IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 10, 2021, accepted February 18, 2021, date of publication February 22, 2021, date of current version March 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3061105

Ultra-Low Latency Multi-Task Offloading in Mobile Edge Computing

HONGXIA ZHANG[1], YONGJIN YANG[1], XINGZHE HUANG[1], CHAO FANG[2,3], (Senior Member, IEEE), AND PEIYING ZHANG[1]

[1]College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China
[2]Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[3]Purple Mountain Laboratory: Networking, Communications and Security, Nanjing 211111, China

Corresponding authors: Peiying Zhang (zhangpeiying@upc.edu.cn) and Hongxia Zhang (zhanghx@upc.edu.cn)

This work was supported in part by the Major Scientific and Technological Projects of China National Petroleum Corporation (CNPC) under Grant ZD2019-183-004, in part by the Fundamental Research Funds for the Central Universities under Grant 20CX05019A, and in part by the Shandong Provincial Natural Science Foundation under Grant ZR2020MF006.

ABSTRACT With the development of computer technology, computational-intensive and delay-sensitive applications are emerging endlessly, and they are limited by the computing power and battery life of Smart Mobile Devices (SMDs). Mobile edge computing (MEC) is a computation model with great potential to meet application requirements and alleviate burdens on SMDs through computation offloading. However, device mobility and server status variability in the multi-server and multi-task scenario bring challenges to the computation offloading. To cope with these challenges, we first propose a parallel task offloading model and a small area-based edge offloading scheme in MEC. Then, we formulate the optimization problem to minimize the completion time of all tasks, and transform the problem into a deep reinforcement learning-based offloading scheme by Markov decision approach. Furthermore, we present a deep deterministic policy gradient (DDPG) approach for obtaining the offloading strategy. Experimental results demonstrate that the DDPG- based offloading approach improves long-term performance by at least 19% in ultra-low latency, efficient usage of servers, and frequent mobility of SMDs over traditional strategies.

INDEX TERMS Mobile edge computing, computation offloading, multi-server, multi-task, deep reinforcement learning, deep deterministic policy gradient.

I. INTRODUCTION

In recent years, the widespread use of Smart Mobile Devices (SMDs) is accelerating the massive growth of computation-intensive and delay-sensitive mobile applications, such as online videos, virtual reality (VR), and augmented reality (AR). SMDs refer to wearable devices, mobile terminals, and intelligent vehicle terminals, which have the characteristics of mobility. Although SMDs have larger storages and stronger computing capabilities, it is still hard to keep up with the ever-growing demand for these complicated applications [1], [2]. Task offloading provides a promising solution to migrate computation-intensive tasks to powerful remote computing platforms (e.g., cloud servers), it reduces the response delay and energy consumption for local resources constraining.

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

Mobile cloud computing (MCC) is a successful computing paradigm of implementing task offloading. However, due to the arrival of the era of the Internet of Everything, the centralized processing mode of cloud computing data center has its problems, for example, the linear growth of centralized analysis and processing power in cloud computing data centers cannot match the explosive growth of massive amounts of data. Compared to MCC, Mobile Edge Computing (MEC) is regarded as a promising technology to offload tasks with stringent delay requirements [3]. In MEC systems, tasks can be offloaded from SMDs to the mobile network edge infrastructures, such as small cell base stations (BSs) with computation and storage resources, which can significantly extend SMDs' computing capabilities. The objective of MEC is to leverage physical proximity to SMDs, reduce latency, ensure effective network operation, and finally offer an enhanced user experience [4]. However, offloading tasks to BSs can incur extra communication latency overhead in the aspects of

delay and energy consumption [5]. Therefore, the offloading decision becomes a vital issue, and some research results have been obtained in recent researches.

The existing research on the problem of offloading mainly focuses on the multi-server MEC environment [6]–[8] and multi-user multi-task environment [9]–[11]. In the multi-server MEC environment, considered the task offloading and the resource allocation, the authors of [6] proposed an innovative heuristic algorithm to address the optimization problem jointly which seeks to maximize the task offloading gains. The authors of [7] proposed a reinforcement learning offloading scheme with a long-term utility, in order to achieve an efficient distributed offloading of multiple edge nodes under, which each mobile user can maximize the number of central processor unit cycles. The authors of [8] proposed an effective distributed algorithm to address the joint optimization problem of the computation offloading with non-orthogonal multiple access, in order to minimize the total energy consumption of Internet of Things devices with the required time limit. In the multi-user and multi-task scenarios, the authors of [9] proposed a computing offloading algorithm to reduce the energy consumption through the joint optimization of user association and computation offloading in consideration of transmission power allocation and computation resource allocation. The authors of [10] proposed a high-performance offloading management scheme in the small-cell-networks MEC system to minimize the energy consumption of all user devices through jointly optimizing computation offloading and computation resource allocation (i.e., spectrum, power, and computation resource). The authors of [11] proposed an efficient online resource allocation scheme to minimize the total energy consumption of the MEC system within a limited time constraint, while taking into account the actual situation with temporary task status information and channel status information.

However, since mobility is a typical feature of SMDs, mobility will cause dynamic communication environments and overloaded edge servers of hot areas in the MEC system with the MEC system. Therefore, minimizing delay-time in a multi-server, multi-SMD and multi-task computation offloading problem in mobile edge computing still imposes two key challenges.

1) Inefficient usage of MEC servers: Computing capacities of edge servers are limited, which leads to long complete time and poor user experience, when many SMDs offload tasks to the same edge server simultaneously. How to balance tasks between multiple edge servers and the cloud-like server reasonably is a problem to be solved in the task offloading.

2) Frequent mobility of SMDs: Smart mobile devices are on-the-move, which causes a dynamic communication environment and affects the upload delay in tasks offloading.

To address these challenges, we design a Deep Deterministic Policy Gradient-based task offloading approach that can achieve the efficient use of edge servers and satisfy the frequent mobility of SMDs in a multi-server multi-SMD and multi-task MEC environment. First, we first propose a parallel offloading model, where each task generated by the SMDs can be divided into two parts, which can be offloaded to the corresponding servers for parallel execution. The parallel offloading model prevents the server from being overloaded and address inefficient usage of edge servers. For example, the task request for offloading to the edge server in hot spot areas can be divided into a smaller task scale, and the task request for offloading to the edge server in light spot areas can be divided into a more extensive task scale. Next, a small areas-based offloading scheme is designed in the MEC model, in which the signal coverage area is divided into many small areas. Since the offloading strategy depends on small area rather than SMD, the parallel offloading model address the frequent mobility problem. Then, we formulate an optimization problem of task offloading, which minimizes the completion time of all computation tasks in the multi-server multi-SMD and multi-task MEC scenario. After that, we transform this problem into an offloading scheme based on deep reinforcement learning through the Markov decision approach. We incorporate the dynamic communication environment and the variable server status information over continuous time slots into the problem transformation process and solve the offloading scheme through a Deep Deterministic Policy Gradient-based offloading approach. Finally, the experimental results show that our approach has a stable ultra-low delay when offloading tasks under continuous movement conditions.

The main contributions are as follows.

1) A parallel offloading model in the in a multi-server multi-SMD and multi-task MEC environment is presented, which makes effectively use of all MEC servers and ensures the completion of tasks with ultra-low latency.

2) A small area-based offloading scheme is designed, which fully takes the frequent mobility of SMDs into consideration and makes the completion time of tasks generated by SMDs under continuous movement conditions stable.

3) A Deep Deterministic Policy Gradient-based is presented, which is feasible for task offloading in ultra-low latency in multi-server multi-SMD and multi-task MEC and improves long-term performance by at least 19% in ultra-low latency.

The rest of this paper is organized as follows. Section II reviews related work. The MEC system model and the formation of an optimization problem is introduced in Section III. In Section IV, the Markov decision process is used to transform the optimization problem and present the Deep Deterministic Policy Gradient-based approach. The performance evaluation is presented in Section V. Finally, we summarize this article in Section VI.

II. RELATED WORK

In the era of Internet of Everything, computing offloading in mobile edge computing has attracted the considerable attention of researchers. The current research can be discussed from two aspects: computation offloading framework, and deep learning-based optimization method.

A. COMPUTATION OFFLOADING FRAMEWORK

To achieve the computation offloading, the works [13]–[16] mainly studies how to minimize the delay or energy consumption in the binary offloading mode. In order to minimize the execution cost expressed by the execution time delay and task failure rate, the authors of [15] propose a dynamic MEC offloading algorithm based on the Lyapunov optimization, which comprehensively considers the offloading strategy, the frequencies of central processor unit cycle, and the transmit power for computation offloading. In order to maximize the (weighted) sum computation rate of all the wireless devices in the binary computation offloading mode, the authors of [16] propose a joint optimization method with the aid of the alternating direction method of multipliers decomposition technique, which effectively solves the difficulty of strong coupling between transmission time allocation and multi-user offloading mode selection.

While comparing with the binary offloading, the partial offloading mode [17], [18] has more advantages in MEC due to the characteristic of parallelism [19]. To address the problem of optimal resource allocation, the authors of [17] propose a sub-optimal resource allocation algorithm, which minimizes the weighted sum energy consumption while meeting the constraint of computation latency. To minimize the weighted sum of terminal energy consumption in the partial computation offloading mode, the authors of [18] propose an optimization algorithm, which comprehensively considers the optimization of terminal execution policy, wireless communication resource allocation, and MEC computation resource allocation.

The MEC system is also a pivotal point to the computation offloading framework. Compared with the single-user MEC system [20], [21], the researches on the multi-user system are more in line with the actual situation. In terms of the multi-user MEC system [5], [22]–[25], the authors of [24] propose a modified genetic algorithm, which addresses a constrained multi-objective optimization to balance between task latency and the system energy consumption, thus satisfying the user needs of a variety of Internet of Things applications. The authors of [5] design an evolutionary algorithm to solve the constrained multi-objective optimization problem in the task offloading model, which minimizes both the mobile devices' energy consumption and task execution latency. The authors of [25] design centralized and distributed Greedy Maximal Scheduling algorithm, which can obtain the task offloading strategy under multi-user multi-task green mobile edge cloud system, and the energy harvesting strategy with the aid of Lyaponuv Optimization Approach is innovative in green MEC system.

The above researches have made outstanding contributions to computation offloading in MEC. Learning from the above research, the partial offloading mode and resource allocation under multi-user system are considered in our paper. However, balancing edge servers to efficient usage of servers has not been considered in detail either in binary or partial offloading modes, and the frequent mobility of multiple users should be considered in a multi-user MEC system.

B. DEEP LEARNING-BASED METHOD

Recently, deep learning technology has been widely used to solve optimization problems in the edge computing paradigm. The authors of [26] propose a robust mobile crowd sensing (RMCS) architecture that creatively combines deep learning method and edge computing, which provide robust data validation and local data processing. The authors of [27] propose a multi-agent deep reinforcement learning algorithm under the edge computing paradigm, which effectively encourages mobile users to participate in sensing activities and contribute high-quality data. The authors of [28] design a deep reinforcement learning based mobile offloading scheme for mobile devices, which supports deep learning to choose the offloading policy without being aware of the task generation model, the edge computing model and jamming/interference model. The authors of [29] propose a new Deep Reinforcement Learning algorithm, which obtains an optimal computation offloading and resource allocation strategy for minimizing system energy consumption. The authors of [30] leverage Lyapunov optimization technique to transform the stochastic computation offloading and resource allocation problem into a deterministic per-time slot problem, and design an Asynchronous Actor-Critic algorithm to find the optimal stochastic computation offloading policy in a new paradigm Digital Twin Networks. The above research inspired us to solve the problem of computation offloading optimization in a MEC system. Learning from them, we combine the deep reinforcement learning technology with the proposed MEC system to obtain the offloading strategy.

In this paper, we propose a parallel offloading model in multi-server multi-SMD and multi-task MEC system, which can solve the challenges of inefficient usage of MEC servers and frequent mobility of SMDs. Furthermore, we formulate a multi-constraint optimization problem to minimize the completion time of all tasks, and transform problem into a deep reinforcement learning-based offloading scheme by Markov decision approach. Finally, we present a deep deterministic policy gradient approach for obtaining the offloading strategy.

III. SYSTEM MODEL

In this section, the scenario of MEC computation offloading is first introduced and then modeled.

MEC is widely used in 5G network environment, especially in mobile scenarios, such as AR, unmanned aerial vehicle (UAV) coordination, autonomous driving, and real-time video analysis. For example, a MEC system is deployed in a larger community, as shown FIGURE 1, AR applications can help pedestrians identify landmarks to reach their destinations quickly. However, on some special occasions with dense mobile personnel, such as football fields and hospitals, AR applications have the problem of service response to excessive delay and even failure. Therefore, our work is

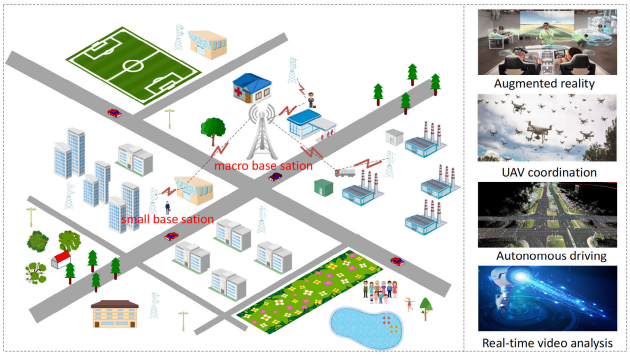

FIGURE 1. The scenario with multiple mobile devices in mobile edge computing.

to design a MEC offloading strategy for computation tasks generated by real-time SMDs to minimize the completion time of all tasks. Then we formulate the system model to simplify this scenario, i.e., the MEC system with multi-server multi-SMD and multi-task. In the end, the communication model and the computation model are introduced in detail.

Then we formulate the system model to simplify this scenario, i.e., the MEC system with multi-server multi-SMD and multi-task. In the end, the communication model and the computation model are introduced in detail.

A. MOBILE EDGE COMPUTING SYSTEM WITH MULTI-SERVER MULTI-SMD AND MULTI-TASK

We propose a mobile edge computing system consisting of one Macro Base Station (MBS) and m Small Base Stations (SBSs). In this system, base stations (BSs) configured with the corresponding MEC servers can provide computing offloading services for resource-constrained SMDs, as shown in FIGURE 2. It is considered that the capacity of the MBS server is much higher than SBS servers. Within the signal coverage of MBS, there are densely distributed SBSs and randomly distributed SMDS. Each SMD communicates with the MBS and one SBS associated with it simultaneously. The tasks can be computed by the MBS server and one SBS server at the same time. We assume that each MEC server has a task queue, and the tasks arriving at the MEC server are first cached in the queue, and then served on a first-come-first-served basis.

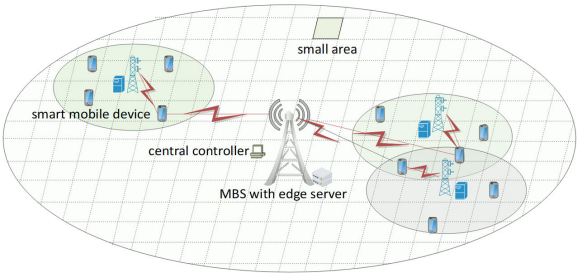

FIGURE 2. The mobile edge computing system with multiple mobile devices.

The MBS coverage area is divided into E small areas on average, and SMDs in a small area have the same distance to the base station. The division of small areas has no apparent relationship with the deployment of SBS. At the same time, because the range of each small area is smaller than the coverage range of the SBS, at most one base station can be deployed in each small area. The control center plays a scheduling role according to the offloading strategy, and it obtains the location information, task information of each SMD, and the status information of each server. Each task has a corresponding offloading pattern (offloading object and offloading bits), the same type of tasks generated in the same small area have the same offloading pattern, and the offloading strategy consists of the offloading patterns of multiple types of tasks in E small areas.

We use serve_0 to represent the MBS server, and $\text{serve}_1, \text{serve}_2, \ldots, \text{serve}_m$ to represent the SBS servers. We assume that there are G types of tasks. A task is described as a two-tuple $k_i = (d_i, c_i), i \in G$, where d_i represents the data size (bits) of the task k_i and c_i represents the number of CPU cycles required by one bit of the task k_i. Each task generated by the SMD will be divided into two parts of any size for parallel offloading. In the computation offloading scenario, different SMDs in the same small area have the same offloading pattern. Then, we design the parallel task offloading model for task k_i, as shown in FIGURE 3. The small area index of SMD $j \in \{1, 2, \ldots, N\}$ is $e_j \in \{1, 2, \ldots, E\}$, and the task k_i generated in this area can be divided into two parts and transmitted to two servers (MBS server and one SBS server) for parallel execution. The data size offloaded to the MBS server is $d_{i,e_j,0}$. The complete time of the data offloaded to the MBS server is $t_{i,e_j,0}$. The data size offloaded to the SBS server bs is $d_{i,e_j,bs}$. The complete time of the data offloaded to the SBS server bs is $t_{i,e_j,bs}$. In this model, to ensure that task k_i generated by SMD j is fully offloaded to the edge server, a task constraint is defined as $d_i = d_{i,e_j} = \sum_{bs=1}^{m} d_{i,e_j,b\,s} + d_{i,e_j,0}$. Due to the parallelism, the complete time of the task k_i is $t_{i,e_j}^{\text{total}} = \max\left(t_{i,e_j,0}, t_{i,e_j,b\,s}\right)$. The notations used in this paper are listed in Table 1.

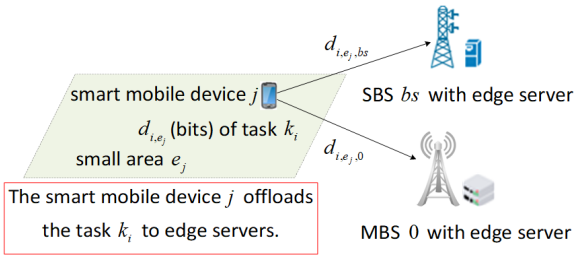

FIGURE 3. The parallel task offloading model.

B. COMMUNICATION MODEL

In the MEC system with multi-server and multi-SMD, the technology of orthogonal frequency division multiple access is adopted for communication between SMDs and

TABLE 1. Notations.

Notation	Definition
m	The number of SBS servers
N	The number of SMDs
G	The number of task types
E	The number of small areas
d_i	The data size (bits) of the task k_i
c_i	The number of CPU cycles required by one bit of the task k_i
e_j	The small areas where SMD j is located
$d_{i,e_j,bs}$	The data size of the task k_i from SMD j to SBS server bs
$d_{i,e_j,0}$	The data size of the task k_i from SMD j to MBS server
$t_{i,e_j,bs}$	The compete time when the task k_i generated by SMD j is offloaded to SBS server bs
$t_{i,e_j,0}$	The compete time when the task k_i generated by SMD j is offloaded to MBS server
t_{i,e_j}^{total}	The complete time of the task k_i generated by SMD j
$D_{\max,k}$	The maximum storage limit of edge server k
d_{i,e_j}	The size (bits) of task k_i generated by SMD j in small area e
N_k	The number of SMDs associated with the base station k
$N_{\max,k}$	The maximum number of SMDs associated with the base station k

BSs. We suppose that the SMDs associated with the same BS are assigned orthogonal frequency spectrum, and the spectrum between SBSs and the MBS is also orthogonal [31]. Subsequently, we solely think about the inter-cell interference between SBSs [32].

In the analysis of the communication model, we only consider the uplink communication process, during which the tasks generated by the SMDs are offloaded to MEC servers. We assume that all SMDs have a fixed transmission power in different communication modes, for example, the power P_M in the SMDs and MBS communication mode and the power P_S in the SMDs and SBSs communication mode.

If the SMD j communicates with the MBS, the signal-to-interference-plus-noise-ratio (SINR) in the uplink communication process will be defined as,

$$r_{j,0} = \frac{P_M G_r}{L_0 d_{j,0}^{\alpha} P_W}, \tag{1}$$

where G_r is the antenna gain at MBS, L_0 is the path loss per reference unit distance, α is the path loss exponent, and P_W is the power of additive white Gaussian noise. Since the bandwidth of MBS is equally allocated to its associated SMDs, the uplink data rate when SMD j communicates with MBS is denoted as,

$$R_{j,0} = \frac{W_{MBS}}{N_0} \log_2 \left(1 + r_{j,0}\right), \tag{2}$$

$$0 < N_0 \leq N_{\max,0}, \tag{3}$$

where W_{MBS} denotes the bandwidth of MBS and (3) means that the number N_0 of SMDs associated with MBS cannot exceed its maximum capacity $N_{max,0}$.

Similarly, if the SMD j is associated with SBS $bs \in \{1, 2, \ldots, m\}$, the SINR is given by,

$$r_{j,bs} = \frac{\frac{P_S}{L_0 d_{j,bs}^{\alpha}}}{P_W + \sum_{j'=1, j\neq j}^{N} \sum_{sbs'=1, bs'\neq bs}^{m} \frac{P_S}{L_0 d_{j',bs'}^{\alpha}}}, \tag{4}$$

where $\sum_{j'=1,j\neq j}^{N} \sum_{sbs'=1,bs'\neq bs} \frac{P_S}{L_0 d_{j',bs'}^{\alpha}}$ is the inter-cell interference among SBSs. The achievable rate of the SMD j associated with the SBS bs will then be,

$$R_{j,bs} = \frac{W_{SBS}}{N_{bs}} \log_2 \left(1 + r_{j,bs}\right), \tag{5}$$

$$0 \leq N_{bs} \leq N_{\max,bs}, \tag{6}$$

where W_{SBS} denotes the bandwidth of MBS and (6) means that the number N_{bs} of SMDs associated with SBS bs cannot exceed its maximum capacity $N_{max,bs}$.

C. COMPUTATION MODEL

In this subsection, we discuss the computation cost of task k on the basis of upload time, wait time in the MEC server and task processing time. Here, we take the first-come-first-served working mechanism of servers into account, which results in the waiting time of tasks on the server. Similarly, $s_0, s_1, \ldots, s_m$ denote the required computation (CPU cycles) of the tasks queuing in serve $_0$, serve $_1$, serve $_2$, $\ldots$, serve $_m$. Let $\{w_0, w_1, \ldots, w_m\}$ denote the computing capacity (HZ) of these servers, respectively.

1) OFFLOADING TO SBS

As shown in FIGURE 3, the SMD j in the small area e_j offloads the $d_{i,e_j,bs}$−bit task k_i to SBS edge server bs for execution. Thus, in this process, the completion time can be written as,

$$t_{i,e_j,bs} = \frac{d_{i,e_j,bs}}{R_{j,bs}} + \frac{s_{bs}}{w_{bs}} + \frac{d_{i,e_j,bs} c_i}{w_{bs}}, \tag{7}$$

where the first term is the upload time, the second term is the wait time and the third term is the execution time in the SBS server bs.

2) OFFLOADING TO MBS

In this case, the completion time $t_{i,e_j,0}$ can be defined as,

$$t_{i,e_j,0} = \frac{d_{i,e_j,0}}{R_{j,0}} + \frac{s_0}{w_0} + \frac{d_{i,e_j,0} c_i}{w_0},$$

$$d_{i,e_j,0} = d_i - \sum_{bs=1}^{m} d_{i,e_j,bs}, \tag{8}$$

$$\sum_{bs=1}^{m} \frac{d_{i,e_j,bs}}{|d_{i,e_j,bs}|} = 1, \tag{9}$$

where (9) indicates that each SMD selects only one SBS for task offloading.

Since each task can be processed at the MBS server and the SBS server concurrently. Thus, the completion time of the task k_i in the small area e_j can be written as,

$$t_{i,e_j}^{total} = \max\left(\sum_{bs=1}^{m} t_{i,e_j,bs}, t_{i,e_j,0}\right)$$

$$= \max\left(\sum_{bs=1}^{m}\left(\frac{d_{i,e_j,bs}}{R_{j,bs}}+\frac{s_{bs}}{w_{bs}}+\frac{d_{i,e_j,bs}c_i}{w_{bs}}\right),\right.$$
$$\left.\frac{d_i-\sum_{bs=1}^{m} d_{i,\theta,bs}}{R_{j,0}}+\frac{s_0}{w_0}+\frac{\left(d_i-\sum_{bs=1}^{m} d_{i,e_j,bs}\right)c_i}{w_0}\right). \tag{10}$$

In the small area-based edge offloading of multi-server multi-SMD and multi-task, we formulate a multi-constraint offloading optimization problem that minimizes the completion time of all tasks. The time-based optimization problem can be defined as,

$$\min_{\left\{d_{i,e_j,bs}\right\}} \quad \sum_{i=1}^{G}\sum_{j=1}^{N} t_{i,e_j}^{total} \tag{11}$$

$$\text{s.t.} \quad \sum_{bs=1}^{m}\frac{d_{i,e_j,bs}}{\left|d_{i,e_j,bs}\right|}=1, \tag{11.a}$$

$$d_i = d_{i,e_j} = \sum_{bs=1}^{m} d_{i,e_j,bs} + d_{i,e_j,0}, \tag{11.b}$$

$$0 \le d_{i,e_j,bs} \le d_i, \tag{11.c}$$

$$\sum_{i=1}^{G}\sum_{j=1}^{N} d_{i,e_j,k} \le D_{\max,k}, \tag{11.d}$$

$$0 \le N_k \le N_{\max,k}, \tag{11.e}$$

$$\forall i \in G,\ k \in \{0,1,2,\ldots,m\},$$
$$j \in \{1,2,\ldots,N\},\ bs \in \{1,2,\ldots,m\}.$$

As for the constraints in the optimization problem, we explain them as follows: constraint (11.a) is essentially an association constraint, that is, which one SBS should the SMD communicate with; constraints (11.a), (11.b), and (11.c) imply that each computation task can be divided into two parts for execution, with one part being offloaded to one SBS server and the other part being offloaded to the MBS server; constraint (11.d) ensures that the task size received by each edge server cannot exceed its maximum storage limit; constraint (11.e) means that the number of SMDs associated with the base station during communication cannot exceed its maximum capacity.

IV. DEEP LEARNING-BASED OPTIMAL OFFLOADING SCHEME

It is quite challenging to optimally address the optimization problem due to the multi-constraints, which essentially implies an association constraint and is usually NP-hard [33]. To make the optimization problem more realistic, we should also consider some dynamic factors in mobile edge computing offloading, such as the changeable state of edge servers and the communication environment in continuous time. Since the above two points, we transform the optimization problem into a deep reinforcement learning-based offloading scheme by the Markov decision approach and design a Deep Deterministic Policy Gradient-based offloading approach to achieve computation offloading.

A. MARKOV DECISION APPROACH

We discrete time into time slots and assume that there will be a batch of task requests within each time slot that can arrive at the corresponding edge servers based on offloading strategy. The length of each time slot is ϑ. Since each edge server has a task waiting queue, the status of the service queue in the previous time slot will affect the completion time of the arrived task in the current time slot. In detail, the task offloading strategy of time period l depends on the server queue status of time period $l-1$ and the communication status of the SMDs in time slot l. Thus, we can transform the time-based optimization problem to the Markov decision process and try to use the Markov decision approach to solve it.

The status of the MEC system with multi-server multi-SMD and multi-task at the beginning of time slot l is represented as $S^l = \{s_0^l, s_1^l, \ldots, s_m^l\}$. The action taken by the control center at the time slot l is denoted as $a^l = \{d_{i,e,bs}^l\}(i \in G, e \in \{1,2,\ldots,E\}, bs \in \{1,2,\ldots,m\})$, where $d_{i,e,bs}^l$ is the data size of the task i offloaded to the SBS server bs in the small area e. We introduce the variable ζ_{bs}^l to analyze the effects caused by the actions to the MEC system states, which represents the amount of computation by SBS server bs in time slot l. The variable is defined as,

$$\zeta_{bs}^l = \min\left(s_{bs}^l + \sum_{i=1}^{G}\sum_{j=1}^{N} d_{i,e_j,bs}c_i,\ \vartheta w_{bs}\right). \tag{12}$$

The state of time slot $l+1$ affected by the above variable and the system state of time slot l is defined as,

$$S^{l+1} = \left\{s_k^l + \sum_{i=1}^{G}\sum_{j=1}^{N} d_{i,e_j,k}^l c_i - \zeta_k^l\right\},$$
$$k \in \{0,1,\ldots,m\}, \tag{13}$$

where $\zeta_{k=0}^l = \min\left(s_0^l + \sum_{i=1}^{G}\sum_{j=1}^{N} d_{i,e_j,0}^l c_i,\ \tau w_0\right)$ is the amount of computation by MBS server 0 in time slot l and $d_{i,e_j,k=0}^l = d_i - \sum_{bs=1}^{m} d_{i,e_j,bs}^l$ is the data size of the task i offloaded to the MBS server 0 in the small area e_j.

When action a^l is taken in state S^l, the gained reward in time slot l is,

$$U^l = \sum_{j=1}^{N}\sum_{i=1}^{G} -t_{i,e_j,l}^{total}, \tag{14}$$

$$t_{i,e_j,l}^{total} = \max\left(\sum_{bs=1}^{m}\frac{d_{i,e}^l j^{bs}}{R_{j,bs}}+\frac{d_{i,e_j,bs}^l c_i}{w_{bs}}+\frac{s_{bs}^l}{w_{bs}},\right.$$
$$\frac{d_i-\sum_{bs=1}^{m} d_{i,e_j,bs}^l}{R_{j,0}}$$
$$\left.+\frac{\left(d_i-\sum_{bs=1}^{m} d_{i,e_j,bs}^l\right)c_i}{w_0}+\frac{s_0^l}{w_0}\right). \tag{15}$$

We assume that there will be a batch of task requests arriving at the MEC server in each time slot, then the utility

of the offloading system in l_{max} time slots can be expressed as,

$$\max_{\psi^*} U = \sum_{l=1}^{l_{\max}} \sum_{j=1}^{N} \sum_{i=1}^{G} -t_{i,e_j,l}^{total}. \tag{16}$$

In the optimization problem, we maximize the utility U of the MEC system by obtaining an optimal strategy Ψ^* composed of offloading patterns for G types of tasks in different small areas.

B. DEEP DETERMINISTIC POLICY GRADIENT-BASED OFFLOADING APPROACH

To obtain the optimal strategy, we start with deep reinforcement learning technology [34].

Algorithm 1 Deep Deterministic Policy Gradient-Based Task Offloading

Initialization:

Initialize mobile edge computing environment, including the location of base station equipped with edge server, SMDs movement trajectory based on small area and task information generated by SMDs;

Randomly initial Critic network $Q\left(S^l, a^l; w\right)$ and Actor $\mu_\theta\left(S^l\right)$ with weights w and θ;

Initialize target network $Q'\left(S^l, a^l; w'\right)$ and $\mu'_{\theta'}\left(S^l\right)$ with weights $w'^{Q'} \leftarrow w^Q$ and $\theta'^{\mu'} \leftarrow \theta^\mu$;

Initialize replay memory pool R;

Iteration:

for each round $t = 1$ **to** T **do**

  Accept initial MEC system state S^0;

  Initialize a random process $\mathcal{N}$ for action exploration;

  for time slots $l = 0, \ldots, l_{\max}$ **do**

    Select action $a^l = \mu_\theta\left(S^l\right) + \mathcal{N}^l$ according to the current MEC system state and exploration noise;

    Combined with mobile edge computing environment, perform action a^l to obtain reward U^l and new MEC system state S^{l+1};

    Store state transition $\left(S^l, a^l, U^l, S^{l+1}\right)$ in pool R;

    Randomly select a minibatch of N transitions $\left(S^i, a^i, U^i, S^{i+1}\right)$ from R as a sample set;

    Set $y_i = U^i + \gamma Q'\left(S^{i+1}, \mu'_{\theta'}\left(S^{i+1}\right); w'\right)$;

    Update the Critic network via minimizing the loss: $L = \frac{1}{N}\sum_i \left(y_i - Q\left(S^i, a^i; w\right)\right)^2$;

    Update the Actor network using the sampled policy gradient:
    $\nabla_{\theta^\mu} J \approx \frac{1}{N}$
    $\sum_{a^l} \mathrm{Q}\left(S^l, a^l; \mathrm{w}\right)\big|_{S^l=S^i, a^l=\mu_\theta(S^i)} \nabla_{\theta^\mu}\mu_\theta\left(S^l\right)\big|_{S^l=S^i}$

    Update the target network:
    $\theta'^{\mu'} = \tau\theta^\mu + (1-\tau)\theta'^{\mu'}$,
    $\mathrm{w}'^{\mathbf{Q}'} = \tau w^Q + (1-\tau)\mathrm{w}'^{\mathbf{Q}'}$;

  end for

end for

The random strategy π can be expressed as,

$$\Pi : S^l \times a^l \to U^l, \tag{17}$$

which means that action a^l is selected from multiple actions under state S^l to execute and get the reward U^l.

The deterministic strategy π can be expressed as,

$$\Pi : S^l \to a^l, \tag{18}$$

which means that only one deterministic action a^l is executed in state S^l.

We first approximate the strategy as a continuous function with parameter θ to solve the optimal strategy using the continuous function optimization method.

The random strategy is changed to

$$\pi_\theta\left(a^l \mid S^l\right) = P\left(a^l \mid S^l; \theta\right), \tag{19}$$

which means that the output action obeys a probability distribution in the given the state and parameters.

The deterministic strategy is changed to

$$a^l = \mu_\theta\left(S^l\right), \tag{20}$$

which means that the output action is deterministic in the given the state and parameters.

The gradient ascent method is used to solve the optimal strategy, and the calculation formula obtained for the random strategy is,

$$\nabla_\theta J\left(\pi_\theta\right) = E_{S^l\sim\rho^\pi, a^l\sim\pi_\theta}\left[V_\theta \log \pi_\theta\left(a^l \mid S^l\right) Q_\pi\left(S^l, a^l\right)\right]. \tag{21}$$

And the calculation formula obtained for the deterministic strategy is,

$$\nabla_\theta J\left(\mu_\theta\right) = E_{S^l\sim\rho^\mu}\left[\nabla_\theta \mu_\theta\left(S^l\right) \nabla_a Q_\mu\left(S^l, a^l\right)\big|_{a^l=\mu_\theta(S^l)}\right]. \tag{22}$$

Since the action of the offloading system belongs to the high dimensional continuous value, we hope to obtain a deterministic optimization strategy μ^*, $\psi^* = \mu^*$.

In the Deterministic Policy Gradient (DPG) algorithm, the Actor network uses the approximate deterministic strategy μ_θ to obtain deterministic action a^l according to state S^l and gets reward U^l as well as the new state S^{l+1}, the Critic network uses the approximate action-value function $Q\left(S^l, a^l; w\right)$ to evaluate the performance of the Actor network and guide the Actor network.

Then, θ of the Actor network is updating according to

$$\theta^{l+1} = \theta^l + \alpha_\theta \nabla_\theta \mu_\theta\left(S^l\right) \nabla_\alpha Q^w\left(S^l, a^l\right)\big|_{a^l=\mu_\theta(s^l)}, \tag{23}$$

w of the Critic network is updating according to

$$\delta^l = U^l + \gamma Q^w\left(S^{l+1}, \mu_\theta\left(S^{l+1}\right)\right) - Q^w\left(S^l, a^l\right), \tag{24}$$

$$w^{l+1} = w^l + \alpha_w \delta^l \nabla_w Q^w \left(S^l, a^l\right), \tag{25}$$

where γ is the attenuation factor.

The Deep Deterministic Policy Gradient (DDPG) algorithm adds experience pool and target network to DPG algorithm, which breaks the correlation between data and increase the stability of the algorithm.

In the target Actor network $\mu'_{\theta'}\left(S^l\right)$, θ' is updated according to

$$\theta'^{\mu'} = \tau\theta^{\mu} + (1-\tau)\theta'^{\mu'}, \tag{26}$$

where τ is the soft update factor. In the target Critic network $Q'\left(S^l, a^l; w'\right)$, w' is updated according to

$$w'^{Q'} = \tau w^{Q} + (1-\tau)w'^{Q'}. \tag{27}$$

Next, the experience replay technology stores the past system state transition experience in the replay memory pool, and randomly samples from the pool for training during the learning process to improve learning performance. The system state transition experience in the replay memory pool is composed of observed state transitions and gained reward caused by actions in each time slot. The system state transition experience obtained at time slot l is denoted as $\left(S^l, a^l, U^l, S^{l+1}\right)$. In order to train the Critic network parameters, a batch of stored experience is randomly selected from the replay memory pool as samples. The purpose of the training is to minimize the difference between $Q\left(S^l, a^l; w\right)$ and $Q'\left(S^l, a^l; w'\right)$. To represent the difference, we define a loss function L, which is,

$$L = \frac{1}{N}\sum_i \left(y_i - Q\left(S^i, a^i; w\right)\right)^2, \tag{28}$$

$$y_i = U^i + \gamma Q'\left(S^{i+1}, \mu'_{\theta'}\left(S^{i+1}\right); w'\right), \tag{29}$$

where N is the number of samples drawn from the experience pool. The Offload-DDPG consists of Actor network, Critic network, target Actor network, target Critic network and repay memory pool in MEC environment, as shown in FIGURE 4. The Actor network is responsible for iteratively updating the network parameters θ, choosing the current action according to the current state, and interacting with the MEC environment to generate the next state and reward. The target Actor network is responsible for selecting the optimal next action based on the next state sampled in the experiential playback pool. The Critic network is responsible for iteratively updating parameter w, and calculating the current value $Q\left(S^l, a^l; w\right)$. The target Critic network is mainly responsible for calculating the value $Q'\left(S^l, a^l; w'\right)$. The proposed Deep Deterministic Policy Gradient-based task offloading (Offload-DDPG) approach is shown in Algorithm 1.

We combine constraint conditions (11.d), (11.e) and selection probability to determine the offloading object in the DDPG algorithm code. As shown in FIGURE 5, under the condition of satisfying the constraints, we select the service object of each task in each small region according to the state of the server. In the Actor network, the task i in small region

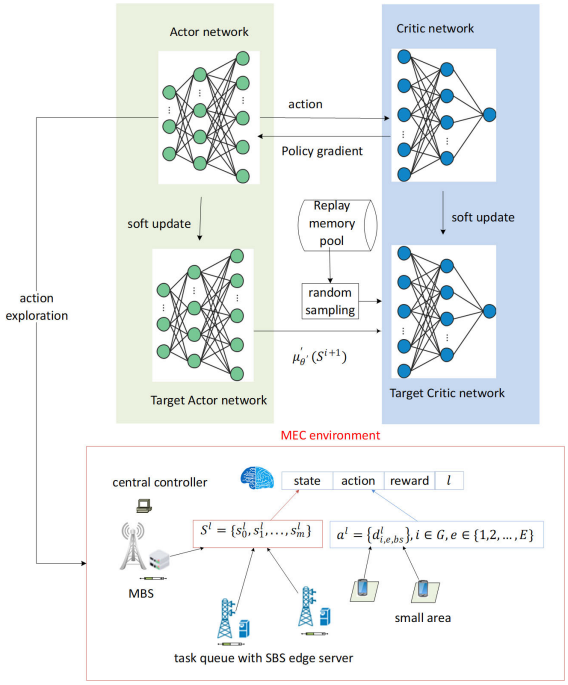

FIGURE 4. The structure of offload-DDPG in MEC environment.

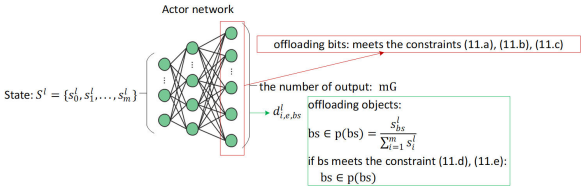

FIGURE 5. The offloading bits and objects in the Actor network.

e only selects one SBS service object BS, which satisfies constraint (11.a). And We add constraints to the output value of the Actor network to satisfy the constraint (11.b), (11.c).

V. PERFORMANCE EVOLUTION

In this section, we verify the performance of the proposed task offloading approach based on real trajectory data [35]–[37] in a mobile edge network topology. We design three groups of experiments, the first group is for verifying the training efficiency with different learning parameters on Offload-DDPG, the second is for the optimization performance with different strategies in multi-server multi-SMD and multi-task scenario, and the third is for observe the optimization performance of individual SMD with different strategies.

A. SIMULATION SETTINGS

We consider a scenario that, a network topology of 10 km × 10 km consists of one Macro Base Station (MBS), 100 Small Base stations (SBSs) and trajectories of SMDs. SBSs are uniformly deployed in the MBS signal coverage area located at the center of the MEC network. The area is divided into 10,000 small areas (100 m × 100 m) on average.

The simulation parameters are defined in Table 2. The parameters of Deep Deterministic Policy Gradient-Based Task Offloading approach (Offload-DDPG) are given in Table 3.

TABLE 2. Simulation Parameters.

Symbol	Quantity	Value
W_{MBS}	channel bandwidth with MBS	$20MHz$
W_{SBS}	channel bandwidth with SBS	$10MHz$
P_M, P_S	transmission power of mobile devices	$33dBm$
G_r	the antenna gain at the BS	$17dBi$
P_W	the power of additive white Gaussian noise	10^{-11} mW
L_0	the path loss at a reference unit distance	$47.86dB1$
α	the path loss exponent	2.75
w_{bs}	the computing capacities of SBSs servers bs, $bs \in \{1, \ldots, m\}$	$50GHz$
w_0	the computing capacities of MBS server	$500GHz$
d_i	the data size of task k_i	[200,500]KB
c_i	the CPU cycles of task k_i	[50,100]$cycles/bit$

TABLE 3. The Parameters of Offload-DDPG.

Parameters	Value
Learning rate for actor δ_a	0.001
Learning rate for critic δ_c	0.002
Discount rate γ	0.9
Replay memory size R	10000
Soft update factor τ	0.01
Batch size	32

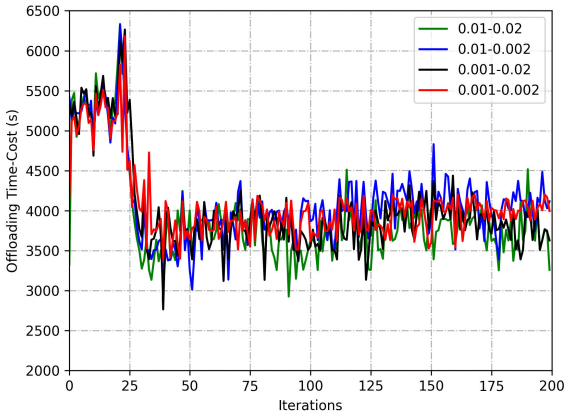

FIGURE 6. The convergence effect with different learning rates (δ_a, δ_c).

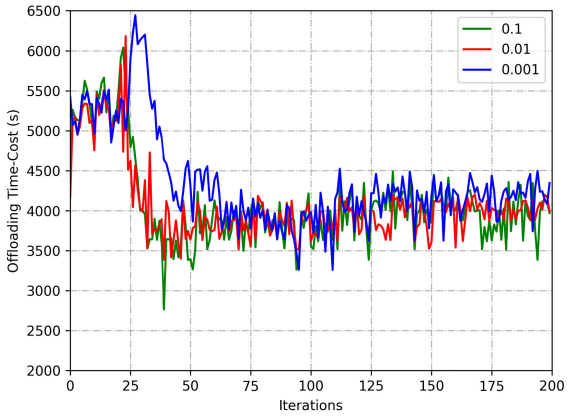

FIGURE 7. The convergence effect with different soft update factor τ.

B. SIMULATION ANALYSIS

1) TRAINING EFFICIENCY WITH DIFFERENT LEARNING PARAMETERS ON OFFLOAD-DDPG

In this subsection, we verify the training efficiency of the proposed Offload-DDPG approach with various learning parameters. We take the offloading time-cost ($|U|$) as the evaluation criterion, which is the sum of the completion time of tasks generated by 182 SMDs in 500 time slots.

FIGURE 6 plots the convergence effect of the proposed Offload-DDPG with different learning rates. Specifically, δ_a is the learning rate of the Actor network in the Offload-DDPG approach and δ_c is the learning rate of the Critic network in the Offload-DDPG approach. We find that for different learning rates, they all have a fast convergence rate and generally converge within 50 iterations. We find that when $\delta_a = 0.001$ and $\delta_c = 0.002$, the proposed Offload-DDPG has a relatively small fluctuation range in the convergence stage, which is more suitable for the quality of service of all SMDs.

FIGURE 7 plots the convergence effect of the proposed Offload-DDPG with different soft update factor τ. It is clear that the Offload-DDPG has a relatively longer convergence iteration when $\tau = 0.001$. In the convergence stage, the soft update factor of $\tau = 0.01$ has a relatively smaller fluctuation range than $\tau = 0.1$. Therefore, Offload-DDPG is cost efficient in terms of computation offloading.

We can observe that the offloading time-cost increase and later decline in FIGURE 6 and FIGURE 7. The offloading strategy determines the offloading time-cost. Through the offloading strategy $\left\{d_{i,e,bs}\right\}, i \in G, bs \in \{1, 2, \ldots, m\}$, we can get the offloading object bs and offloading bits$d_{i,e,bs}$ of task k_i in small area e. Thus, the choice of offloading object and the determination of offloading bits of tasks together affect the time-cost. In the training of the proposed DDPG-offloading algorithm, the offloading bits of actions $a^l = \left\{d^l_{i,e,bs}\right\}, i \in G, bs \in \{1, 2, \ldots, m\}$ is determined mainly through the Actor network. However, the selection of offloading object of actions is based on the probability determined by the task queue state of the edge servers at the time slot l, as shown in FIGURE 5. At the beginning of an iteration, a low offloading bits of Actor network affects the server state in the next time slot, resulting in a deficient offloading object, thus increasing the offloading time-cost. Similarly, when the offloading bits of Offload-DDOG

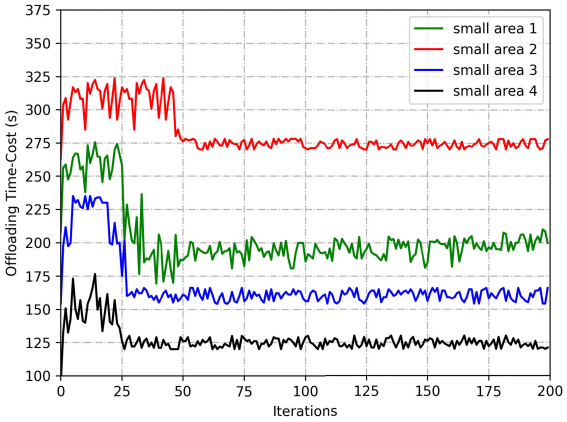

FIGURE 8. The convergence effect with different small areas.

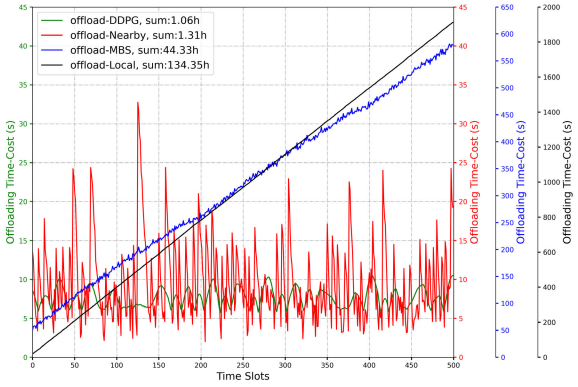

FIGURE 9. The Offloading Time-Cost with different strategies.

algorithm is better, the selection of offloading objects will be better, so the offloading time-cost decline.

FIGURE 8 plots the convergence effect of the proposed Offload-DDPG with different small areas. The actions of different small areas have conflict interests with the MEC system utility. We randomly select four small areas to evaluate the proposed DDPG-based offloading method. Taking the total time cost of all tasks generated in each small area as the evaluation criterion, we find that the Offload-DDPG reaches convergence within 50 iterations and different small areas have different offloading time costs. The reason for the different time cost is that the total scale of tasks occurs in each small area in 500 time slots is different. Therefore, the proposed Offload-DDPG method is feasible with multiple small areas.

2) OPTIMIZATION PERFORMANCE WITH DIFFERENT STRATEGIES IN MULTI-SERVER MULTI-SMD AND MULTI-TASK SCENARIO:

For evaluating the optimization performance of the proposed Offload-DDPG approach, We compare it with the following traditional strategies.

Offload-MBS: During each time slot, the tasks generated by the SMDs in the trajectory information are offloaded to the MBS server for execution.

Offload-Nearby: During each time slot, the tasks generated by the SMDs in the trajectory information are offloaded to the edge servers closest to the corresponding SMD.

Offload-Local: During each time slot, the tasks generated by the SMDs in the trajectory information are executed on local SMDs without resorting to the edge servers.

We assume that there will be a batch of task requests arriving at the MEC servers in each time slot and each mobile device will generate a task request in each time slot. We compare the performance of different strategies with the offloading time-cost ($|U|$) in each time slot. The offloading time-cost represents the sum of time spent by all devices to complete their respective tasks in a time slot.

FIGURE 9 shows that the offloading time-cost and the offloading time-cost fluctuation range using the Offload-DDPG approach was lower than the other three strategies for most of the time slots. The Offload-DDPG approach adopts the parallel task offloading model to ensure the completion of tasks with ultra-low latency. This approach considers the status information of the edge servers and ensures that the edge server is fully utilized to reduce the waiting time for tasks on the server. Moreover, this approach adopts the small-area based offloading scheme which is more suitable for the mobile characteristics of the devices.

We observe that the time-cost of the Offload-MBS strategy is increasing from FIGURE 9. The MBS generates an iterative process, in which the MBS server obtains a batch of new task requests before it has not finished processing the tasks in the previous time slot. This iterative process leads to a continuous increase in the waiting time of each batch of tasks on the MBS server, which in turn leads to a continuous increasing in the time-cost.

In order to explain the experimental phenomenon more clearly, we define a Cumulative Process and the continuous increase of task processing time caused by this process is called the Cumulative Phenomenon. The Cumulative Process refers to the iterative process in which the server has not finished processing the tasks in the previous time slot, and will obtain a batch of new task requests in the next time slot.

We observe that the Offload-Local strategy's time-cost is increasing from FIGURE 9, which is due to the Cumulative Process of SMDs.

We calculate the sum of time-cost in FIGURE 9 and conclude that Offload-DDPG improves the long-term performance by at least 19% compared to other traditional strategies. Therefore, the proposed Offload-DDPG approach has better performance in ultra-low latency and efficient usage of servers than traditional strategies.

3) OPTIMIZATION PERFORMANCE OF INDIVIDUAL SMART MOBILE DEVICES WITH DIFFERENT TRAJECTORIES

We select three individual smart mobile devices from all SMDs to compare the impact of different Trajectories on the

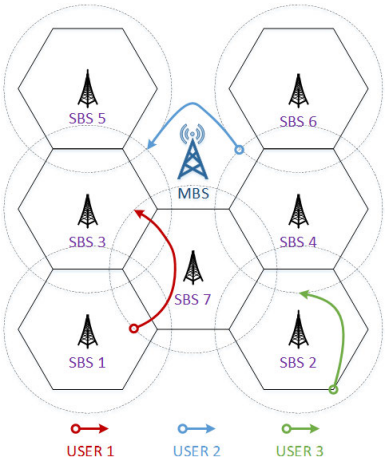

FIGURE 10. Diagram of individual user trajectory.

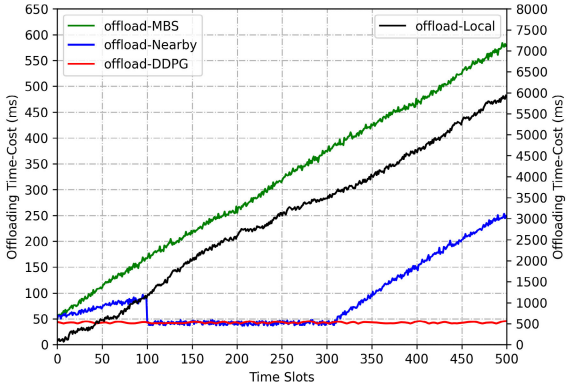

FIGURE 11. The offloading time-cost of user 1 with different strategies.

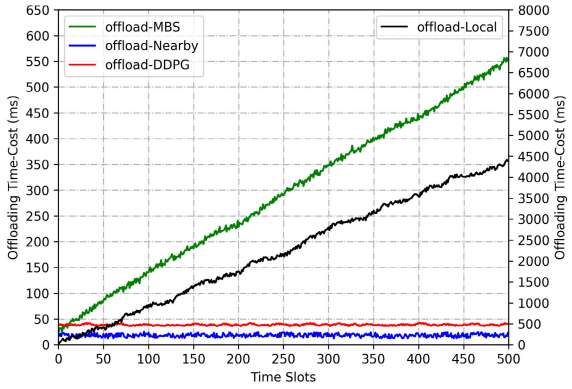

FIGURE 12. The offloading time-cost of user 2 with different strategies.

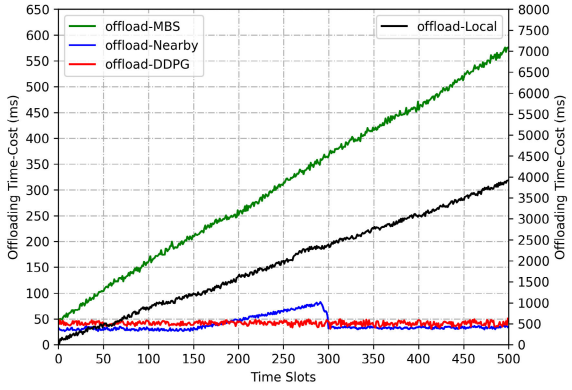

FIGURE 13. The offloading time-cost of user 3 with different strategies.

offloading time-cost. FIGURE 10 shows the diagram of three individual user trajectories.

Across-SBSs: The SMD moves across in the cellulars of SBSs, i.e., user 1.

In-MBS: The SMD moves in the cellular of the MBS, i.e., user2.

In-SBS: The SMD moves in the cellular of one SBS, i.e., user 3.

FIGURE 11, FIGURE 12 and FIGURE 13 show that the offloading time-cost of using the Offload-MBS strategy or the Offload-Local strategy is increasing under different moving trajectories, that is, Cumulative Phenomenon. The reason for this phenomenon is that the MBS server under Offload-MBS and SMDs under Offload-Local have the Cumulative Process in 500-time slots. Therefore, the Offload-MBS strategy and the Offload-Local strategy cannot satisfy the user experience in the multi-SMD scenario.

It can be seen from FIGURE 11, FIGURE 12 and FIGURE 13, the offloading time-cost of the Offload-DDPG approach has the smallest fluctuation range and is always less than 50ms under different moving trajectories. The Offload-DDPG strategy adopts the small area-based offloading scheme in the parallel offloading model to ensure ultra-low latency, and avoids the impact of a dynamic offloading environment on the time-cost. Therefore, the Offload-DDPG approach can be more adapted to the frequent mobility of SMDs and satisfy the user experience in the multi-SMD scenario.

The moving trajectory affects the offloading time-cost of the Offload-Nearby strategy, as shown in FIGURE 11, FIGURE 12 and FIGURE 13, so this strategy cannot satisfy the user experience in the multi-SMD scenario. The detail is as follows,

Across-SBSs: FIGURE 11 shows the offloading time-cost of user 1 in 500 time slots.

When the Offload-Nearby strategy is adopted, we can find that the task completion time generated by user 1 in 0-100 time slots and 200-500 time slots is increasing. The SMDs distributed near the nearest base station of user 1 are more densely during these time slots, which causes the nearest server selected by user 1 to be in the Cumulative Process. In 100-300 time slots, the nearest server selected by user 1 has a smaller task request scale and no Cumulative Process. Therefore, in 100-300 time slots, the task completion time is lower and balanced.

In-MBS: FIGURE 12 shows that the offloading time-cost of user 2 in 500 time slots.

When the Offload-Nearby strategy is adopted, the task completion time is the lowest. Because the nearest edge server selected by user 2 only receives task requests from user 2 within 500 time slots, there is no waiting delay.

In-SBS: FIGURE 13 shows the offloading time-cost of user 3 in 500 time slots.

When the Offload-Nearby strategy is adopted, user 3 has a Cumulative Phenomenon in the 150-300 slots. The nearest server selected by user 3 received too many task requests during these time slots, and a Cumulative Process occurs, that is, the previous task of user 3 has not been processed yet, and the next task arrives at the nearest server.

The above simulation results prove that the Offload-DDPG approach can better adapt to multi-server multi-SMD and multi-task scenarios, which achieve the ultra-low latency of tasks and the efficient usage of servers, and also can meet the frequent mobility of SMDs. Therefore, the proposed Offload-DDPG approach has better performance in ultra-low latency, efficient usage of servers, and frequent mobility of SMDs than traditional Strategies.

VI. CONCLUSION

This paper focuses on researching the task offloading in a dense distributed cellular network with multiple MEC servers, multiple SMDs, and multiple tasks. Considering both MEC servers and SMDs mobility, we propose the small area-based parallel task offloading model to achieve ultra-low latency requirements. In this model, we formulate the task offloading optimization problem for minimizing the completion time of all computation tasks. To solve this problem, we transform the optimization problem into a computation offloading scheme based on deep reinforcement learning through the Markov decision approach and propose a Deep Deterministic Policy Gradient-based offloading (Offload-DDPG) approach to solve this optimization problem. The dynamic communication environment and the variable server status information over continuous time slots are considered in the process of solving the problem. We can get the offloading object and offloading bits of tasks in each small area through the offloading strategy. Simulation results prove that our proposed Offload-DDPG approach improves long-term performance by at least 19% in ultra-low latency.

In the future, our work will focus on the collaboration between SBS in order to further make full use of MEC servers in task offloading. Besides, the deployment of services on MEC servers should also be considered in task offloading.

REFERENCES

[1] M. Gao, R. Shen, J. Li, S. Yan, Y. Li, J. Shi, Z. Han, and L. Zhuo, "Computation offloading with instantaneous load billing for mobile edge computing," *IEEE Trans. Serv. Comput.*, early access, May 25, 2020, doi: 10.1109/TSC.2020.2996764.

[2] M. Gao, J. Li, D. N. K. Jayakody, H. Chen, Y. Li, and J. Shi, "A super base station architecture for future ultra-dense cellular networks: Toward low latency and high energy efficiency," *IEEE Commun. Mag.*, vol. 56, no. 6, pp. 35–41, Jun. 2018.

[3] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[4] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, pp. 54074–54084, 2020.

[5] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, early access, May 12, 2020, doi: 10.1109/TMC.2020.2994232.

[6] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[7] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.

[8] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, early access, Jun. 10, 2020, doi: 10.1109/TII.2020.3001355.

[9] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12313–12325, Dec. 2018.

[10] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[11] F. Wang, H. Xing, and J. Xu, "Real-time resource allocation for wireless powered multiuser mobile edge computing with energy and task causality," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 7140–7155, Nov. 2020.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[13] M. Deng, H. Tian, and X. Lyu, "Adaptive sequential offloading game for multi-cell mobile edge computing," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1–5.

[14] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[15] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[16] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[17] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[18] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1524–1537, Mar. 2020.

[19] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[20] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Apr. 2013.

[21] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[22] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[23] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[24] L. Cui, C. Xu, S. Yang, J. Z. Huang, J. Li, X. Wang, Z. Ming, and N. Lu, "Joint optimization of energy consumption and latency in mobile edge computing for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4791–4803, Jun. 2019.

[25] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 726–738, Sep. 2019.

[26] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "Robust mobile crowd sensing: When deep learning meets edge computing," *IEEE Netw.*, vol. 32, no. 4, pp. 54–60, Jul. 2018.
[27] B. Gu, X. Yang, Z. Lin, W. Hu, M. Alazab, and R. Kharel, "Multi-agent actor-critic network-based incentive mechanism for mobile crowdsensing in industrial systems," *IEEE Trans. Ind. Informat.*, early access, Sep. 21, 2020, doi: 10.1109/TII.2020.3024611.
[28] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, "Reinforcement learning-based mobile offloading for edge computing against jamming and interference," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6114–6126, Oct. 2020.
[29] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12175–12186, Oct. 2020.
[30] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Trans. Ind. Informat.*, early access, Aug. 13, 2020, doi: 10.1109/TII.2020.3016320.
[31] W. Lu, Q. Fan, Z. Li, and H. Lu, "Power control based time-domain intercell interference coordination scheme in DSCNs," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.
[32] T. Zahir, K. Arshad, A. Nakata, and K. Moessner, "Interference management in femtocells," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 293–311, 1st Quart., 2013.
[33] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1990.
[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 33–529, 2019.
[35] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 791–800.
[36] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on GPS data," in *Proc. 10th Int. Conf. Ubiquitous Comput.*, 2008, pp. 312–321.
[37] Y. Zheng, X. Xie, and W.-Y. Ma, "GeoLife: A collaborative social networking service among user, location and trajectory," *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–40, Jun. 2010.

HONGXIA ZHANG received the bachelor's and master's degrees from the College of Computer Science and Technology, China University of Petroleum (East China), in 2003 and 2006, respectively, and the Ph.D. degree from the State Key Laboratory of Networking and Switching, Beijing University of Posts and Telecommunications, in 2013. She is currently an Associate Professor with the College of Computer Science and Technology, China University of Petroleum (East China). Her research interests include cloud computing, mobile edge computing, and artificial intelligence for networking.

YONGJIN YANG is currently pursuing the degree with the College of Computer Science and Technology, China University of Petroleum (East China). His research interests include mobile edge computing, machine learning, deep reinforcement learning, and natural language processing.

XINGZHE HUANG is currently a Graduate Student with the College of Computer Science and Technology, China University of Petroleum (East China). His research interests include artificial intelligence and natural language processing.

CHAO FANG (Senior Member, IEEE) received the B.S. degree in information engineering from the Wuhan University of Technology, Wuhan, China, in 2009, and the Ph.D. degree from the State Key Laboratory of Networking and Switching Technology in Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2015. He joined the Beijing University of Technology, in 2016, where he is currently an Associate Professor. From August 2013 to August 2014, he had been visiting Carleton University, Ottawa, ON, Canada, as a Visiting Scholar. Moreover, he is also a Visiting Scholar with the University of Technology Sydney, The Hong Kong Polytechnic University, and Kyoto University. His current research interests include future network architecture design, information-centric networking (ICN), software-defined networking (SDN), big data for networking, mobile edge computing, resource management, and content delivery. He served as the Session Chair for IEEE ICC NGN'2015 and the Poster Co-Chair for IEEE HotICN'2018. He also served as a member of the Technical Program Committee for IEEE ISCIT'2017, ISCIT'2018, and CMST'2019.

PEIYING ZHANG received the Ph.D. degree from the School of Information and Communication Engineering, University of Beijing University of Posts and Telecommunications, in 2019. He is currently an Associate Professor with the College of Computer Science and Technology, China University of Petroleum (East China). Since 2016, he has been publishing multiple IEEE/ACM transactions/journal/magazine articles, such as IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, *IEEE Network*, IEEE ACCESS, IEEE INTERNET OF THINGS JOURNAL, *ACM TALLIP*, *Computer Communications*, and *IEEE Communications Magazine*. His research interests include semantic computing, future internet architecture, network virtualization, and artificial intelligence for networking. He served as a member of the Technical Program Committee for ISCIT'2016, ISCIT'2017, ISCIT'2018, ISCIT'2019, GLOBECOM'2019, COMNETSAT'2020, SOFT-IoT'2021, IWCMC-Satellite'2019, and IWCMC-Satellite'2020.