Deep-Deterministic-Policy-Gradient-Based Task Offloading With Optimized *K*-Means in Edge-Computing-Enabled IoMT Cyber-Physical Systems

Chenyi Yang, Xiaolong Xu, *Senior Member, IEEE*, Muhammad Bilal, *Senior Member, IEEE*, Yiping Wen, and Tao Huang

***Abstract*—In recent years, the Internet of Medical Things (IoMT) is expanding its value as a key enabling technology for smart medical cyber-physical systems. In order to overcome the constraints of constrained local resources, smart medical equipment (ME) in IoMT cyber-physical systems offload data to edge or cloud servers for processing. However, due to the limited edge resources and huge time delay caused by offloading data to the cloud, the lack of a reasonable task unloading scheme will lead to the unbearable time delay and energy consumption of the IoMT system, resulting in uneven workloads among edge servers and threatening the security of medical data. To cope with these challenges, a deep deterministic policy gradient (DDPG)-based task-offloading method assisted by clustering is proposed. We first improved the initialization process of *K*-means, and based on this, designed an optimized *K*-means algorithm to carry out scientific and reasonable clustering of MEs according to their quality-of-service requirements. Then, DDPG is employed to obtain an optimal task-offloading scheme to minimize average latency and total energy consumption of IoMT and to ensure load balancing among edge servers. Finally, experimental results justify the scientific nature of optimized *K*-means and the superiority of DDPG in reducing the system overhead of IoMT. Compared with benchmark algorithms, DDPG reduces average time delay and total energy consumption by at least 16.9% and 12.3%, respectively.**

***Index Terms*—Clustering, deep reinforcement learning, edge computing, Internet of Medical Things (IoMT), task offloading.**

I. Introduction

IN RECENT years, despite the continuous development of medical and health information construction and the increase of medical resources, the problem of difficult and expensive medical treatment has not been solved essentially because the information isolation between hospitals and patients is not familiar with medical services [1]. What is worse, hospitals collect and store massive amounts of patients' private information, which puts people's privacy security under threat. Fortunately, due to the symbiotic growth of machine learning and artificial intelligence, the value of the Internet of Medical Things (IoMT) is increasing and people are entering the era of intelligent medical care [2], [3]. In the IoMT system, large numbers of medical devices deployed close to users at the edge of the network can collect massive data from people and their surroundings all the time with the aid of sensors deployed on them [4], [5]. Through the efficient and rapid processing of this massive amount of data, IoMT can provide high-quality remote health care, pathological diagnosis, long-term care, and disease prevention services for people. This allows people to easily access timely and inexpensive medical care [6], [7].

To realize the efficient processing of massive complex data requires a large amount of computing resources and energy [8], [9]. Medical devices with limited local computing resources and battery power have to transmit part of the original data to the cloud for processing to ensure the completion of tasks [10], [11]. However, the cloud processes a large amount of patients' health information in a centralized computing way, such as electronic medical records and medical images, which is difficult to protect patients' privacy and medical data from leakage and reduce the risk of patient privacy data being tampered with. Besides, although the rich computing resources in the cloud make up for the shortage of local resources in medical equipment (ME), the cloud server is far away from the edge of the network, which will cause unbearable time delays in data transmission and ultimately leads to ME not completing tasks in time [12], [13]. Relying

Manuscript received 12 December 2022; revised 31 May 2023 and 23 July 2023; accepted 28 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62372242, Grant 62177014, and Grant 92267104, in part by the Research Foundation of Hunan Provincial Education Department of China under Grant 20B222, and in part by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant 21KJB520001. *(Corresponding authors: Xiaolong Xu; Muhammad Bilal.)*

Chenyi Yang is with the School of Software, Nanjing University of Information Science and Technology, Nanjing 211544, China (e-mail: 201983290234@nuist.edu.cn).

Xiaolong Xu is with the School of Software, Nanjing University of Information Science and Technology, Nanjing 211544, China, and also with the Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology, Nanjing 211544, China (e-mail: xlxu@ieee.org).

Muhammad Bilal is with the School of Computing and Communications, Lancaster University, LA1 4WA Lancaster, U.K. (e-mail: m.bilal@ieee.org).

Yiping Wen is with the Hunan Key Laboratory for Service Computing and Novel Software Technology, Hunan University of Science and Technology, XiangTan 411201, China (e-mail: ypwen81@gmail.com).

Tao Huang is with the School of Computer Science and Technology, Silicon Lake College, Suzhou 215332, China (e-mail: nuisthuangtao@163.com).

Digital Object Identifier 10.1109/JSYST.2023.3311454

solely on cloud computing will impose incalculable system costs on IoMT.

Edge computing, as a complement and optimization to cloud computing, can provide Big Data processing services at the edge of the network in close proximity to users [14], [15], which obviously reduces the delay and cost of sending data to the cloud for processing [16], [17]. The application of edge computing enables MEs to offload data to edge servers for calculation to reduce data processing and transmission delay and ensure the timeliness of task completion [18], [19]. In addition, in the cloud-edge collaborative IoMT environment, MEs take advantage of cloud and edge resources to complete data processing [20], [21], which significantly alleviates the limitation of local battery energy shortage of MEs [22]. Although the edge server can provide considerable computational storage resources, it is worth noting that if MEs blindly conduct task offloading, lacking a scientific and reasonable task-offloading scheme, some edge servers will encounter the threat of overload or some edge servers will not receive tasks for a long time and remain idle [23], resulting in a waste of resources [24].

Due to the large action space of task offloading, it is extremely difficult to obtain the optimal task unloading scheme, which minimizes the system loss and ensures the load balance of edge servers. In addition, the tasks accomplished by different MEs vary greatly, with different delay requirements and reliability requirements. When these heterogeneous tasks are offloaded at the same time, some tasks that require high quality of service occupy a large amount of resources, delay the completion of other tasks, and even cause computing resource overload [25].

In order to solve the above problems, we first utilize the clustering algorithm to cluster MEs according to service quality requirements of tasks aiming to ensure that relatively similar tasks can participate in offloading together. The task offloading of MEs with similar service quality requirements greatly reduces the computational complexity of subsequent task offloading, facilitates the preallocation of edge resources, and effectively reduces the risk of computing resource overload. Then, a deep reinforcement learning method is employed to get an optimal task-offloading scheme [26]. DRL is the combination of deep learning and reinforcement learning. It makes use of the powerful representation ability of neural networks to fit strategies [27]. Therefore, DRL is very good at solving problems with excessive state and action space or continuous action space. The main contributions of this article are summarized as follows.

1) An ME-edge network framework that enables MEs to offload their original data to any edge server or cloud for processing to make up for the lack of local resources is proposed.
2) An optimized *K*-means algorithm is designed to cluster MEs according to their respective quality-of-service requirement.
3) A task-offloading method driven by deep deterministic policy gradient [28] is proposed to obtain an optimal task-offloading scheme for IoMT.
4) Comparative experiments prove the scientificity of optimized *K*-means in clustering and the superiority of DDPG in reducing system delay and energy consumption while ensuring load balancing among edge servers.

The rest of the article is organized as follows. Section II summarizes the related work. In Section III, the system model and problem formulation are described. Section IV introduces our proposed optimized *K*-means and DDPG-based task-offloading method. In Section V, the performance of proposed methods is evaluated through extensive comparative experiments. Finally, Section VI concludes this article.

II. Related Work

In the context of 5G, with the development of the Medical Internet of things and artificial intelligence, more and more edge medical devices close to people can serve as tools for health detection and epidemic prevention. However, edge medical devices lack sufficient energy supply and computing resources due to its own hardware conditions and other reasons, so they have to offload part of the data to edge servers or cloud for processing. In addition, due to the increasing number of medical devices and the increasing richness of medical applications, the amount of task data and task complexity are also rapidly increasing, which leads to the limited resources provided by edge servers. In other words, if there is no scientific and reasonable task-offloading scheme, the whole IoMT will suffer huge time delay and energy consumption, and even face the risk of collapse, and the utilization rate of edge resources will be greatly reduced.

In order to cope with the above problems, some researchers have carried out the task-offloading work based on DRL. Lin et al. modeled task offloading and resource allocation as a Markov decision problem and designed a novel collective reinforcement learning algorithm to allocation resources. Their approach significantly reduces system energy consumption [29]. To save energy while reducing resource-sharing costs, Mekala et al. [30] proposed a two-step service offloading method based on deep reinforcement learning to reduce the cost of edge servers. Shi et al. [31] developed a DRL algorithm based on soft actor–critic aiming to reduce the average latency of processing tasks in Internet of vehicles. Their method greatly improves the operational efficiency of heterogeneous vehicular networks. To deal with the problem that it is difficult for edge devices to make decisions of task offloading by a decentralized method, Tang et al. [32] proposed a distributed algorithm driven by model-free DRL. Ning et al. divided task unloading into two stages: 1) traffic redirection and 2) offloading decision, and proposed a task-offloading strategy based on deep reinforcement learning. Experimental data proved that their method could reduce the average energy consumption by about 60% compared with the baseline method [33].

The above studies only take energy consumption reduction or time delay reduction as the optimization objective and do not optimize these two indicators simultaneously. There are some research works on task-offloading algorithms based on DRL, which can reduce time delay and save energy. In order to solve problems of the energy shortage of equipment and lack of computing resources in IoMT, Yadav et al. [34] proposed a task-offloading method based on reinforcement learning to reduce time delay and energy consumption. Math et al. [35] designed a task-offloading method based on DRL, taking the weighted sum of time delay, energy consumption, and the cost of acquiring

TABLE I
NOTATION AND INTERPRETATIONS

Notations	Interpretations
M	Number of medical devices.
N	Number of edge servers.
K	Number of clusters.
H	Channel matrix.
S_e	Signal energy.
N_d	Noise power spectral density.
ds_i	Total amount of data to be processed of ME_i.
θ_i	Percentage of data offloaded from ME_i to the cloud.
η_i	Percentage of data offloaded from ME_i to the edge.
p_i,p_i^{edge}	Computing capability of ME_i and E_i.
w_i^{local},w_i^{edge}	Energy power of ME_i and E_i.

cached content as the optimization objective. Their method can not only reduce the system overhead but also ensure the reasonable allocation of edge resources. Chen et al. [36] aimed to investigate a dynamic computation offloading method and they designed a cooperative multiagent DRL to obtain decentralized offloading strategies to minimize power consumption and time delay. He et al. proposed PS-DDPG, a task-offloading algorithm using the priority empirical replay mechanism and the random weight average mechanism based on DDPG. Experiments show that the proposed method has superior stability and convergence compared with the existing work and has excellent performance in energy saving and time delay reduction [37].

However, existing DRL-based task-offloading algorithms focus on reducing system loss or improving task completion rate but ignore the load balancing problem of edge resources. Load balancing among edge servers can effectively prevent idle or overloaded computing resources when the resources of edge servers are insufficient. Our research aims to develop a DRL-based task-offloading algorithm that enables to minimize time delay and energy consumption of IoMT systems while ensuring load balancing among edge servers.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, an ME-edge network framework is proposed where each ME possesses very limited computing resources and enables to offload some of the data to edge servers or cloud. In the system, every single ME involved in task offloading has only one task pending at a certain time. In order to ensure the normal operation of IoMT, we have to design a task-offloading method to minimize the average time delay and total energy consumption of the system and ensure the load balance of all edge servers. Table I shows the key notation in this article and corresponding interpretations.

A. ME-Edge Network Framework and Communication Model

In the proposed ME-edge network framework, due to the insufficient local computing resources, it is difficult to complete tasks in a limited time, which is fatal to those MEs in IoMT that require high timeliness. Thus, each ME needs to offload part of the data to edge servers or the cloud for processing. N edge servers (E_1, $E_2, E_3, \ldots, E_N$) with different computing capabilities are deployed in this system. There are M different medical devices ($\mathrm{ME}_1, \mathrm{ME}_2, \mathrm{ME}_3, \ldots, \mathrm{ME}_M$) in different areas that can communicate with any edge server or cloud. Each ME is guaranteed to be within the communication range of all edge servers and can communicate directly with the cloud. ME-edge network framework is described in Fig. 1.

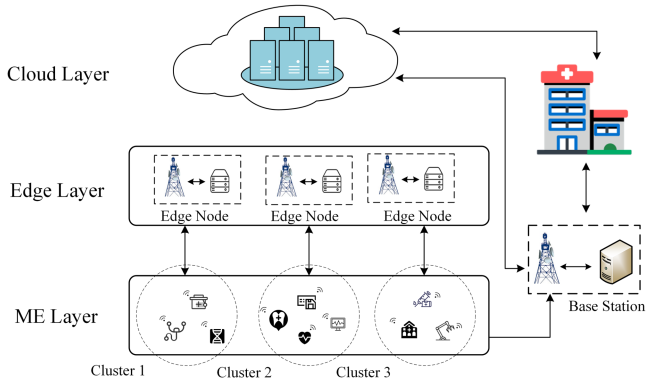


Fig. 1. ME-edge network framework.

In the ME-edge network framework, each edge server is equipped with Z transceiver antennas and each ME is equipped with a single transceiver antenna. An edge server can provide data processing for multiple tasks from different MEs but a ME can only communicate with a maximum of one edge server at a certain time. Obviously, the communication process between the edge server and ME can be viewed as a single-input and multiple-output (SIMO) model. The transmission rate (R_{edge}) between MEs and edge servers can be calculated by

$$R_{\mathrm{edge}} = W_{\mathrm{edge}} \log_2 \left(1 + \frac{S_e}{N_d} \sum_{i=1}^{Z} |H_i|^2\right) \tag{1}$$

where W_{edge} represents the channel bandwidth and the signal energy and noise power spectral density are denoted by S_e and N_d separately. The channel matrix H is a vector whose size is $1 \times Z$. H_i is an element of the channel matrix H, which represents the channel attenuation coefficient of the link between ME and the ith transceiver antenna of the edge server, which can be calculated by the ratio of the signal intensity of the transceiver end. In fact, $\frac{S_e}{N_d} \sum_{i=1}^{Z} |H_i|^2$ is the signal-to-noise ratio of the communication channel between ME and edge servers.

The cloud can provide data processing services for multiple MEs simultaneously. In the ME-edge network framework, the cloud and ME communicate through a single antenna, and the communication process between them can be expressed as a single-input and single-output (SISO) model. The transmission rate (R_{cloud}) between MEs and the cloud can be calculated by

$$R_{\mathrm{cloud}} = W_{\mathrm{cloud}} \log_2 (1 + \mathrm{SNR}_{\mathrm{cloud}}) \tag{2}$$

where W_{cloud} and $\mathrm{SNR}_{\mathrm{cloud}}$ represent the channel bandwidth and the signal-to-noise ratio of the communication channel between MEs and the cloud, respectively.

B. Time Delay Model

When ME_i offloads part of the task data to E_j and the cloud for processing, the time delay generated by task processing consists of three parts, namely, local time delay (T_i^{local}), cloud time delay (T_i^{cloud}), and time delay of E_j (T_{ij}^{edge}).

Let ds_i represents the total task data size of ME_i. η_i and θ_i represent the proportion of ME_i offloading data to edge server and cloud, respectively. T_{ij}^{edge} is divided into the time for E_j to process the data and the delay for ME_i to transmit the original data to E_j. Therefore, the time delay generated by E_j can be calculated by

$$T_{ij}^{\text{edge}} = \eta_i \frac{\mathrm{ds}_i}{p_j^{\text{edge}}} + \eta_i \frac{\mathrm{ds}_i}{R_{\text{edge}}} \tag{3}$$

where p_j^{edge} is the computing capability of E_j and transmission delay is calculated by $\eta_i \frac{\mathrm{ds}_i}{R_{\text{edge}}}$. Since the result data are very small compared with the amount of data transmitted, the delay of transmitting result data from E_j to ME_i is ignored.

When calculating T_i^{cloud}, the latency associated with transferring result data from the cloud to ME_i is not taken into account. T_i^{cloud} can be obtained by

$$T_i^{\text{cloud}} = \theta_i \frac{\mathrm{ds}_i}{p^{\text{cloud}}} + \theta_i \frac{\mathrm{ds}_i}{R_{\text{cloud}}} \tag{4}$$

where p^{cloud} represents the computing capability of cloud.

Let p_i denote the computing capability of ME_i. T_i^{local} can be obtained by

$$T_i^{\text{local}} = (1 - \eta_i - \theta_i) \frac{\mathrm{ds}_i}{p_i}. \tag{5}$$

The minimum value between T_i^{local}, T_i^{cloud}, and T_{ij}^{edge} is the final time delay generated by ME_i task processing (T_i). Thus, T_i can be expressed by

$$T_i = \min\left(T_i^{\text{local}}, T_i^{\text{cloud}}, T_{ij}^{\text{edge}}\right). \tag{6}$$

The average time delay of the entire system T^{ave} can be calculated by

$$T^{\text{ave}} = \frac{1}{M} \sum_{i=1}^{M} T_i. \tag{7}$$

C. Energy Consumption Model

When calculating the total energy consumption of IoMT, the energy consumption generated by edge servers and MEs should be considered, which is mainly generated by calculation energy consumption ($\mathrm{EC}_i^{\text{cal}}$) and data transmission energy consumption ($\mathrm{EC}_i^{\text{trans}}$). The energy consumption generated by cloud computing is not included in $\mathrm{EC}_i^{\text{cal}}$. $\mathrm{EC}_i^{\text{cal}}$ consists of the energy consumption generated by data processing of ME_i and E_j. $\mathrm{EC}_i^{\text{trans}}$ includes the energy consumption of transmission of original data from ME_i to the cloud and E_j.

Let w_i^{local} and w_j^{edge} denote the energy power of ME_i and E_j, respectively. The energy power of data transmission from ME_i to E_j is represented by w_{ij}. The energy power of data transmission from ME_i to the cloud is denoted by w_i^{cloud}. $\mathrm{EC}_i^{\text{cal}}$ and $\mathrm{EC}_i^{\text{trans}}$ can be calculated by

$$\mathrm{EC}_i^{\text{cal}} = \eta_i w_j^{\text{edge}} \frac{\mathrm{ds}_i}{p_j^{\text{edge}}} + (1 - \eta_i - \theta_i) w_i^{\text{local}} \frac{\mathrm{ds}_i}{p_i} \tag{8}$$

$$\mathrm{EC}_i^{\text{trans}} = \eta_i w_j^{\text{edge}} \frac{\mathrm{ds}_i}{R_{\text{edge}}} + \theta_i w_i^{\text{cloud}} \frac{\mathrm{ds}_i}{R_{\text{cloud}}} \tag{9}$$

respectively.

Energy consumption generated by ME_i (EC_i) can be obtained by

$$\mathrm{EC}_i = \mathrm{EC}_i^{\text{cal}} + \mathrm{EC}_i^{\text{trans}}. \tag{10}$$

The total system energy consumption of IoMT (EC^{tol}) can be calculated by adding the energy consumption generated by M MEs. EC^{tol} can be expressed by

$$\mathrm{EC}^{\text{tol}} = \sum_{i=1}^{M} \mathrm{EC}_i. \tag{11}$$

D. Load Balancing of Edge Servers

In order to make full use of the computing resources of each edge server, it is very important to ensure the load balance among all edge servers. If uneven data distribution occurs during task offloading, some edge servers may be overloaded or the utilization of some edge servers may be extremely low. If an edge server is overloaded, then the offloading scheme is not qualified anyway. If an edge server is overloaded, then the offloading scheme is not qualified anyway.

Generally, the total amount of data received by an edge server can be used as a measure of its load. However, in the IoMT scenario, each edge server has different computing resources, resulting in different data processing capabilities and load-bearing capabilities. Load to computing capacity ratio (LCR) is used as a measure of edge server workload, and its value is equal to the ratio of the total amount of data received by the edge server to its computing capacity. Let $\mathrm{ds}_i^{\text{tol}}$ denote the total data received by E_i. The value of LCR of E_i (LCR_i) can be obtained by

$$\mathrm{LCR}_i = \frac{\mathrm{ds}_i^{\text{tol}}}{p_i^{\text{edge}}}. \tag{12}$$

The square deviation of LCR of all edge servers can be calculated to indicate the deviation degree of LCR from the average value. Through this, the load balancing situation among all edge servers can be obtained. $\overline{\mathrm{LCR}}$ ($\overline{\mathrm{LCR}} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{LCR}_i$) is employed to represent the average value of all edge servers' LCR. The square deviation of LCR (σ_{LCR}^2) can be obtained by

$$\sigma_{\text{LCR}}^2 = \frac{1}{N} \sum_{i=1}^{N} \left(\mathrm{LCR}_i - \overline{\mathrm{LCR}}\right). \tag{13}$$

σ_{LCR}^2 represents the dispersion degree of all LCR values. The smaller its value is, the more balanced the load is among all edge servers.

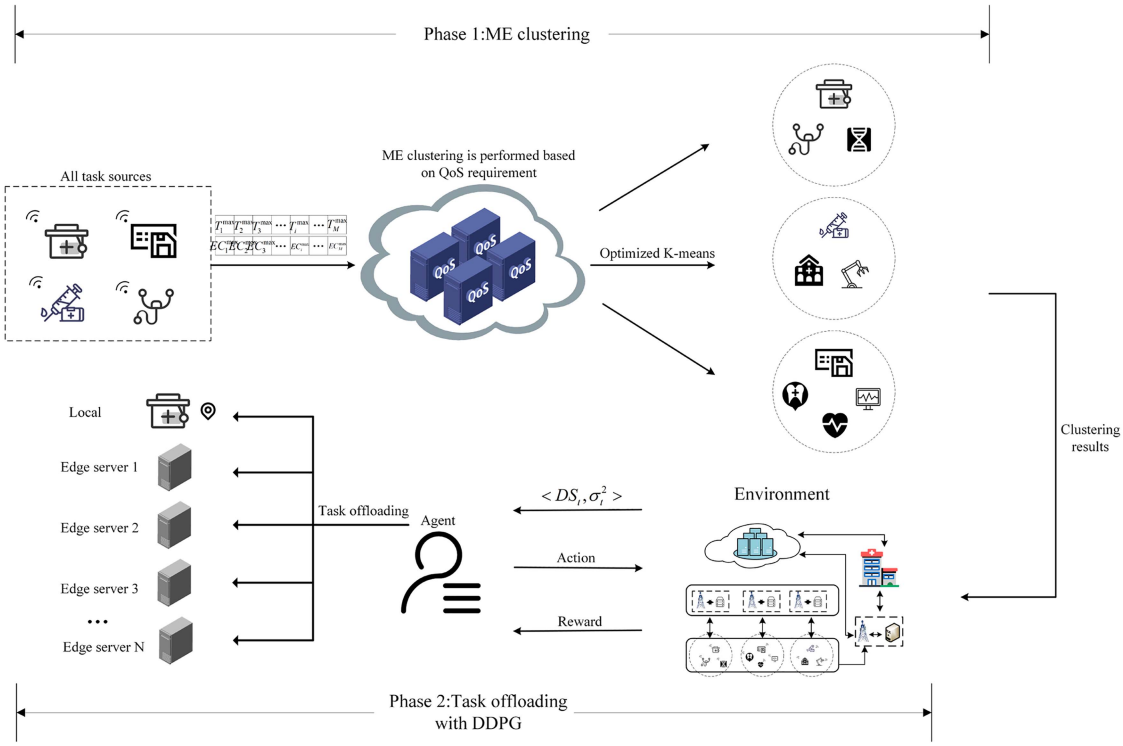


Fig. 2. Overview of DDPG-based task-offloading method assisted by ME clustering.

E. Problem Formulation

In this article, we focus on obtaining a task-offloading scheme to minimize the overall power consumption and the average time delay of the IoMT system while ensuring the load balancing among edge servers. The problem formulation can be expressed as

$$\min T^{\text{ave}},\ \min \text{EC}^{\text{tol}},\ \min \sigma^2_{\text{LCR}} \tag{14}$$

$$\text{s.t.} \quad \sigma^2_{\text{LCR}} < \tau \tag{15}$$

$$\text{LCR}_x \leq \text{LCR}_x^{\max} \quad \forall x = 1, 2, \ldots, N \tag{16}$$

$$T_i \leq T_i^{\max} \quad , \text{EC}_i \leq \text{EC}_i^{\max} \tag{17}$$

where τ is a preset parameter. Equation (15) guarantees that load balancing must meet a minimum of τ. Equation (16) ensures that each edge server cannot exceed the maximum workload it can handle. $T_i^{\max}$ and $\text{EC}_i^{\max}$ indicate the maximum time delay and energy consumption that ME_i can endure. Equation (17) ensures that the resulting task-offloading scheme is such that time delay and energy consumption generated by each ME do not exceed its maximum tolerable limit.

IV. DDPG-Driven Computation Offloading Algorithm Assisted by an Improved *K*-Means

Since the data amount of each task to be processed is different, and different MEs have various delay requirements and different capacities for enduring energy consumption, offloading so many tasks with huge differences together will greatly aggravate the risk of IoMT collapse. If all tasks are simply mixed together and offloaded at the same time, some edge servers may be occupied for a long time or even overloaded, and some tasks will be difficult to be processed in time, which ultimately prevents the normal operation of the IoMT. In our IoMT system, each ME possesses a maximum tolerable delay and a maximum tolerable energy consumption and from these two values, the QoS of the ME can be calculated. Then, we utilize a clustering algorithm to divide ME with similar QoS values into the same group to avoid the competition among tasks that are extremely different from each other. In the decision process of task offloading, every time a task is offloaded, the IoMT environment will change. Thus, every decision step depends on the decision of the previous step. The task-offloading process can be modeled as a Markov decision process. A deep reinforcement learning algorithm is very good at making action decisions according to the current environment. Therefore, DRL is employed to obtain an optimal task-offloading scheme. Fig. 2 is an overview of the DDPG-based task-offloading method assisted by ME clustering.

A. Clustering MEs With an Optimized K-Means

In this article, only the maximum time delay and energy consumption that a task can bear are taken as the measurement criteria of its service quality requirements, and there is no other noise information. The *K*-means algorithm has the characteristics of low computational complexity, easy implementation, strong robustness, and high interpretability, which is suitable for ME clustering. The traditional *K*-means clustering algorithm divides the dataset into k different clusters and adjusts centers of clusters iteratively until the optimal clustering effect is achieved. However, traditional *K*-means uses random generation of cluster centers to carry out the initialization process. If the initialization

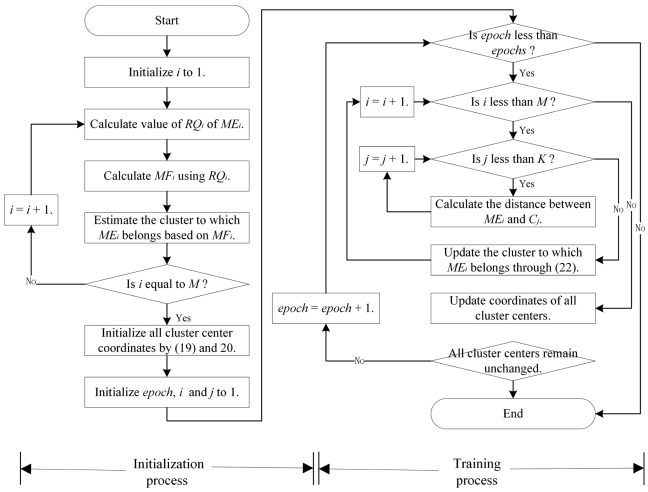


Fig. 3. Flow diagram of the optimized *K*-means.

result of the cluster is not good enough, the final clustering result is often difficult to reach the global optimal and fall into the local optimal. Therefore, the traditional *K*-means algorithm cannot be directly applied to ME clustering. In order to solve this algorithm defect, we designed a more scientific initialization process to make the clustering result closer to the global optimal solution and, finally, obtained an optimized version of the *K*-means algorithm. Fig. 3 shows the flow diagram of the optimized *K*-means.

1) Requirement of Quality of Service: Let $T_i^{\max}$ and $\mathrm{EC}_i^{\max}$ denote the maximum tolerable delay and the maximum tolerable energy consumption of ME_i, respectively. When clustering MEs, the tuple $< T_i^{\max}, EC_i^{\max} >$ are used to denote the location of ME_i. According to $T_i^{\max}$ and $EC_i^{\max}$, the requirement of QoS of ME_i (RQ_i) can be calculated by

$$\mathrm{RQ}_i = \frac{1}{\mathrm{EC}_i^{\max}} lg\left(1 + \zeta - T_i^{\max}\right) \tag{18}$$

where ζ is a preset constant that greater than $T_i^{\max}$. Equation (18) ensures that the smaller the value of $\mathrm{EC}_i^{\max}$ and $T_i^{\max}$, the higher the requirement of QoS of ME_i. The value of RQ_i is inversely proportional to $\mathrm{EC}_i^{\max}$ and directly proportional to minus $T_i^{\max}$. RQ_i is a significant criterion to determine which cluster each ME belongs to during the initialization process.

2) Initialization Process: A membership function MF with RQ_i as input is used as the basis to estimate the cluster to which each ME belongs in the initialization stage. The membership function MF is expressed by

$$\mathrm{MF}\left(\mathrm{RQ}_i\right) = \begin{cases} 0, & 0 < \mathrm{RQ}_i \le \phi \\ 1 \Big/ \left[1 + \left(\frac{1}{\mathrm{RQ}_i - \phi}\right)^2\right], & \phi < \mathrm{RQ}_i \le \chi \\ 1 + \ln\left(1 + \mathrm{RQ}_i - \chi\right), & \mathrm{RQ}_i > \chi. \end{cases} \tag{19}$$

Through (18) and (19), the membership function value of ME_i (MF_i) can be obtained to estimate the cluster that ME_i belongs to. For example, when the number of clusters K is equal to 3, all MEs whose MF value is less than constant γ are divided into the first cluster, and those whose MF value is greater than constant δ are divided into the third cluster, and the remaining MEs are all grouped into the second cluster. These three clusters, respectively, represent the set of MEs with high service quality requirements, the set of MEs with low service quality requirements, and the set of MEs with medium service quality requirements.

After completing the initialization of all clusters to which ME belongs, the average value of coordinates of MEs can be used to initialize the coordinates of cluster center C_k. The location of C_k ($\mathrm{LOC}(C_k) = (C_k^T, C_k^{\mathrm{EC}})$) can be obtained by

$$\mathrm{LOC}\left(C_k\right) = \left(\frac{1}{M}\sum_{i=1}^{M}\rho_{ik}T_i^{\max}, \frac{1}{M}\sum_{i=1}^{M}\rho_{ik}\mathrm{EC}_i^{\max}\right) \tag{20}$$

$$\text{s.t.} \quad \rho_{ik} = \begin{cases} 0, & \mathrm{ME}_i \notin \mathrm{CL}_k \\ 1, & \mathrm{ME}_i \in \mathrm{CL}_k \end{cases} \tag{21}$$

where CL_k represents the set of all MEs that belong to the kth cluster. When ME_i belongs to the kth cluster, the value of ρ_{ik} is 1; otherwise, the value of ρ_{ik} is 0.

3) Training Process: After the initialization process has been completed, we continued to update which cluster each ME belongs to and cluster center coordinates to obtain an optimal clustering result ultimately. We utilize the Euclidean distance to measure the distance from each ME to the center of the cluster. The distance between ME_i and C_k (dis_{ik}) can be calculated by

$$\mathrm{dis}_{ik} = \left[\left(T_i^{\max} - C_k^T\right)^2 + \left(\mathrm{EC}_i^{\max} - C_k^{\mathrm{EC}}\right)^2\right]^{\frac{1}{2}}. \tag{22}$$

Equation (22) is employed to calculate the distance between ME_i and all cluster centers, and then, the cluster with the smallest distance from its cluster center to ME_i will be selected as the new cluster to which ME_i belongs. The new cluster to which ME_i belongs can be obtained by

$$C_i^{\mathrm{new}} = \arg\max_k \left[\left(T_i^{\max} - C_k^T\right)^2 + \left(\mathrm{EC}_i^{\max} - C_k^{\mathrm{EC}}\right)^2\right]. \tag{23}$$

After updating the cluster to which each ME belongs to, (20) and (21) are used to update the coordinates of cluster centers. Repeat the above training process until cluster centers no longer changes or the number of iterations reaches the upper limit. Finally, an optimal clustering result of MEs is obtained. The detailed process of our optimized *K*-means for clustering MEs is described by Algorithm 1.

N_E represents the number of iterations of the optimized *K*-means algorithm in the training process. *K* and *M* are the number of clusters and MEs, respectively. Since the coordinates of each ME are determined by the maximum time delay and energy consumption it can withstand, the dimension of each cluster object is 2. Compared with the training process, the time loss of the optimized K-means algorithm in the initialization process is ignored. Therefore, the time complexity and space complexity of the optimized *K*-means algorithm are $O(2N_EKM)$ and $O(2M)$, respectively.

B. DDPG-Based Task-Offloading Method for MEs

Each ME makes the decision of task offloading according to the current IoMT environment and offloads part of the data to an edge server and the cloud for data processing. When an ME completes the task-offloading process, it is bound to change the

Algorithm 1: Optimized *K*-means Algorithm for Clustering MEs.

Initialization Process :
for $i \leftarrow 1$ *to* M **do**
  Calculate the value of RQ_i for ME_i by $\mathrm{RQ}_i = \frac{1}{EC_i^{\max}} lg\,(1+\zeta-T_i^{\max})$.
end
for $i \leftarrow 1$ *to* M **do**
  Calculate the value of ME_i for ME_i through (19).
end
for $i \leftarrow 1$ *to* M **do**
  Estimate which cluster ME_i belongs to according to ME_i, γ, and δ.
end
for $k \leftarrow 1$ *to* K **do**
  Initialize the coordinates of C_k through (20) and (21).
end
Iteration :
for $epoch \leftarrow 1$ *to* $epochs$ **do**
  for $i \leftarrow 1$ *to* M **do**
    for $k \leftarrow 1$ *to* K **do**
      Calculate the distance between ME_i and C_k by (22).
    end
    Update the cluster to which ME_i belongs to (23).
  end
  for $k \leftarrow 1$ *to* K **do**
    Update the coordinate of all C_k by (20) and (21).
  end
  if *All cluster centers remain unchanged* **then**
    break.
  end
end

state of the IoMT environment, for example, the load that edge servers can bear and the load balance between edge servers will change. Therefore, the action taken by each ME is based on the task-offloading action taken by the previous ME. Obviously, the task-offloading process in IoMT can be viewed as a Markov decision process. In addition, with all the decision options available to MEs, finding an optimal task offload scheme is an NP-hard problem. It is worth noting that the ME selects actions in a continuous space during task offloading. DDPG, with its strong perception and decision-making ability, is outstanding in dealing with continuous control problems.

1) Algorithm Framework: DDPG is a deterministic actor–critic algorithm. The algorithm core of DDPG is a policy network $\pi(s_t;\alpha)$ and a value network $q(s_t,a_t;\beta)$. Let s_t and a_t denote the state of the environment and the action that the agent takes under step t, respectively. α and β are parameters of the policy network and the value network, respectively. According to the current environment, the policy network decides the next action to be taken by the agent, so it is also called an actor. Once the state of the environment is determined, the output action of the policy network is also determined. The value network is responsible for scoring the actions of the policy network output, which is a real number. The value network is also known as a critic. The higher the score of value network output, the better the decision made by the policy network. The value network does not participate in the decision-making of the agent, but it guides the policy network to make improvements. The policy network and the value network are trained together. Our objective is to make actor and critic progress together and ultimately achieve the goal of enabling actor to make better decisions while making the score of action more accurate.

2) Action Space and State Space: Different from discrete control problems, the agent's action space in DDPG is continuous and has countless action choices. When performing task offloading, each ME should consider how much proportion of task data to be offloaded to the cloud and how much to be offloaded to the edge server, and decide which edge server should be taken as the task-offloading target. Therefore, the agent's action has three degrees of freedom. The three dimensions of the action are the proportion of data offloaded to the cloud, the proportion of data offloaded to the edge, and the selected edge server. For agent, the size of the action space is $[0,1]\times[0,1]\times N$.

The state in phase t (s_t) can be represented by the tuple $<\mathrm{DS}_t,\sigma_t^2>$. DS_t is a one-dimensional vector with a size of N, which represents the set of the maximum amount of data that each edge server can receive at the stage t. σ_t^2 represents the square deviation of the LCR value of all edge servers at stage t. When s_t is input into the neural network, each element of DS_t will be input by a corresponding neuron.

3) Reward Function: Every time the agent makes an action, it will receive a reward. In order to obtain an optimal training result, we need to ensure that the cumulative rewards obtained by agents in the training process are maximized. Assuming that in state s_t, ME_i offloads η_i^t of the original data to E_j and θ_i^t of the original data to the cloud, then the reward r_t obtained in stage t can be calculated by

$$r_t = \left(\mathrm{LCR}_j - \mathrm{LCR}_j^{\max}\right)\cdot\left(\frac{1}{T_{ij}^{\mathrm{edge}}}+\frac{1}{\mathrm{EC}_{ij}^{\mathrm{edge}}}\right) + \left(\frac{1}{\mathrm{EC}_i^{\mathrm{local}}}+\frac{1}{\mathrm{EC}_i^{\mathrm{cloud}}}\right)+\frac{1}{\min\left(T_i^{\mathrm{local}},T_i^{\mathrm{cloud}},T_{ij}^{\mathrm{edge}}\right)}. \tag{24}$$

$\mathrm{EC}_{ij}^{\mathrm{edge}}$ represents the sum of the energy consumption generated by E_j processing data and the energy consumption generated by transmitting data from ME_i to E_j and its value

is equal to $\eta_i w_j^{\text{edge}} \frac{\text{ds}_i}{p_j^{\text{edge}}} + \eta_i w_j^{\text{edge}} \frac{\text{ds}_i}{R_{\text{edge}}}$. Besides, $\text{EC}_i^{\text{local}} = (1 - \eta_i - \theta_i) w_i^{\text{local}} \frac{\text{ds}_i}{p_i}$ and $\text{EC}_i^{\text{cloud}} = \theta_i w_i^{\text{cloud}} \frac{\text{ds}_i}{R_{\text{cloud}}}$ represent the energy consumed by ME_i's local computing and the energy consumption generated by sending data from ME_i to the cloud, respectively.

4) Training Process: To make the algorithm performance more stable and avoid bootstrapping, target policy network $\pi(s_t; \widehat{\alpha})$ and target value network $q(s_t, a_t; \widehat{\beta})$ with the same structure as the policy network and value network and different parameters are used in the training process respectively. $\widehat{\alpha}$ and $\widehat{\beta}$ are parameters of the target policy network and the target value network separately. Given the current environment state s_t, the policy network is employed to calculate the next action to be taken through $a_t = \pi(s_t; \alpha)$. Then, the value network scores a_t and obtains the scoring result $q_t = q(s_t, a_t; \beta)$.

To train the parameters of the policy network, we have to rely on the value network. Our goal is to update the policy network to make the value network give high scores to decisions it makes. Therefore, we can calculate the gradient of $q(s_t, a_t; \beta)$ with respect to α and use gradient ascent to update the parameters of the policy network. The gradient of $q(s_t, a_t; \beta)$ with respect to α is $\frac{\partial q(s_t, a_t; \beta)}{\partial a} \cdot \frac{\partial a}{\partial \alpha}$ and α can be updated by

$$\alpha = \alpha + L_1 \cdot \frac{\partial q\left(s_t, a_t; \beta\right)}{\partial a} \cdot \frac{\partial a}{\partial \alpha} \tag{25}$$

where L_1 is the preset learning rate.

In order to avoid bootstrapping and reduce deviation, target networks are employed to update the parameters of the value network. First, utilize target policy network to estimate the action $a_{t+1}^{-} = \pi(s_{t+1}; \widehat{\alpha})$ to be taken in phase $t+1$. Target value network calculates the value in phase $t+1$ through $q_{t+1} = q(s_{t+1}, a_{t+1}^{-}; \widehat{\beta})$. TD error can be obtained by

$$\Delta_t = q\left(s_t, a_t; \beta\right) - \left[r_t + \xi \cdot q\left(s_{t+1}, a_{t+1}^{-}; \widehat{\beta}\right)\right] \tag{26}$$

where ξ is attenuation coefficient. And then we use gradient descent to update β. β can be updated by

$$\beta = \beta - L_2 \cdot \Delta_t \cdot \frac{\partial q\left(s_t, a_t; \beta\right)}{\partial \beta} \tag{27}$$

where L_2 is the preset learning rate.

Parameters of target networks also require updating. Set a hyperparameter ι between 0 and 1 first. Then, we update $\widehat{\alpha}$ by taking a weighted average of α and $\widehat{\alpha}$. Updates to $\widehat{\beta}$ can be done in the same way. Thus, $\widehat{\alpha}$ and $\widehat{\beta}$ can be updated through

$$\widehat{\alpha} = \iota \cdot \alpha + (1 - \iota) \cdot \widehat{\alpha} \tag{28}$$

$$\widehat{\beta} = \iota \cdot \beta + (1 - \iota) \cdot \widehat{\beta} \tag{29}$$

respectively.

n_1 and n_2 are used to represent the number of episodes and the number of agent decisions in each episode. When n_1 and n_2 values are large, the time of parameter initialization of the DDPG-based task-offloading algorithm in the training process can be ignored. The time complexity of DDPG is $O(n_1 n_2)$.

Experience replay is utilized to improve the effectiveness of the training process. We store a set of learned experiences in the experience pool and then randomly select some past experiences to learn during the training process. This approach disrupts the correlation between experiences, making the training process of neural networks more efficient. Algorithm 2 describes the DDPG-based task-offloading method for MEs in detail.

TABLE II
PARAMETER SETTINGS

Parameter	Value
Number of edge servers N	45
Number of clusters K	3
Computing capability of ME	10Mb/s 20Mb/s
Computing capability of the edge server	150Mb/s 200Mb/s
Learning rate L_1 L_2	0.001,0.001
Activation function	ReLU
Number of neurons in the two hidden layers	80, 120
Hyperparameter ι	0.02
Attenuation coefficient ξ	0.1

Algorithm 2: DDPG-Based Task-Offloading Method for MEs.

Iterator :
for *episode* $\leftarrow 1$ *to episodes*
do
  for $t \leftarrow 1$ *to* M
  do
    Input s_t into the policy network to compute a_t by $a_t = \pi\left(s_t; \alpha\right)$.
    The value network scores a_t by $q_t = q\left(s_t, a_t; \beta\right)$.
    Gradient ascent is employed to update the policy network:$\alpha = \alpha + L_1 \cdot \frac{\partial q(s_t, a_t; \beta)}{\partial a} \cdot \frac{\partial a}{\partial \alpha}$.
    Utilize target policy network to estimate a_{t+1} by $a_{t+1}^{-} = \pi\left(s_{t+1}; \widehat{\alpha}\right)$.
    Utilize target value network to calculate q_{t+1} through $q_{t+1} = q\left(s_{t+1}, a_{t+1}^{-}; \widehat{\beta}\right)$.
    Obtain TD error by $\Delta_t = q\left(s_t, a_t; \beta\right) - \left[r_t + \xi \cdot q\left(s_{t+1}, a_{t+1}^{-}; \widehat{\beta}\right)\right]$.
    Update the value network by $\beta = \beta - L_2 \cdot \Delta_t \cdot \frac{\partial q(s_t, a_t; \beta)}{\partial \beta}$.
    Update parameters of target networks through
    $\widehat{\alpha} = \iota \cdot \alpha + (1 - \iota) \cdot \widehat{\alpha}$
    and
    $\widehat{\beta} = \iota \cdot \beta + (1 - \iota) \cdot \widehat{\beta}$.
  end
end

V. EXPERIMENTAL EVALUATION

In this part, we justify the superiority of our proposed optimized *K*-means algorithm and the DDPG-based task-offloading method through multiple groups of comparative experiments. We compare our proposed algorithms with other typical methods. Many different indexes have proved the superiority of the optimized *K*-means algorithm. In addition, experimental results

show that the algorithm has excellent performance in reducing the total energy consumption of the system, reducing the average time delay of the system, and ensuring the load balance between edge servers. We employed Pytorch framework for ME clustering and DDPG task offloading in Python3.8 environment.

A. Parameter Settings

In the IoMT environment, each ME enables to offload data to the cloud or to any edge server for further processing. The number of edge servers N is set to 45. In order to ensure the normal operation of IoMT and improve the efficiency of data processing, we first clustered ME into K group. The value of K is set to 3, based on the results of subsequent experiments. The local computing capability of ME is between 10 Mb/s and 20 Mb/s and the computing capability of edge server is between 150 Mb/s and 200 Mb/s. The number of layers of the DDPG neural network is 3. Use ReLU as the activation function. The number of neurons in the two hidden layers is 80 and 120. Learning rate L_1 and L_2 is set to 0.001 and 0.001. Hyperparameter ι, which is employed to update target networks, is set to 0.02. Attenuation coefficient ξ, which is used to calculate Δ_t, is set to 0.1. Parameter settings are shown in Table II.

B. Evaluation of Optimized K-Means Algorithm for Clustering MEs

1) Algorithms for Comparison: In this section, the optimized *K*-means algorithm is compared with *K*-means and fuzzy *C*-means in the performance of clustering MEs. *K*-means and fuzzy *C*-means are outlined as follows.

a) *K*-means: *K*-means first initializes the cluster to which each element belongs and cluster centers in a random way and then updates the result by calculating the distance between each element and each cluster center [38]. However, random initialization makes it easy for *K*-means to achieve only local optimum.

b) Fuzzy *C*-means: Fuzzy *C*-means incorporates the essence of fuzzy theory [39],[40]. Compared with *K*-means hard clustering, fuzzy *C* provides more flexible clustering results. Fuzzy *C*-means is a process of iteratively calculating the membership degree and cluster center until they reach the optimum.

The above two clustering algorithms will be used for comparison with optimized *K*-means.

2) Evaluation Indicators: The sum of squares due to error (SSE) and silhouette coefficient (SC) are employed as evaluation indicators to evaluate the performance of clustering algorithms. SSE is used to measure the looseness of members in a cluster and it can be calculated by

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{\mathrm{ME}_j \in C_i} (\mathrm{dis}_{ji})^2 . \tag{30}$$

SC is the evaluation index of the density and the dispersion degree of a cluster. SC can be calculated by

$$\mathrm{SC} = \frac{(b-a)}{\max(a,b)} \tag{31}$$

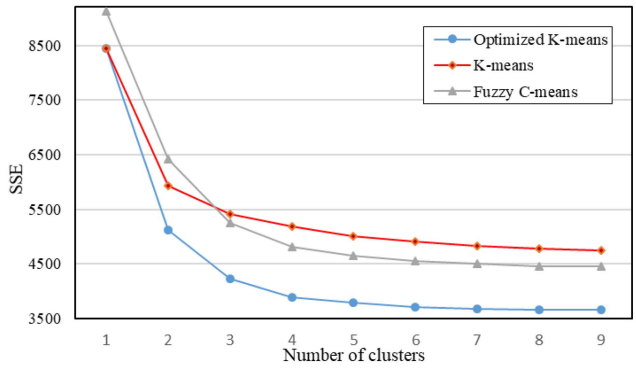


Fig. 4. SSE value under different number of clusters.

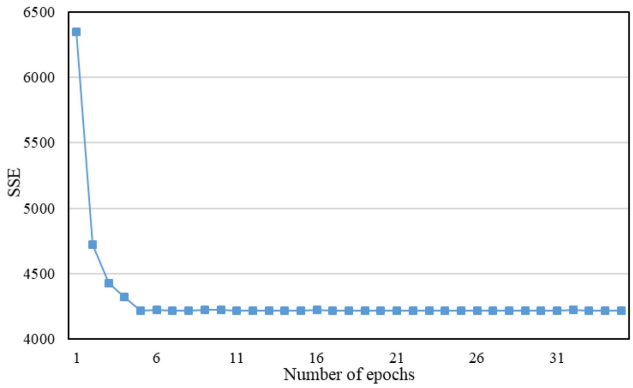


Fig. 5. SSE value during training of optimized *K*-means.

where a represents the average degree of dissimilarity between a member and other members in the same cluster and b represents the minimum value of the average dissimilarity degree from a member to other clusters.

3) Selection of K: The sum of the square distance error between the particle of each cluster and the sample point in the cluster is called distortion. For a cluster, the lower the distortion degree is, the closer the members in the cluster are; the higher the distortion degree is, the looser the structure in the cluster is. The degree of distortion will decrease with the increase of the category, but for the data with a certain degree of discrimination, the degree of distortion will be greatly improved when it reaches a certain critical point, and then slowly decreases. This critical point can be considered as the point with better clustering performance. SSE can work as an indicator to measure the distortion degree of clusters, and the appropriate K can be selected based on SSE.

Fig. 4 shows the tendency of SSE values obtained by the three clustering algorithms to change as K increases. It can be seen from Fig. 4 that regardless of which clustering algorithm is employed, the value of SSE drops sharply at K from 1 to 3 and then stabilizes. Since the decline rate of SSE slows dramatically when K is greater than 3, 3 is the optimal number of clusters. Fig. 4 also indicated that no matter what the value of K is, the SSE value of optimized *K*-means is always the smallest, which proves the superiority of the algorithm.

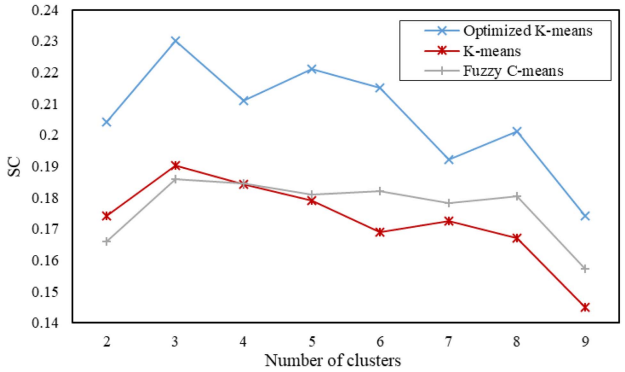


Fig. 6. SC value under different number of clusters.

Fig. 5 shows the change of SSE value in the optimized *K*-means training process when the number of clusters is 3. It can be obviously seen that the SSE value decreased sharply during the first five epochs and finally stabilized at about 4220. This indicates that the optimized *K*-means algorithm has superior convergence performance when K is equal to 3.

4) Evaluation of Clustering Performance: The clustering performance is excellent, that is, the samples of the same cluster are as similar as possible, and the samples of different clusters are as different as possible. SC is the index that can correctly reflect the intracluster similarity and intercluster similarity. The value of SC is between -1 and 1 and the closer its value is to 1, the more reasonable the clustering result is.

Fig. 6 shows the comparison results of SC values obtained by the three clustering algorithms under different K values. As can be seen from Fig. 6, no matter what the number of clusters is, the SC value obtained by the optimized *K*-means is always the largest and closest to 1 among the three comparison algorithms. Therefore, the clustering result obtained by optimized *K*-means is the most reasonable, and the performance of optimized *K*-means is also optimal.

C. Evaluation of DDPG-Based Task-Offloading Method for MEs

In this section, we compare the performance of DDPG, A3C, and DPG, which are DRL algorithms with the actor–critic framework and the binary task-offloading algorithm based on DQN in terms of reducing average delay, reducing total system energy consumption and ensuring load balancing among edge servers. Our ultimate goal is to obtain an optimal task-offloading scheme to achieve the lowest average delay and total energy consumption of the IoMT system while ensuring the load balance of all edge servers. Therefore, we compare the performance of these three algorithms from the perspective of average time delay, total power consumption and edge server load balancing to illustrate the superiority of the DDPG-driven task-offloading method.

In the simulation experiments, we compare how much average time delay and total energy consumption will be brought by task-offloading schemes obtained by the above three algorithms and the load balance of edge servers when the total data amount is 50 GB, 100 GB, 150 GB, 200 GB, or 250 GB. A3C, DPG, and DQN are outlined as follows.

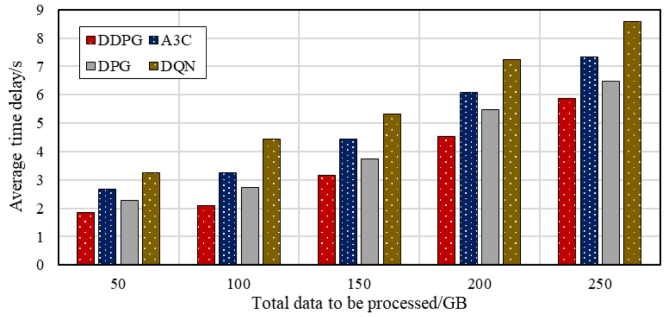


Fig. 7. Comparison of average time delay under different amounts of data among DDPG, A3C, DPG, and DQN.

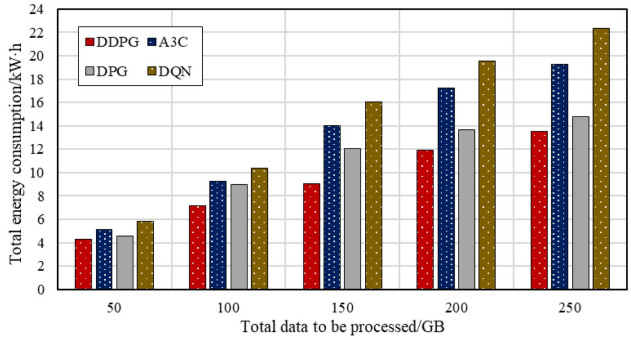


Fig. 8. Comparison of total energy consumption under different amounts of data among DDPG, A3C, DPG, and DQN.

1) *A3C:* A3C greatly improves the difficulty of actor-critic convergence. A3C uses the method of multithreading to conduct interactive learning with the environment in multiple threads at the same time and summarizes the learning results of each thread to guide its subsequent learning interaction with the environment [41].
2) *DPG:* DPG obtains the optimal scheme through the coordination of the policy network and value network [42]. However, unlike DDPG, DPG does not use target networks during training, which cannot alleviate problems of bootstrapping.
3) *DQN:* The binary task-offloading algorithm based on DQN uses the output of a deep neural network to generate the corresponding binary task-offloading strategy. In this approach, the ME either processes all data locally or offloads the task to the nearest edge server.

1) Comparison of Average Time Delay and Total Power Consumption: As can be seen from Figs. 7 and 8, with the increase in the total amount of data, the average time delay and total energy consumption of IoMT both rise. An increase in the total population of the system inevitably leads to greater time delay and energy consumption, and the trends shown in Figs. 7 and 8 are consistent with this. The scheme calculated by DDPG can always obtain the least average time delay and total energy consumption under any data amount. The data in Figs. 7 and 8 fully show that DDPG has excellent performance in reducing system overheads of IoMT and improving system operation efficiency. The binary task-offloading algorithm based

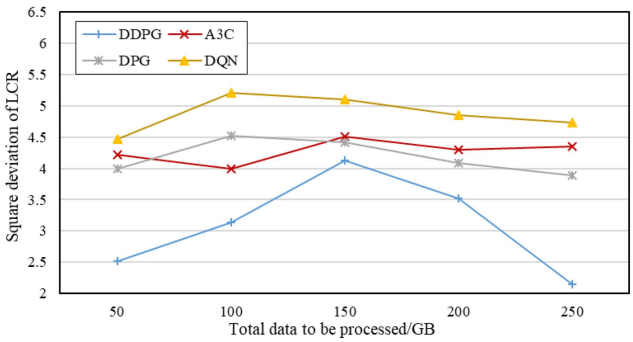


Fig. 9. Comparison of σ^2_{LCR} under different amounts of data among DDPG, A3C, DPG, and DQN.

on DQN always obtains the maximum average delay and total energy consumption.

2) Comparison of Load Balancing Among Edge Servers: Fig. 9 shows the comparison of σ^2_{LCR} under different amounts of data among DDPG, A3C, DPG, and DQN. The LCR value represents the load on the edge server. The square deviation of LCR value σ^2_{LCR} can represent the workload balance of all edge servers. A smaller value of σ^2_{LCR} indicates that data are evenly distributed among edge servers, avoiding low utilization of some edge servers or overload of some edge servers. It can be seen from Fig. 9 that DDPG always gets the smallest σ^2_{LCR} value, regardless of the total data to be processed. The binary task-offloading algorithm based on DQN always gets the largest σ^2_{LCR} value. This shows that the task-offloading scheme obtained by DDPG can allocate data rationally and make the computing resources of edge servers be used more fully. DQN is used to deal with discrete action problems while DDPG is extended to deal with continuous action problems based on DQN. Therefore, DQN and DDPG are suitable for binary task offloading and partial task offloading, respectively. However, compared with partial task offloading, the strategy of binary task offloading cannot make full use of the edge resources in the IoMT system and the local resources of ME. Therefore, DQN is inferior to DDPG in reducing system loss and balancing load among edge servers.

VI. Conclusion

In this article, a task-offloading algorithm based on DRL assisted by clustering is proposed to reduce the costs of the IoMT system and ensure load balancing among edge servers. First, an optimized *K*-means with a more scientific initialization process is designed to cluster MEs to improve the effectiveness of the subsequent task offloading. Then, a DDPG-driven task-offloading method is proposed to minimize the average time delay and total energy consumption of IoMT while ensuring load balancing among edge servers. A large number of comparative experiments demonstrate that an optimized *K*-means is more scientific and in clustering MEs and our unloading algorithm is more advantageous in minimizing system overhead.

However, in this article, only static IoMT systems are considered, and the increasing number of mobile medical devices in a real-world scenario makes the task-offloading problem more complex. The future research will combine the mobile edge computing technology with IoMT to develop a more effective task-offloading method under the premise of fully considering the mobility of MEs. In addition, edge servers in this study cannot transfer data to each other, so the load balancing between them can only be realized by the centralized task-offloading strategy. In future research, we will build an IoMT system, which can realize load balancing by transferring data between edge servers, and expect to achieve better performance through the distributed task-offloading method.

References

[1] Z. Li et al., "Integrated CNN and federated learning for COVID-19 detection on chest X-ray images," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, to be published, 2022, doi: 10.1109/TCBB.2022.3184319.

[2] Z. Li et al., "A knowledge-driven anomaly detection framework for social production system," *IEEE Trans. Comput. Social Syst.*, to be published, 2022, doi: 10.1109/TCSS.2022.3217790.

[3] X. Lin, J. Wu, A. K. Bashir, W. Yang, A. Singh, and A. A. AlZubi, "FairHealth: Long-term proportional fairness-driven 5G edge healthcare in Internet of medical things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 8905–8915, Dec. 2022, doi: 10.1109/TII.2022.3183000.

[4] S. I. Zida, Y.-D. Lin, C. L. Lee, and Y. L. Tsai, "Evaluation of an intelligent edge computing system for the hospital intensive care unit," in *Proc. IEEE 3rd Eurasia Conf. Biomed. Eng., Healthcare, Sustainability*, 2021, pp. 179–182, doi: 10.1109/ECBIOS51820.2021.9510541.

[5] L. Yang, K. Yu, S. X. Yang, C. Chakraborty, Y. Lu, and T. Guo, "An intelligent trust cloud management method for secure clustering in 5G enabled Internet of medical things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 8864–8875, Dec. 2022, doi: 10.1109/TII.2021.3128954.

[6] S. K. S. Tyagi, A. Mukherjee, S. R. Pokhrel, and K. K. Hiran, "An intelligent and optimal resource allocation approach in sensor networks for smart Agri-IoT," *IEEE Sensors J.*, vol. 21, no. 16, pp. 17439–17446, Aug. 2021, doi: 10.1109/JSEN.2020.3020889.

[7] Z. Ming, M. Zhou, L. Cui, and S. Yang, "Faith: A fast blockchain-assisted edge computing platform for healthcare applications," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 9217–9226, Dec. 2022, doi: 10.1109/TII.2022.3166813.

[8] X. Xu, C. Yang, M. Bilal, W. Li, and H. Wang, "Computation offloading for energy and delay trade-offs with traffic flow prediction in edge computing-enabled IoV," *IEEE Trans. Intell. Transp. Syst.*, to be published, 2022, doi: 10.1109/TITS.2022.3221975.

[9] X. Xu, J. Gu, H. Yan, W. Liu, L. Qi, and X. Zhou, "Reputation-aware supplier assessment for blockchain-enabled supply chain in Industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 19, no. 4, pp. 5485–5494, Apr. 2023, doi: 10.1109/TII.2022.3190380.

[10] M. Armbrust et al., "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010, doi: 10.1145/1721654.1721672.

[11] X. Xu, H. Li, Z. Li, and X. Zhou, "Safe: Synergic data filtering for federated learning in cloud-edge computing," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1655–1665, Feb. 2023, doi: 10.1109/TII.2022.3195896.

[12] X. Xu, Q. Wu, L. Qi, W. Dou, S.-B. Tsai, and M. Z. A. Bhuiyan, "Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1787–1796, Mar. 2021, doi: 10.1109/TITS.2020.2995622.

[13] G. Manogaran and B. S. Rawal, "An efficient resource allocation scheme with optimal node placement in IoT-Fog-cloud architecture," *IEEE Sensors J.*, vol. 21, no. 22, pp. 25106–25113, Nov. 2021, doi: 10.1109/JSEN.2021.3057224.

[14] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020, doi: 10.1109/JIOT.2020.2996784.

[15] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019, doi: 10.1109/TVT.2019.2904244.

[16] X. Xu, H. Tian, X. Zhang, L. Qi, Q. He, and W. Dou, "DisCOV: Distributed COVID-19 detection on X-ray images with edge-cloud collaboration," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1206–1219, May/Jun. 2022, doi: 10.1109/TSC.2022.3142265.

[17] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017, doi: 10.1109/TVT.2016.2593486.

[18] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.
[19] X. Zhou et al., "Edge computation offloading with content caching in 6G-enabled IoV," *IEEE Trans. Intell. Transp. Syst.*, to be published, 2023, doi: 10.1109/TITS.2023.3239599.
[20] S.-H. Park, S. Jeong, J. Na, O. Simeone, and S. Shamai, "Collaborative cloud and edge mobile computing in C-RAN systems with minimal end-to-end latency," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 7, pp. 259–274, 2021, doi: 10.1109/TSIPN.2021.3070712.
[21] H. Babbar, S. Rani, and S. A. Alqahtani, "Intelligent edge load migration in SDN-IIoT for smart healthcare," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8058–8064, Nov. 2022, doi: 10.1109/TII.2022.3172489.
[22] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 3448–3459, Sep. 2021, doi: 10.1109/TNSM.2021.3087258.
[23] X. Xu et al., "Edge server quantification and placement for offloading social media services in industrial cognitive IoV," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2910–2918, Apr. 2021, doi: 10.1109/TII.2020.2987994.
[24] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang, "Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2410–2422, Dec. 2020, doi: 10.1109/TNSM.2020.3020249.
[25] X. Xu, S. Tang, L. Qi, X. Zhou, F. Dai, and W. Dou, "CNN partitioning and offloading for vehicular edge networks in Web3," *IEEE Commun. Mag.*, vol. 61, no. 8, pp. 36–42, Aug. 2023, doi: 10.1109/MCOM.002.2200424.
[26] M. S. Mekala, A. Jolfaei, G. Srivastava, X. Zheng, A. Anvari-Moghaddam, and P. Viswanathan, "Resource offload consolidation based on deep-reinforcement learning approach in cyber-physical systems," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 2, pp. 245–254, Apr. 2022, doi: 10.1109/TETCI.2020.3044082.
[27] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, doi: 10.1609/aaai.v32i1.11694.
[28] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[29] P. Lin, Q. Song, F. R. Yu, D. Wang, and L. Guo, "Task offloading for wireless VR-enabled medical treatment with blockchain security using collective reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15749–15761, Nov. 2021, doi: 10.1109/JIOT.2021.3051419.
[30] M. S. Mekala et al., "A DRL-based service offloading approach using DAG for edge computational orchestration," *IEEE Trans. Comput. Social Syst.*, to be published, 2022, doi: 10.1109/TCSS.2022.3161627.
[31] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020, doi: 10.1109/TVT.2020.3041929.
[32] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022, doi: 10.1109/TMC.2020.3036871.
[33] Z. Ning et al., "Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019, doi: 10.1109/TCCN.2019.2930521.
[34] R. Yadav et al., "Smart healthcare: RL-based task offloading scheme for edge-enable sensor networks," *IEEE Sensors J.*, vol. 21, no. 22, pp. 24910–24918, Nov. 2021, doi: 10.1109/JSEN.2021.3096245.
[35] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intell. Converged Netw.*, vol. 1, no. 2, pp. 181–198, 2020, doi: 10.23919/ICN.2020.0014.
[36] Z. Chen, L. Zhang, Y. Pei, C. Jiang, and L. Yin, "NOMA-based multi-user mobile edge computation offloading via cooperative multi-agent deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 350–364, Mar. 2022, doi: 10.1109/TCCN.2021.3093436.
[37] X. He, H. Lu, M. Du, Y. Mao, and K. Wang, "QoE-based task offloading with deep reinforcement learning in edge-enabled internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2252–2261, Apr. 2021, doi: 10.1109/TITS.2020.3016002.
[38] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.
[39] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, no. 2/3, pp. 191–203, 1984.
[40] X. Xu et al., "Game theory for distributed IoV task offloading with fuzzy neural network in edge computing," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 11, pp. 4593–4604, Nov. 2022, doi: 10.1109/TFUZZ.2022.3158000.
[41] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[42] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

Chenyi Yang is currently working toward the B.S. degree in software engineering with the School of Software, Nanjing University of Information Science and Technology, Nanjing, China.

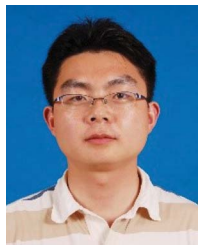
His research interests include deep learning and edge computing.

Xiaolong Xu (Senior Member, IEEE) received the Ph.D. degree in computer science and technology from Nanjing University, Nanjing, China, in 2016.

He was a Research Scholar with Michigan State University, USA, from 2017 to 2018. He is currently a Professor with the School of Software, Nanjing University of Information Science and Technology, Nanjing. His research interests include edge computing, the Internet of Things, cloud computing, and big data.

Muhammad Bilal (Senior Member, IEEE) received the Ph.D. degree in information and communication network engineering from the School of Electronics and Telecommunications Research Institute (ETRI), Korea University of Science and Technology, Daejeon, South Korea, in 2017.

From 2017 to 2018, he was with Korea University, where he was a Postdoctoral Research Fellow with the Smart Quantum Communication Center. In 2018, he joined the Hankuk University of Foreign Studies, South Korea, where he was an Associate Professor with the Division of Computer and Electronic Systems Engineering. He is currently a Senior Lecturer (Associate Professor) with the School of Computing and Communications, Lancaster University, Lancaster, U.K.



Yiping Wen received the Ph.D. degree in computer science from Central South University, Changsha, China, in 2013.

He is currently a Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. From 2015 to 2017, he did his postdoctoral research with the Department of Computer Science and Technology, Nanjing University, China. He is currently a Full Professor with the Hunan Key Laboratory for Service Computing and Novel Software Technology, Hunan University of Science and Technology. His current research interests include business process management, machine learning, big data, and cloud computing.

Tao Huang received the bachelor's and master's degrees in computer science from the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China, in 2010 and 2013, respectively.

He is currently a Lecturer with the School of Computer and Technology, Silicon Lake College, Suzhou, China. His research interests include issues related to artificial intelligence algorithms, big data, and cloud computing.