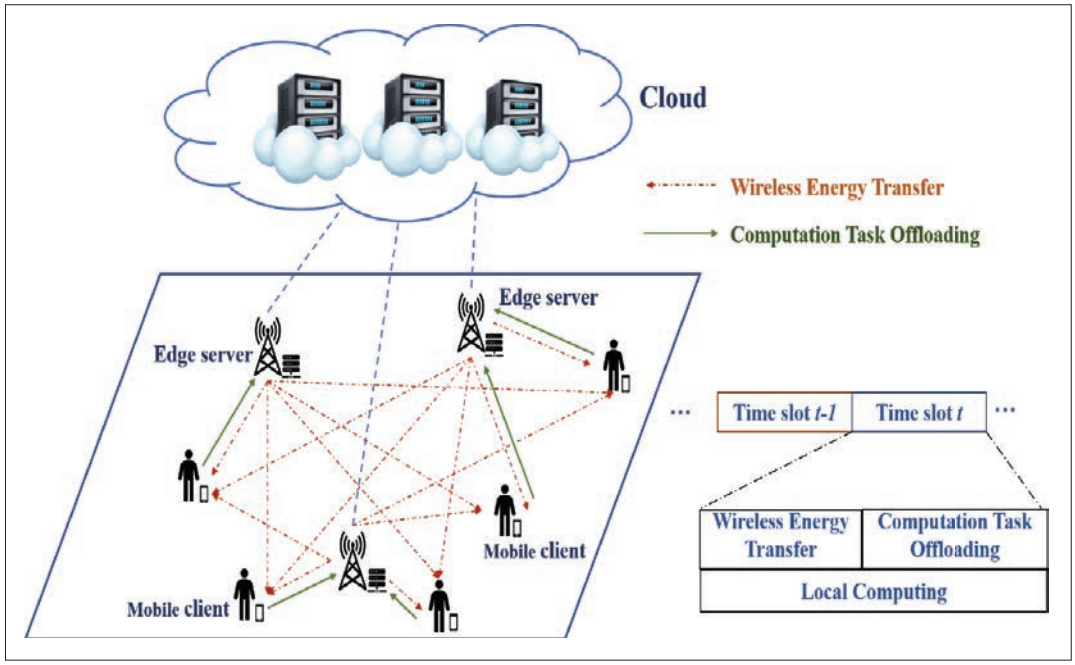

FIGURE 1. An example of a WP-MEC network.

Difficulty in Action Coordination: When the scheduling model is distributed on multiple edge servers, it is hard to coordinate their actions for online scheduling due to the fact that they are unaware of others' policies. Thus, the scheduled results of one server may not be acceptable for mobile clients, since there may be better scheduling results generated by other edge servers.

Long-Term Performance Optimization: Since the task scheduling result of mobile clients in the former time slot affects the scheduling decision of the succeeding time slots, how to optimize system performance in the long term is challenging.

To solve the above-mentioned challenges, we integrate deep reinforcement learning (DRL) and federated learning (FL) with computation offloading in WP-MEC networks. Specifically, a learning-based framework is constructed with the purpose of minimizing the average task completion delay by optimizing both time division and resource allocation for task scheduling. To the best of our knowledge, *this is the first work to investigate distributed computation offloading in WP-MEC networks*. The FL-enabled framework is established based on multiple edge servers and a remote cloud server. The learning agents are distributed on edge servers and responsible for generating task scheduling decisions for mobile clients based on DRL. A parameter synchronization process is conducted on the cloud server.

In this article, we first present the task scheduling model and then describe the FL-enabled framework in detail. We evaluate the system performance and demonstrate the algorithm superiority on various metrics. Finally, we discuss several research challenges and open issues for WP-MEC.

Task Scheduling in WP-MEC

Figure 1 is an illustrative example of the considered system. The task scheduling process can be summarized as the following parts.

Request Broadcasting: Due to their limited computation capabilities, mobile clients always broadcast their task offloading requests, including the task identity, task size, required CPU cycles, and task deadline, to edge servers.

Energy Harvesting and Task Scheduling: After edge servers receive those offloading requests, they begin to make energy transmission (the energy harvesting period for mobile clients) and task scheduling (the estimated task processing sequence with task identities) decisions based on their own policies. Meanwhile, they are also responsible for training their own policies periodically by interacting with the environment.

Result Selection: After action selection, edge servers broadcast their scheduling results to mobile clients. Then mobile clients select the most suitable edge server for task offloading by considering the task deadline and their remaining energy. Thus, edge servers receive offloaded tasks from clients, and process tasks according to their estimated processing sequence by omitting unreceived task identities, which also ensures that one task is processed by only one server.

Communication and Computation Models

We consider that there are M fixed edge servers and N mobile clients in a 2D geographical area. Mobile clients can not only harvest energy from edge servers but also offload tasks to them. Actually, a part of the campus map can be regarded as the considered 2D area, where walkers or riders holding mobile terminals act as mobile clients. The communication between clients and servers is supported by orthogonal frequency-division multiple access (OFDMA) technology. For simplicity and without lose of generality, we consider that each server processes tasks in sequence. Mobile clients can be regarded as unmoving during the period of task processing due to their limited moving speeds and the short time duration. In addition, energy harvesting and task offloading processes also cannot be conducted simultaneously. Thus, a harvest-then-offloading mode is adopted for our framework in each time slot [3]. For a generated task, it can be represented by three elements: the task size,

the required CPU cycle, and the task deadline. The task is atomic and cannot be split further. To complete a task, three parts of delay are included.

Transmission Delay: It refers to the delay consumed by sending one task from its original client to one edge server, and can be computed by the result of the task size divided by the transmission speed. In addition, the transmission speed between one mobile client and one server can be obtained by the Shannon equation [4].

Computation Delay: If the task is processed locally, its computation delay can be computed by its required CPU cycle divided by the CPU frequency of the mobile client. If the task is processed on one edge server, its computation delay can be computed by its required CPU cycle divided by the CPU frequency of the edge server.

Waiting Delay: It refers to the delay for a task waiting for scheduling. Generally, it includes two items: one is the number of time slots it has been waiting before it is scheduled, and the other is the period before it is scheduled in the current time slot.

Specifically, if the task is processed locally without offloading, the transmission delay is zero. In the following, we discuss energy harvesting and consumption for mobile clients.

Energy Harvesting: Similar to [3], we consider all edge servers to be transmitting RF power to mobile clients simultaneously within a period from the beginning of each time slot. The harvested energy by each mobile client is related to the transmission frequency of each edge server, the energy conversion efficiency, and the distance between the receiver and the transmitter. A virtual transmitter can be formed by aggregating the transmission power of all edge servers.

Energy Consumption: After energy harvesting, the task scheduling period starts. To complete a task, the consumed energy can be obtained in two parts: transmission energy and computation energy. The former can be caused when the task is offloaded to edge servers and computed by the transmission power of the mobile client multiplying by the transmission delay. The latter is mainly for local computing, and computed by the product of energy coefficient, the cube of the CPU frequency of the mobile client, and the required CPU cycle of the task.

Problem Formulation

Based on the discussion of the previous subsection, we can formulate the optimization problem with the objective of minimizing the long-term average task completion delay, and the constraint is that the consumed energy for task scheduling cannot exceed the residual energy in the current time slot. However, it is quite challenging to solve the formulated problem even in a centralized manner since it is NP-hard. We intend to construct an FL-enabled framework. The network division and cooperation among edge servers are specified, and an online scheduling algorithm is designed based on DRL, where network performance and algorithm coverage can both be guaranteed.

The FL-Enabled Framework

Since the distributed requirements make the optimization problem hard to resolve, we construct an FL-enabled framework, which enables the DRL-based task scheduling to be trained and executed in a distributed manner.

Integration of FL with WP-MEC

Learning Agent Deployment: In our framework, one fundamental issue is where to deploy learning agents in WP-MEC networks. Naturally, there are two choices: deploying on mobile clients or edge servers. If learning agents are deployed on mobile clients, three deficiencies exist: First, the computation capabilities of mobile clients are limited, and they do not have the ability to support the intensive computation of neural networks; second, the energy shortage of mobile clients can deteriorate, challenging the network stability; and third, mobile clients may only consider their own benefits while neglecting the overall network performance.

Thus, it is preferred to deploy the learning agent on edge servers in WP-MEC networks, since they not only have more powerful computation capabilities, but also have sufficient energy. Then another issue arises: how different edge servers collaborate to improve the whole performance by the distributed learning model.

Framework Overview: We consider there to be three kinds of entities: a cloud server, a set of edge servers, and mobile clients. The cloud server can synchronize parameters for learning agents deployed on edge servers. The edge servers provide computation and energy-harvesting services for mobile clients. The interactions include the following parts.

From the Edge to the Cloud: For each edge server, the learning model is trained locally with its neural networks: value network, policy network, and dual network. For each training period, the state value, the energy harvesting period, and dual parameters are sent to the cloud after edge servers finish model training. Then an update step is conducted for the three elements on the cloud server after it receives all copies from edge servers. After that, the cloud server sends back updated results to the edge. Following that, learning agents update their parameters.

From Mobile Clients to the Edge: Each mobile client first decides whether to offload their tasks to edge servers for computation based on their local computation ability. Then it sends offloading requests to edge servers. After receiving those computation requirements, edge servers make scheduling and time division decisions for all mobile clients. After energy harvesting, they decide which edge server is suitable to process their offloaded tasks based on their residual energy and the received scheduling results broadcasted by edge servers. We consider that mobile clients are fully cooperative and always choose the best task scheduling result generated by edge servers.

MDP Formulation: To solve the formulated problem, we first define it as a Markov decision process (MDP). A tuple can be utilized to represent it by (S, A, R, P,), where S is the state space, A is the action space, R is the reward, and P is the discount factor between 0 and 1. In the following, we provide a detailed description of our framework.

State: The state of the MDP includes three parts: the state of servers, mobile clients, and communication channels. The first contains the service abilities and locations. The second includes the locations, the total amount of local tasks, the total required CPU cycles, and the minimum and max-

imum life cycles of tasks for each mobile client. The third includes the transmission speed.

Action: The action taken by each edge server includes two parts: the energy harvesting period and the task scheduling decision for all mobile clients in each time slot.

State Transition: It refers to the possibility that the state in the current time slot transfers to the next time slot by taking one action.

Reward: The immediate received reward by scheduling tasks and transmitting energy to mobile clients can be computed by the opposite value of the formulated optimization objective.

Then the formulated optimization problem is transferred into a reward-maximization one, that is, finding the policy that can reach the optimal value of the formulated MDP, which can be obtained based on the accumulated reward from the current state to possible future ones.

However, such a reward-maximization problem cannot be solved directly. First, this MDP problem is formulated based on the entire system state, and the solution is based on the collaboration of all edge servers. Second, the energy constraint complicates this MDP problem. Third, solving the formulated problem in a distributed manner is rather challenging.

Distributed Learning Model Based on DRL: For the formulated optimization problem, we transform it into subproblems for each learning agent to resolve. The following three steps are included to relax the original problem.

Action Reformation: The system action includes two parts: the energy harvesting period and the task scheduling decision. For the former, all edge servers need to reach consensus to improve the system performance by making mobile clients charge with much energy in a short period. For the latter, we consider that each edge server makes its own decision for the processing sequence of task offloading. That is, if all tasks are offloaded to the edge server, how to schedule them in a reasonable sequence requires investigation. The advantages of action reformation are first, avoiding reduplicative choices for some clients with no choice for others; and second, largely reducing the communication overhead among edge servers for frequent exchanges of their model parameters.

Reward Reformation: Since edge server j chooses actions independently, the received reward can also be formed by the average delay of tasks processed by server j plus $1/M$ of the average delay of tasks processed locally, where M is the total number of edge servers. The sum reward of edge servers equals the whole system reward. To relax the energy constraint, we can add a penalty to the reward function for server j.

Subproblem Formulation: Based on the action and reward of each mobile client, we can formulate the optimization problem for each learning agent. Similar to [5], it is to minimize a theoretical gap based on the state value function, which is formed by the soft Bellman optimality. Thus, we just need to establish the neural network model, and make a synchronous update for the state value, the energy harvesting period, and the dual parameters induced by the transformed problem. Thus, an approximate solution for the original problem can be obtained by periodically training the neural network.

Training Process: The training process for each learning agent can be summarized as:

- Initialize the three neural networks locally.
- Learning agents choose actions in each time slot.
- The state, actions, and received reward are recorded and stored in edge servers.
- Train the three neural networks based on historical records.
- Send parameters to the cloud server for synchronous update.

The training process can be found in Fig. 2.

Performance Evaluation

Simulation Setup: Python 3.6 and TensorFlow 2.1 are utilized in our simulation to validate the performance of the designed FL-enabled framework, named FIEF. We consider a geographical area, the size of which is 1 km × 1 km [6]. The area is divided by intersecting lines with 104 cross points. Servers are uniformly distributed in this area, and users representing mobile clients move along lines based on the Manhattan mobility model [7]. We set simulation parameters similar to those in [8]. The number of edge servers is between 3 and 8, and that of mobile clients is between 20 and 100. The length of each time slot is set by 0.3 s, and the task generation possibility for each mobile client in each time slot is between 0.4 and 0.8. The task size is set between 50 kbits and 100 kbits. The CPU frequency of each mobile client is 500 MHz and that of each edge server is 2.5 GHz. The energy harvesting efficiency of mobile clients is 0.51. The transmission power of edge servers is 3 W and that of mobile clients is 0.1 W. Multi-layer perceptions are leveraged for the construction of the learning model, and the Momentum Optimizer [9], Adam Optimizer [10], and Softmax activation function are utilized for model training. Since there are no existing distributed computation offloading algorithms in WP-MEC networks, we utilize the following two benchmark algorithms for comparison:

Energy-Aware Scheduling (EAS) [11]: Mobile clients select the edge server that can minimize their energy consumption. The division of energy harvesting period and task scheduling period in each time slot is the same as in FIEF.

Local Computing (LC): Mobile clients process all their tasks locally, and the energy harvesting time is set by the length of each time slot.

Simulation Results: Figure 3 shows the trend of average task completion delay with different numbers of mobile clients. We can observe that the designed algorithm, FIEF, is always the best. The performance of EAS has a gap with that of FIEF, and that of LC is the worst. This is because FIEF aims to minimize the average completion delay by designing an FL-based framework, and makes decisions for task scheduling based on DRL. With a set of training iterations, the performance of FIEF becomes better and better. EAS aims to minimize energy consumption of mobile clients. Thus, they may not select the edge server that can minimize the task completion delay, but instead the one that can minimize their energy consumption. Due to the limited CPU cycles of mobile clients, the LC algorithm spends much time in processing tasks locally and consumes more energy. Consequently, its performance is the worst.

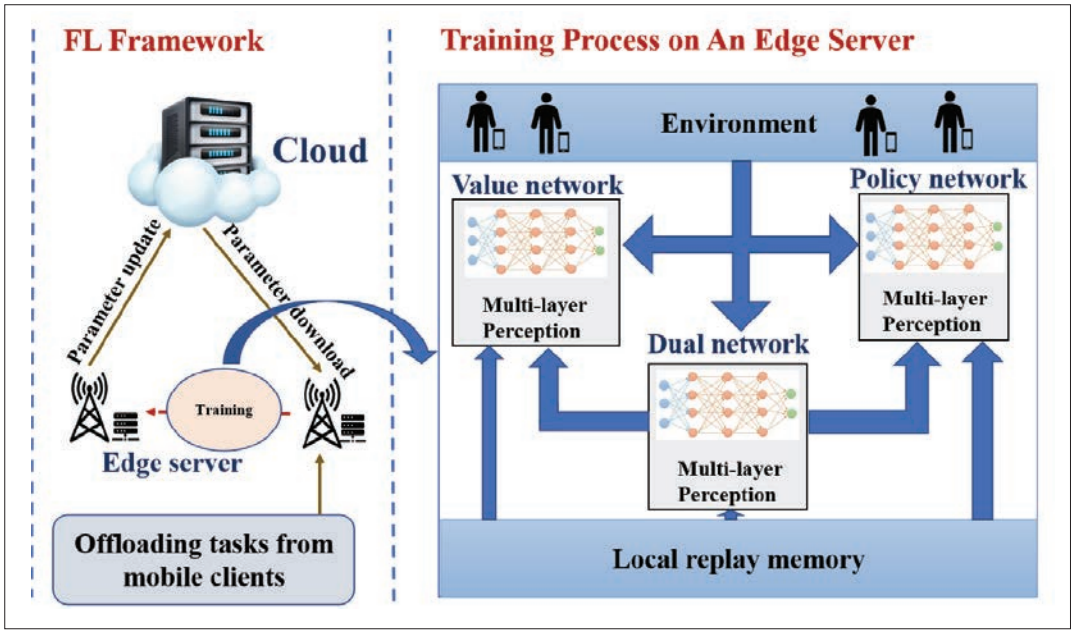

FIGURE 2. The training process in the FL-enabled framework.

We also evaluate average task completion delay with different numbers of edge servers in Fig. 4. When the number of edge servers becomes large, the average task completion delay of the three algorithms becomes small. The reason is that when there are more edge servers, more computing resources are available for mobile clients, and they have more choices for task offloading. Comparing the three algorithms, the FIEF algorithm is always the best, while EAS is second, and the performance of LC is far from that of the other two algorithms. This is because FIEF can learn good policies for task scheduling, while the others cannot make good decisions to minimize average task completion delay.

Figure 5 illustrates average task completion delay with different task generation possibilities. We can observe that when the possibility becomes high, average task completion delay also increases. This is because the increase of the task generation possibility means more tasks generated in one time slot. Thus, when the computing resources are fixed, more tasks can indirectly cause longer task completion delay.

Figure 6 is the trend of average task completion ratio with different numbers of mobile clients, where average task completion ratio refers to the ratio of total completed tasks before their deadline and total generated tasks. It is observed that the designed FIEF algorithm performs better than the other two algorithms. This is because reasonable edge servers can be found for different tasks to minimize their completion delay based on the designed framework. However, with the limited processing abilities, LC spends much time on local task processing, resulting in more out-of-date tasks. EAS aims to minimize energy consumption first, and may not always select a reasonable edge server for delay minimization.

Discussion: The advantage of FL lies in its distributed characteristic, which allows each learning agent to train its model with periodic parameter synchronization and does not require centralized training. Based on this benefit, several improvements can be made compared to traditional centralized computation offloading in WP-MEC.

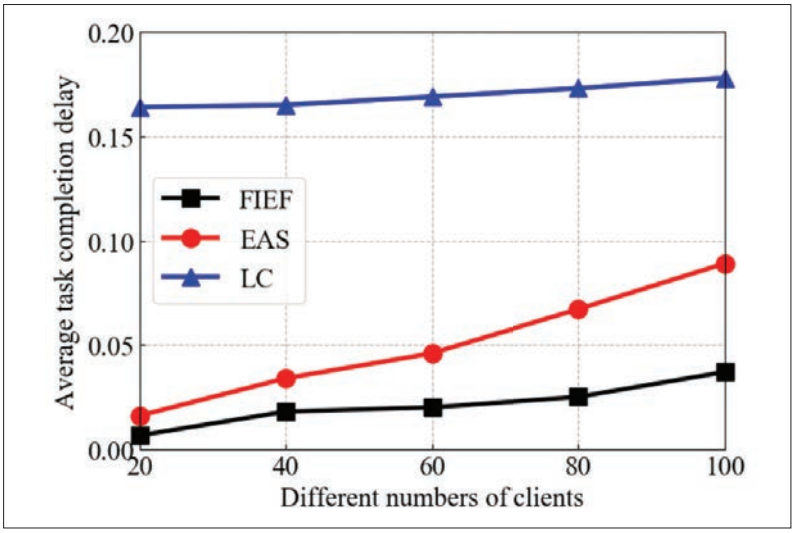

FIGURE 3. Average task completion delay with different numbers of mobile clients.

Network Scalability: In traditional WP-MEC networks, the centralized algorithm is designed for a specific number of mobile clients and edge servers. When there are newly joined clients, the scheduling algorithm needs to be redesigned and becomes complex. Thus, a heavy computation burden is posed on the centralized server. In our FL-enabled framework, the computation burden is distributed on different edge servers. When there are more computation offloading requests, each edge server merely needs to decide a processing sequence based on the current network state. The mobile clients can randomly join and leave the network without interruption of the normal operation.

Communication Efficiency: In our DRL-based task scheduling algorithm, several parameters need to be updated through the consensus awareness of edge servers. With FL, edge servers do not need to frequently exchange information, which

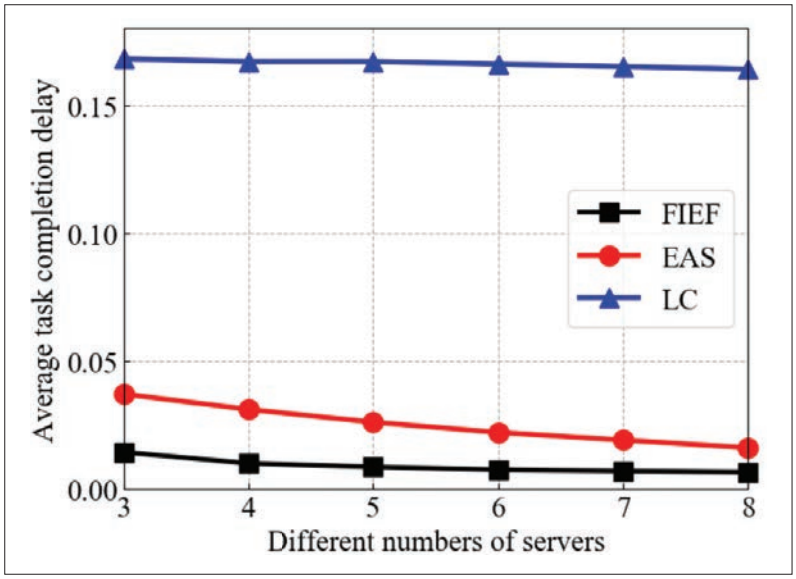

FIGURE 4. Average task completion delay with different numbers of edge servers.

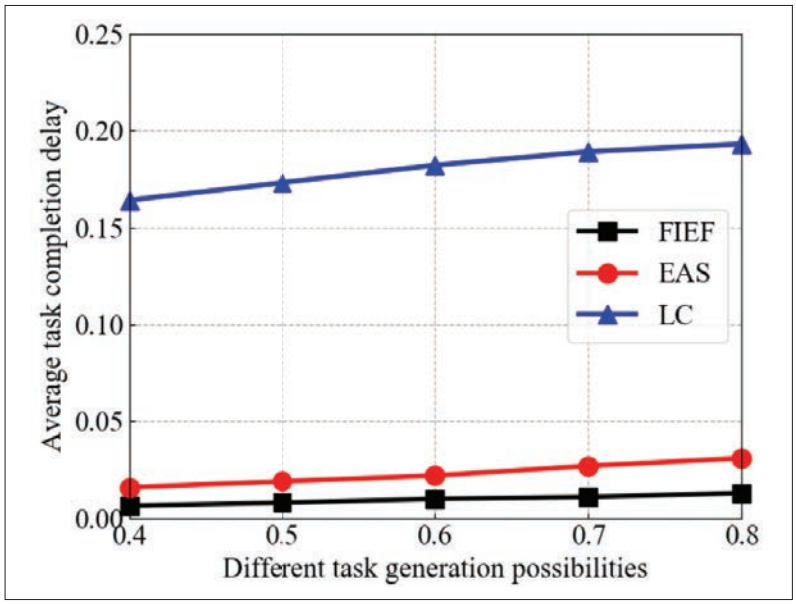

FIGURE 5. Average task completion delay with different task generation possibilities.

can largely reduce the system communication overhead. The system merely needs to update some parameters on the cloud side periodically. Meanwhile, the wireless channels among mobile clients and edge servers can be efficiently utilized, since communication requirements are largely reduced for edge servers.

Training Flexibility: Since the learning agent is distributed on different edge servers, their models can be trained based on their own demands. For example, more training rounds can be used to improve the algorithm convergence, and training parameters can be fine-tuned based on their algorithm performance. Thus, the learning agent deployed on each edge server has some degree of autonomy on model training.

Research Challenges and Open Issues

In this section, we discuss several research challenges and open issues for the future development of WP-MEC.

Trade-off between Energy and Achieved Performance

In WP-MEC networks, both energy and computation resources can be provided for mobile clients. On one hand, this brings some advantages: supporting energy-limited devices for normal operation, especially in some rural areas where there are few charging stations; and providing edge computing for mobile clients like traditional MEC networks. On the other hand, there are some challenges to be addressed. For example, if mobile clients spend more time on energy harvesting, the task scheduling period is short in each time slot, and then the network performance related to computation tasks may be worse based on the squeezed scheduling period. Although some existing studies have addressed the trade-off between energy and some metrics such as the computing possibility and the execution cost [12], some progress is still needed on the balance between energy and network performance. For example, if the requirements in different time periods are not the same, how to design an adjustment algorithm to meet the time-varying purpose from a long-term perspective is rather challenging.

Decentralized WP-MEC

6G intends to provide 100 percent global coverage for ground users, even in rural areas. In this way, many sub-control centers can exist to manage the network in a distributed manner. Thus, the centralized task scheduling algorithm based on WP-MEC may not be suitable for large-scale networks. In addition, centralized computing can pose a heavy burden on one server, and cause communication congestion between the server and mobile clients [13]. Thus, distributed algorithms and decentralized execution for WP-MEC call for further investigation. However, it is not easy to realize distributed scheduling, because cooperation among edge servers is needed to handle the offloading and energy harvesting requirements in a system-wide area.

Algorithm Optimization

The mutual impacts of energy harvesting and task scheduling periods make the optimization problem NP-hard, even with centralized control. Existing researchers always formulate the optimization problem of task offloading as a mixed integer programming (MIP) problem. To solve it, an approximation method and convex optimization are utilized. However, optimal scheduling results cannot always be discovered, and suboptimal solutions are derived. Thus, how to find scheduling algorithms with better network performance still deserves to be investigated [14]. In addition, how to integrate other kinds of resources with computation to improve the network performance is advocated. For example, the joint optimization of computation, communication, and caching resources in WP-MEC has not been investigated. Furthermore, how to apply DRL in WP-MEC with large state and action spaces needs to be solved, while guaranteeing the algorithm accuracy and convergence.

Security and Privacy

The system security and user privacy in WP-MEC networks are still unaddressed. The traditional security and privacy mechanisms for MEC networks are not efficient enough for WP-MEC net-

works. This is because the combination of energy harvesting and task scheduling brings new challenges. Malicious users can not only attack normal clients and edge servers for privacy data, but also launch energy attacks to abuse server power consumption [15]. In addition, the centralized server managing all user-related information is vulnerable to attackers, which may violate user privacy once controlled illegally. In FL, each learning agent keeps its training data, and secure aggregation and differential privacy can be applied for the training process, avoiding leakage of privacy-sensitive data. However, little attention has been paid to the integration of FL and WP-MEC.

Conclusion

WP-MEC is a promising platform to support 6G for realizing ultra-low-latency and energy-efficient services. This article has proposed an FL-enabled computation offloading algorithm in WP-MEC networks, with the purpose of minimizing the long-term average task completion delay. Specifically, we have designed an online learning framework by optimizing computation and communication resources, and trained the learning algorithms in a distributed manner over edge servers. Performance results demonstrate the effectiveness of our designed algorithm from the aspects of average task completion delay with different metrics. Finally, we discuss several research challenges and open issues for future development of WP-MEC.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 61971084, 61931019, and 62001073, by the Chongqing Talent Program CQYC2020058659, by the National Natural Science Foundation of Chongqing under Grants cstc2021ycjh-bgzxm0039 and cstc2021jcyj-msxmX0031, by the Support Program for Overseas Students to Return to China for Entrepreneurship and Innovation under Grants cx2021003 and cx2021053, and by the Dalian Young Science and Technology Star 2020RQ002.

References

[1] S. Bi *et al.*, "Computation Rate Maximization for Wireless Powered Mobile-Edge Computing With Binary Computation Offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, 2018, pp. 4177–90.
[2] H. Mirghasemi *et al.*, "Optimal Online Resource Allocation for SWIPT-Based Mobile Edge Computing Systems," *Proc. IEEE WCNC*, 2020, pp. 1–8.
[3] T. Zhu *et al.*, "Computation Scheduling for Wireless Powered Mobile Edge Computing Networks," *Proc. IEEE INFOCOM*, 2020, pp. 596–605.
[4] K. Zhang, *et al.*, Adaptive Digital Twin and Multi-agent Deep Reinforcement Learning for Vehicular Edge Computing and Networks. *IEEE Trans. Industrial Informatics*, 2021. DOI: 10.1109/TII.2021.3088407.
[5] C. Qu *et al.*, "Value Propagation for Decentralized Networked Deep Multiagent Reinforcement Learning," *Proc. Advances Neural Info. Processing Sys.*, 2019, pp. 1184–93.
[6] X. Wang *et al.*, "Multi-Agent Imitation Learning for Pervasive Edge Computing: A Decentralized Computation Offloading Algorithm," *IEEE Trans. Parallel and Distributed Systems*, vol. 32, no. 2, 2020, pp. 411–25.
[7] G. Walunjkar *et al.*, "Simulation and Evaluation of Different Mobility Models in Disaster Scenarios," *Proc. IEEE RTEICT*, 2019, pp. 464–69.
[8] F. Wang *et al.*, "Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, 2017, pp. 1784–97.
[9] S. Ilya *et al.*, "On the Importance of Initialization and Momentum in Deep Learning," *Proc. ICML*, 2013, pp. 1139–47.

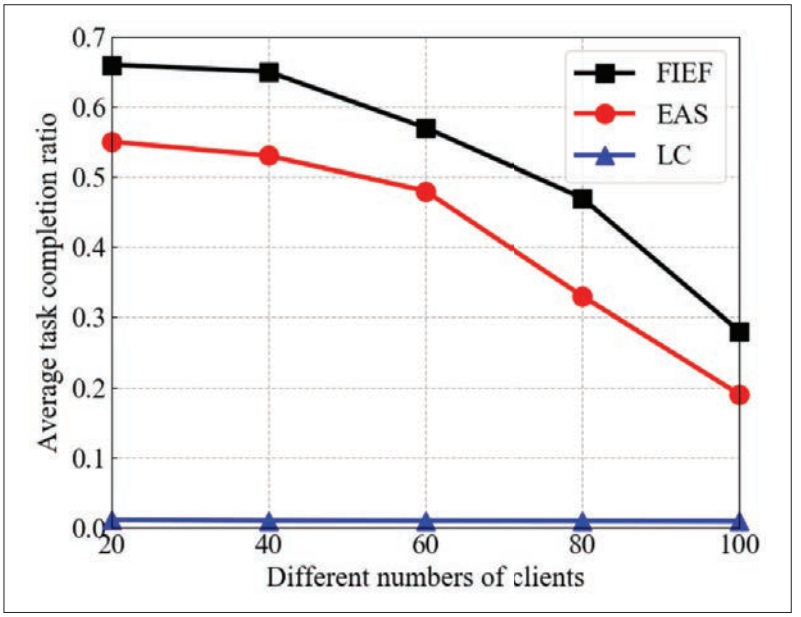

FIGURE 6. Average task completion ratio with different numbers of mobile clients.

[10] D. P. Kingma *et al.*, "Adam: A Method for Stochastic Optimization," *Proc. ICLP*, 2015, pp. 1–11.
[11] E. Meskar *et al.*, "Energy Aware Offloading for Competing Users on a Shared Communication Channel," *IEEE Trans. Mobile Computing*, 2016, vol. 16, no. 1, pp. 87–96.
[12] Y. Mao *et al.*, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," *IEEE JSAC*, 2016, vol. 34, no. 12, pp. 3590–3605.
[13] K. Zhang *et al.*, "Transfer Learning for Distributed Intelligence in Aerial Edge Networks," *IEEE Wireless Commun.*, 2021.
[14] Y. Wu *et al.*, "Digital Twin Networks: A Survey," *IEEE IoT J.*, 2021. DOI: 10.1109/JIOT.2021.3079510.
[15] Z. Wu *et al.*, "Energy Attack on Server Systems," *Proc. WOOT*, 2011, pp. 62–70.

Biographies

Xiaojie Wang (xiaojie.kara.wang@ieee.org) received her Ph.D. degree from Dalian University of Technology, China, in 2019. From 2019 to 2021, she was a postdoctoral researcher at Hong Kong Polytechnic University. Currently, she is a Distinguished Professor at Chongqing University of Posts and Telecommunications, China. Her research interests are wireless networks, mobile edge computing, and machine learning.

Shupeng Wang (wangshupeng@iie.ac.cn) received his M.S. and Ph.D. degrees from the Harbin Institute of Technology, China, in 2004 and 2007, respectively. He is currently a senior engineer with the Institute of Information Engineering, Chinese Academy of Sciences (CAS), Beijing. His research interests include big data management and analytics, network storage, and so on.

Yongjian Wang (wyj@cert.org.cn) received his M.Sc. and Ph.D. degrees in communication engineering from the Harbin Institute of Technology in 2001 and 2005, respectively. He is currently a professor with the LAB of National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC). His research interests focus on communication signal processing, cognitive radio, 5G, and the security of IoT.

Zhaolong Ning (z.ning@ieee.org) received his M.S. and Ph.D. degrees from Northeastern University, Shenyang, China. He was an associate professor at Dalian University of Technology. Currently, he is a full professor with Chongqing University of Posts and Telecommunications. He has published more than 100 scientific papers in international journals and conferences. His research interests include the Internet of Vehicles, mobile edge computing, and artificial intelligence.

Lei Guo (guolei@cqupt.edu.cn) received his Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, in 2006. He is currently a full professor at Chongqing University of Posts and Telecommunications. He has authored or coauthored more than 200 technical papers in international journals and conferences. His current research interests include communication networks, optical communications, and wireless communications.