Journal of Grid Computing (2022) 20: 14
https://doi.org/10.1007/s10723-022-09605-2

Collaborative Cloud-Edge-End Task Offloading in NOMA-Enabled Mobile Edge Computing Using Deep Learning

RuiZhong Du · Cui Liu · Yan Gao · PengNan Hao · ZiYuan Wang

Received: 22 November 2021 / Accepted: 18 March 2022 / Published online: 22 April 2022

Abstract Aiming at the problem that it is quite hard to guarantee the real-time requirements of medical users with high efficiency and low latency in the current Internet of Medical Things (IoMT), we investigate the task offloading for collaborative cloud-edge-end computing in mobile networks. Non-orthogonal multiple access (NOMA) is suitable for wireless networks with higher spectral efficiency, faster speed, and larger capacity, while the existing cloud-edge-end cooperative computing ignores the advantages of NOMA. Therefore, by exploiting NOMA for improving the efficiency of radio transmission, we integrate collaborative cloud-edge-end computing and NOMA to propose a novel network communication model, which can provide medical users with energy-efficient and low latency services. Specifically, considering the energy consumption, transmission delay, and quality of service, we jointly optimize the offloading decision and its radio resource allocations for NOMA-transmission to reduce the system cost (the weighted sum of consumed energy and delay) in the IoMT of cloud-edge-end computing networks supported by NOMA. Although the joint optimization problem is non-convex, we use its hierarchical structure and propose a collaborative computing offloading algorithm based on deep learning to find the optimal offloading solution. Through extensive simulations, it is shown that the proposed algorithm stably converges to its optimal value, provides approximately 25.2% and 79.2% lower system cost than schemes such as only using edge computing and fully local processing, respectively. In addition, compared with the traditional orthogonal multiple access(OMA), our proposed NOMA-enabled multi-access computation offloading can reduce the system cost by approximately 93.4%.

Keywords Computing offloading · Non-orthogonal multiple access · Deep learning · Internet of medical things · Offloading decision · Mobile edge computing

C. Liu (✉) · R. Z. Du · Y. Gao · P. N. Hao · Z. Y. Wang
School of Cyber Security and Computer, Hebei university, Baoding, 071000, Hebei, China
e-mail: luckyliucui@163.com

R. Z. Du
e-mail: durz@hbu.edu.cn

Y. Gao
e-mail: morsiback@foxmail.com

P. N. Hao
e-mail: 877909917@qq.com

Z. Y. Wang
e-mail: zywang@hbu.edu.cn

1 Introduction

Combined with the current development trend of the Internet of Things, the maturing development of Internet of Things devices (IoTDs) is projected to revolutionize healthcare [1, 2]. The healthcare industry has become one of the fastest developing fields in

the application of the Internet of things. For example, the clinical will collect the patient's body temperature, respiratory rate, resting blood oxygen saturation, and oxygenation index detected by the intelligent thermometer and intelligent stethoscope for data analysis. However, IoTDs are usually equipped with limited computing resources, some parts of computing-intensive medical tasks need to be offloaded to the hospital cloud center for processing. The cloud center can collect user information online and accurately complete patient diagnosis and treatment [3]. With the increasing number of medical IoTDs and the emergence of more advanced medical applications, offloading medical tasks to remote cloud centers for processing, it is difficult to ensure the real-time needs of medical users. To solve the problem, mobile edge computing (MEC) is introduced as a supplement to mobile cloud computing (MCC). The MEC deploys computing resources at the network edge so that IoTDs can offload parts of tasks to closer edge nodes for computing and storage, which can reduce the delay of completing computing tasks and improve user quality requirements [4]. Unfortunately, the computing speed and storage capacity of the MEC servers are relatively weak. When dealing with computing-intensive tasks, the capacity and resources will be insufficient. Therefore, MCC and MEC can complement each other. Using the advantages of these two technologies, many researchers have studied the cloud-edge-end collaborative computing model to reduce the delay and energy consumption in task offloading and improve the user's service quality requirements [5, 6].

In the cloud-edge-end cooperative networks, the offloading selection of medical tasks is more flexible. Medical tasks based on different service requirements can be processed on local devices, edge nodes or cloud servers. The existing computing offloading literatures reduce latency or energy consumption by optimizing offloading decision and computing resource allocation [7, 8]. Furthermore, various physical layer technologies are also used to further reduce the energy consumption and computation resources consumption while satisfying the offloading latency, such as multiple-input multiple-output (MIMO) [9] and orthogonal frequency division multiple access (OFDMA) [10]. In fact, the existing cloud-edge-end collaborative computing networks literatures ignore the non-orthogonal multiple access (NOMA). NOMA is an innovative multiple access technology that can allocate a single resource to multiple users, which is very different from the traditional orthogonal multiple access technology. NOMA has higher spectral efficiency and access to more IoTDs, which is more suitable for future wireless communication systems. Therefore, to further improve the offloading efficiency of cloud-edge-end computing networks, we integrate cloud-edge-end computing and NOMA to propose a novel network communication model, which can effectively provide comprehensive and personalized services according to different medical service levels and performance requirements. The main contributions of this paper are summarized as follows:

1) This paper proposes a novel network communication model, which focuses on the optimization of system cost in the IoMT of cloud-edge-end networks supported by NOMA. Considering the energy consumption, transmission delay and quality of service, we jointly optimize the offloading decision and its radio resource allocations for NOMA-transmission to reduce the system cost (the weighted sum of consumed energy and delay).
2) Despite the non-convexity of the joint optimization problem, we identify its layered structure and decompose it into two subproblems for optimizing NOMA-transmission time and offloading decision. For the offloading decision problem, we establish its convexity and use the structural properties of the optimal solution to propose an offloading algorithm based on deep learning. This results in an optimal offloading decision (under a given transmission-time). Our algorithm uses multiple parallel deep neural networks (DNNs) to effectively generate offloading decisions. These generated offloading decisions are stored in a shared memory according to experience replay technique to train DNNs to further improve accuracy. Next, we propose an algorithm based on linear search to find the optimal transmission time. The algorithm guarantees the global delay minimization.
3) We provide a large number of numerical results to verify the convergence and effectiveness of our proposed algorithm. Numerical results show that compared with the traditional orthogonal multiple access algorithm, our proposed algorithm has reliable convergence and near-optimal perfor-

mance in reducing energy consumption and task completion delay.

2 Related work

In recent years, many scholars have done a lot of research on computing offloading in different fields. The research on computing offloading mainly focuses on different fields to reduce energy consumption, delay and improve quality of service by jointly optimizing the parameters such as offloading decision, transmission efficiency and resource allocation. One popular research field in the past decade for providing high-computational capability service is mobile cloud computing (MCC), which can offload the computing tasks of IoTDs to the more powerful cloud center for processing. Literature [11] mainly on integrating cloud services and resources with mobile applications is to reduce battery usage and improve the efficiency of mobile devices. In order to overcome the estimation of communication cost of devices in task offloading, this literature proposes an effective task offloading model, which can improve the efficiency of mobile devices by reducing battery use. Literature [12–17] deploy computing resources at the edge of the network so that IoTDs can offload tasks to closer edge nodes for computing and storage, which can reduce the delay of completing computing tasks and improve resource utilization efficiency. Literature [12] studies that a large number of devices choose to offload tasks to edge servers. To minimize the total delay and corresponding energy consumption of completing all user tasks, a mobile edge computing network offloading algorithm based on distributed deep learning is proposed. The algorithm can produce a near optimal offloading decision in less than one second. The goal of literatures is to minimize energy consumption [13, 14, 16] and task execution delay [15, 18].

In addition, various physical technologies are used to further reduce the energy and computing resource consumption in the edge computing offloading system and meet the offloading delay. Recently, many studies have demonstrated the potential advantages of NOMA, such as improving transmission efficiency, energy efficiency and spectrum efficiency. For example, suppose that only one resource block can transmit data at a given time and that two users need to offload their tasks to the edge node. If traditional OMA transmission is applied, only one user can perform the offloaded tasks, while the other user must wait. However, if NOMA is applied, then both users can offload to the edge node simultaneously. As a result, the device can use NOMA to send its workloads to different edge servers and the cloud server. Using NOMA, the device can offload its computing workloads to multiple edge servers at the same time by using wireless access, so as to further improve the flexibility of offloading. In the literature [19, 20], NOMA allows multiple users to share the same time and spectrum resources, which is better than OMA in spectrum and energy efficiency. Therefore, scholars have made a lot of efforts in the research of mobile edge computing supported by NOMA. The goal of literature is to minimize energy consumption [21, 22] and task execution delay [23–25], and literature [26, 27] achieve a compromise between energy and delay.

These studies only focus on the computing offloading between devices and edge servers, ignoring the huge computing resources in the cloud center. Therefore, task offloading in cloud-edge-end networks is an important topic for many scholars. Literature [28] proposes a general cloud and edge computing architecture to provide vertical and horizontal offloading between service nodes. In order to study the effectiveness of design in different operation scenarios, it is expressed as a workload and capacity optimization problem in order to minimize the system computing and communication costs. Literature [29] developes a low complexity and efficient offloading scheme to minimize the average task duration under the limitation of devices' battery capacity by jointly optimizing offloading decision and computing resource allocation. In order to solve the optimization problem, a series of reconfigurations based on reconfiguration linearization technology are carried out, and a parallel optimization framework based on alternating direction multiplier method and convex function difference method is proposed to further reduce the complexity. The literature [30] studies the task offloading problem of cloud-edge-end collaborative computing in mobile networks. Optimize server selection and resource allocation through federation to minimize the weighted sum of average costs. The paper [31] formulates an adaptive task scheduling (ATS) problem, with the objective of minimizing the overall service latency by best cooperating those heterogeneous nodes on the

cloud, edge and terminal layers. Further, we propose a genetic algorithm to solve the problem.

However, most of the research on computing offloading of cloud-edge-end networks believe that the wireless channel from the devices to different edge servers is OMA. With the increasing number of medical IoTDs and the emergence of more advanced medical applications, the demand for wireless transmission rate increases exponentially. The transmission rate of wireless communication will still be difficult to meet the application demand of mobile communication in the future. The NOMA can be applied to wireless networks with higher frequency, faster rate and larger capacity. Therefore, we integrate cloud-edge-end and NOMA to propose a novel network communication model and realize the balance of high resource consumption and high communication cost between high energy consumption and low delay services. Table 1 summarizes the purpose of computing offloading in different fields and the comparison of communication technologies.

3 System Model And Problem Formation

In this section, we will focus on the system model, communication and computing model of integrating NOMA and the cloud-edge-end computing networks.

3.1 System Model

we investigate a cloud-edge-end collaborative computing network, which consists of I base stations (BSs), N IoTDs, and a distant cloud server. Each BS is equipped with an edge server and can be regarded as an edge node. We denote the set of edge nodes and the set of all IoTDs as $I = \{1, 2, ..., i\}$ and $N = \{1, 2, ..., n\}$. To reduce computing latency, IoTDs offload parts of their computation workloads to edge servers via NOMA-transmission and to a remote cloud via a wired backhaul link. Figure 1 shows a detailed example of the medical application system in the IoMT. Medical users can continuously sense and collect the healthcare information (temperature, blood pressure and heart rate) by wearing thermometer ,smart-watch ,smart-stethoscope and so on. All medical devices in each user work at any time and place for healthcare, resulting in massive data. Then, these data are processed locally or simultaneously offload parts of workloads (for data analysis) to different edge servers and the cloud server via NOMA-transmission and wired backhaul link. Finally, the server provides users with a variety of health services by evaluating and processing the collected sensing data. Each IoTD has a separable application that can be divided into M independent computing tasks to deal with and the set of all tasks is denoted $M = \{1, 2, ..., m\}$. IoTD n will decide which tasks are processed locally, which are offloaded to the edge node, or offloaded to the cloud according to the offloading decision. The tasks of device n are described by $Q_n = \left(L_n, T_n^{\max}\right)$, where L_n stands for the data size of tasks of device n, and $T_n^{\max}$ stands for the maximum delay allowed for device n to complete its tasks. The offloading decision variable of task m of device n is expressed as $\Pi_n^m = \left\{x_n^m, y_n^m, z_n^m\right\}$, $x_{\mathrm{n}}^m, y_n^m, z_n^m \in \{0, 1\}$, which denotes whether task m of device n is processed by the device itself, by the edge node or the cloud server, respectively. The offloading decision of device n is constrained by:

Table 1 Comparison of studies on computing offloading in different fields

Articles	Focus area	Offload vector	Using NOMA	Objective minimize
[11]	Cloud Computing	Mobile device to Cloud	NO	Communication cost
[15, 18]	Edge Computing	Mobile device to MEC	NO	Completion latency
[13, 14, 16]	Edge Computing	Mobile device to MEC	NO	Energy consumption
[12, 17]	Edge Computing	Mobile device to MEC	NO	Completion latency and Energy consumption
[21, 22]	Edge Computing	Mobile device to MEC	YES	Energy consumption
[23–25]	Edge Computing	Mobile device to MEC	YES	Completion latency
[26, 27]	Edge Computing	Mobile device to MEC	YES	Completion latency and Energy consumption
[28]	Cloud-edge-end	Mobile device to MEC or Cloud	NO	Computation and Communication cost
[29–31]	Cloud-edge-end	Mobile device to MEC or Cloud	NO	Completion latency

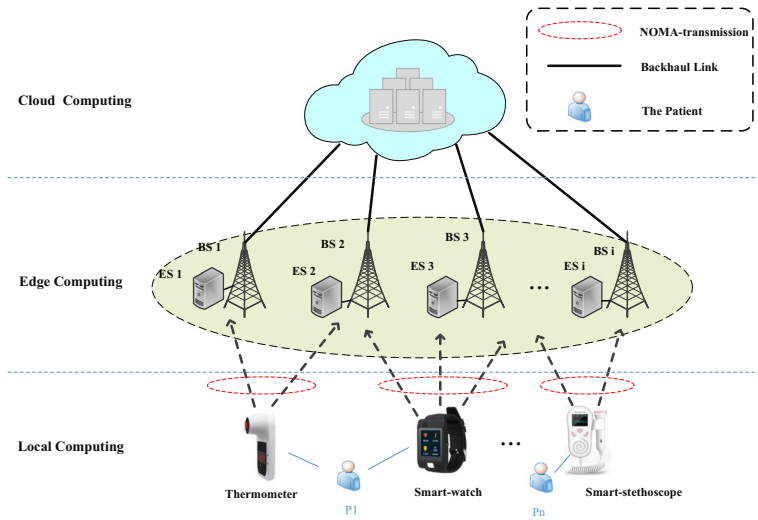

Fig. 1 The medical application system in the internet of things

$$x_n^m + y_n^m + z_n^m = 1, (n \in N, m \in M). \quad (1)$$

This constraint implies that task m of each device is either executed locally, by one of edge servers, or by the cloud server. Only one of x_n^m, y_n^m and z_n^m for device n can have the value 1 at any time.

3.2 Communication Model

We use g_{ni} to denote the channel power consumption gain from the IoTD n to the edge server i. For the sake of convenience, we assume that the edge servers in I are ordered according to:

$$g_{n1} \geq g_{n2} \geq \cdots \geq g_{ni} \quad (2)$$

When device n is connected to edge server i, let p_{ni} denote the IoTD's transmit power to edge server i. Based on (2) and the Successful Interference Cancellation (SIC), we can express the uplink NOMA-transmission rate from the IoTD n to edge server i as follows:

$$R_{ni} = W_n \log_2^{\left(1+\frac{g_{ni}p_{ni}}{g_{ni}\sum_{j=1}^{i-1}p_{nj}+W_n n_0}\right)}, \forall i \in I \quad (3)$$

where W_n denotes the IoTD n's channel bandwidth, and n_0 denotes the spectral power density of the background noise. Let γ_i denote the received signal-to-interference plus noise ratio (SINR) at edge server i, i.e.,

$$\gamma_i = \frac{g_{ni}p_{ni}}{g_{ni}\sum_{j=1}^{i-1}p_{nj} + W_n n_0}, \forall i \in I. \quad (4)$$

Then, suppose that $\{\gamma_i\}_{i\in I}$ is given. Then we obtain :

Proposition 1: The IoTD n's minimum total transmit power for reaching $\{\gamma_i\}_{i\in I}$ can be given by:

$$P_n^{tot}\left(\{\gamma_i\}_{\forall i\in I}\right) = W_n n_0 \sum_{i\in I}\left(\frac{1}{g_{ni}} - \frac{1}{g_{ni-1}}\right)\prod_{j=i}^{I}(1+\gamma_j) - \frac{W_n n_0}{g_I}, \quad (5)$$

where parameter g_0 is a large number such that $\frac{1}{g_0} = 0$.

Proof The key to the proof is based on forward deduction, which is essentially similar to that for Proposition 1 in [32]. □

Then, based on Proposition 1, we have the following important result.

Corollary 1 *Given the NOMA-transmission duration t_n and device n's offloaded computing workloads $\{L_n^m\}_{\forall m\in M}$ to different edge servers, device n's minimum transmit power (for offloaded computing workloads $\{L_n^m\}_{\forall m\in M}$ to the edge servers with duration t_n) is given by:*

$$P_n^{tot}\left(t_n, L_n^m\right) = W_n n_0 \sum_{i\in I}\left(\frac{1}{g_{ni}} - \frac{1}{g_{ni-1}}\right)2^{\frac{1}{W_n}\frac{1}{t_n}\sum_{h=m}^{M}L_n^h - \frac{W_n n_0}{g_{nI}}} \quad (6)$$

Proof Given the NOMA-transmission duration t_n and the n's offloaded workloads $\{L_n^m\}_{\forall m\in M}$, the transmission rate from device n to edge server i can be given by $R_{ni} = \frac{L_n^m}{t_n}$. Further with (3) and (4), we can obtain (6) by substituting $\gamma_i = 2^{\frac{L_n^m}{t_n W_n}} - 1$ into (5). Let P^{max} denote device n's maximum transmit power. We

impose the following constraint to ensure that the n's total transmit power for sending the offloading workloads $\left\{L_n^m\right\}_{\forall m \in M}$ to the respective edge servers cannot exceed $\mathrm{P}^{\max}$:

$$E_{\mathrm{n}}^{trans} = t_n \mathrm{P}_n^{tot}(t_n, \left\{L_n^m\right\}_{m \in M}). \tag{7}$$

□

3.3 Computing Model

3.3.1 Local computing

For tasks on each device, parts of tasks will be processed locally. To model the delay in completing IoTD n's computation requirement . we introduce f_n^l and f_{nm}^i to represent the local computing rate of device n and the computing rate of edge server i. For the sake of clear presentation, we consider f_n^l and f_{nm}^i to be in units of bits per second (i.e., $bits/s$) in this work. e.g., f_n^l can be calculated as $f_n^l = \frac{v_n}{c_n^l}$, where v_n represents the CPU frequency in Hz (i.e., cycles per second) and c_n^l represents the consumed CPU cycles per bit. If task m is processed locally, then $x_{\mathrm{n}}^m = 1, y_n^m = 0, z_n^m = 0, \Pi_n^m = \{1, 0, 0\}$ and the computation delay for completing task m of device n can be expressed as:

$$t_m^l = \frac{L_n^m}{f_n^l} \tag{8}$$

Here, we use ρ_n to denote IoTD n's CPU power consumption (in the unit of joule per second). Additionally, the energy consumption of processing task m locally can be expressed as:

$$E_m^l = \rho_n t_m^l = \rho_n \frac{L_n^m}{f_n^l} \tag{9}$$

Therefore, given the offloading decision $\Pi_n^m = \left\{x_n^m, y_n^m, z_n^m\right\}$, the total delay of tasks performed locally by the device n is:

$$T_n^l = \sum_{m=1}^{M} t_m^l x_n^m, \forall m \in M \tag{10}$$

The total energy consumption of device n for performing tasks locally is:

$$E_n^l = \sum_{m=1}^{M} E_m^l x_n^m . \forall m \in M. \tag{11}$$

3.3.2 Edge computing

Let f_{nm}^i represent the computing resource of task m assigned to device n. If task m is processed on edge server i, then $x_n^m = 0, y_n^m = 1, z_n^m = 0, \Pi_n^m = \{0, 1, 0\}$, and the computing delay of task m on edge server i is $\mathrm{t}_{\mathrm{m}}^{\mathrm{comp}} = \frac{L_n^m}{f_{nm}^i}$.

Therefore, given the offloading decision $\Pi_n^m = \left\{x_n^m, y_n^m, z_n^m\right\}$, the total delay in processing tasks at the edge node is:

$$T_n^{mec} = \left(t_n + \max_{i \in I}\left\{\frac{L_n^m}{f_{nm}^i}\right\}\right) y_n^m \tag{12}$$

3.3.3 Cloud computing

If task m is offloaded to the cloud for processing, the task is first transmitted to the edge node via NOMA, and then the edge node forwards the task to the cloud through a wired backhaul link. If the cloud performs task m, then $x_n^m = 0, y_n^m = 0, z_{\mathrm{n}}^m = 1, \Pi_n^m = \{0, 0, 1\}$. Denote the round trip time for task transmission between the edge node and the cloud server as $\mathrm{t}_c^{\mathrm{trans}}$. Let f_n^c be the cloud computation capability (in CPU cycles/s) assigned to device n. Ordinarily, the computing capability of the cloud is much higher than that of edge servers, i.e., $f_n^c >> f_{nm}^i$. The execution time of cloud processing the task is:

$$t_m^c = t_n + t_c^{trans} + \frac{L_n^m}{f_n^c}. \tag{13}$$

Therefore, given the offloading decision $\Pi_n^m = \left\{x_n^m, y_n^m, z_n^m\right\}$, the total delay in processing tasks at the cloud center is:

$$T_n^c = \sum_{m=1}^{M} t_m^c z_n^m, \forall n \in N \tag{14}$$

As in [33, 34], we ignore the downlink transmission delay of edge node and the cloud center sending the task back to the device because the data size after the task processing is usually much smaller than its size before processing.

3.4 Problem Formulation

In this study, we minimize the total system cost by jointly optimizing durations t_n and the computation offloading decision Π_n^m. Thus, we formulate the fol-

lowing optimization problem (("SCM" means "system cost minimization"):

$$SCM: V(t,\Pi) = \sum_{n\in N}\left(\sum_{m\in M} E_m^l x_n^m + E_n^{trans}\left(y_n^m + z_n^m\right) + \beta \max\left\{T_n^l, T_n^{mec}, T_n^c\right\}\right)$$

$$s.t. \max\left\{T_n^l, T_n^{mec}, T_n^c\right\} \le T_n^{\max}, \quad (15)$$

$$\mathrm{P}_n^{tot}\left(t_n, \left\{L_n^m\right\}_{m\in M}\right) \le \mathrm{P}^{\max}, \quad (16)$$

$$\mathrm{t}_n \mathrm{P}_n^{tot}\left(t_n, \left\{L_n^m\right\}_{m\in M}\right) + \rho_L \frac{L_n^m}{f_n^l} \le E_n^{\max}, \forall i \in I, m \in M, n \in N, \quad (17)$$

$$x_{\mathrm{n}}^m + y_n^m + z_n^m = 1, n \in N, m \in M, \quad (18)$$

$$x_{\mathrm{n}}^m, y_n^m, z_n^m \in \{0,1\}, n \in N, m \in M, \quad (19)$$

$$0 \le t_n \le T_n^{\max}. \quad (20)$$

Note that the objective function covers the quality of service experienced by the device in using multi-access mobile edge computing, as well as its energy consumption. Parameter β denotes the weight of the device in the total delay in completing its workloads. By changing the parameter β, we can achieve different tradeoffs between the quality experience and the energy consumption of devices.

In problem (SCM), constraint (15) ensures that the delay required by the device to complete the total workloads cannot exceed the maximum delay. Constraint (16) ensures that the total transmission power that the device consumes to send its workloads to its respective edge server cannot exceed the maximum transmission power. Constraint (17) ensures that the total energy consumption of the device transmission data and local computing cannot exceed the energy budget of the device. Constraint (18) denotes that task m is processed either on the device, on the edge server, or on the cloud server. Constraint (19) denotes whether the m task of IoTD n is processed in the device itself, the edge node or the cloud server. Constraint (20) denotes the device n's transmission time can not exceed the maximum delay allowed. Table 2 lists the important notations used in this paper.

4 Algorithm for solving problem (SCM)

In this section, because the joint optimization problem is non-convex and the problem is computationally limited by the curse of dimensionality, especially for large-scale medical devices, we focus on using the hierarchical structure of the problem and propose corresponding effective algorithms to determine the optimal offloading solution. Firstly, the key idea of solving the problem(SCM) is to use the hierarchical structure of Fig. 2. The original problem(SCM) is decomposed into two subproblems: offloading decision (P_1) and radio resource transmission duration allocation (P_2). Next, we propose a computing offloading algorithm based on deep learning and a linear search algorithm to obtain the optimal offloading decision and NOMA-transmission time, respectively. Figure 3 shows the hierarchical structure of the algorithm. Specifically, given NOMA-transmission time within the time allowable range of problem (P_2), we propose an offloading algorithm based on deep learning to determine its optimal offloading decision. Next, by adjusting the NOMA-transmission time, we propose a linear search algorithm to find the optimal solution. We emphasize that as long as both subproblems can be solved accurately, the hierarchical algorithms will enable us to optimally solve the original problem (SCM). Details are shown in the remainder of this section.

4.1 Problem Decomposition

Offloading decision (P_1): We tackle here the subproblem of optimizing the offloading decision of each task at a given NOMA transmission t_n. First, we consider the transmission duration t_n of each IoTD and aim to find the optimal offloading decision Π^* to minimize the overall system cost. Here, we assume that the duration t_n of transmitting different workloads to different edge servers is the same. Note that t_n is a decision variable in the system model. Then, problem (SCM) will induce the following subproblem (P_1):

$$(P_1): \min V_{(t_n)}$$

$$s.t. \max\left\{T_n^l, T_n^{mec}, T_n^c\right\} \le T_n^{\max}, \quad (21)$$

$$\mathrm{P}_n^{tot}\left(t_n, \left\{L_n^m\right\}_{m\in M}\right) \le \mathrm{P}^{\max}, \quad (22)$$

$$\mathrm{t}_n \mathrm{P}_n^{tot}\left(t_n, \left\{L_n^m\right\}_{m\in M}\right) + \rho_L \frac{L_n^m}{f_n^l} \le E_n^{\max}, \forall i \in I, m \in M, n \in N, \quad (23)$$

$$x_{\mathrm{n}}^m + y_n^m + z_n^m = 1, n \in N, m \in M, \quad (24)$$

$$x_{\mathrm{n}}^m, y_n^m, z_n^m \in \{0,1\}, n \in N, m \in M, \quad (25)$$

Table 2 Model notations

Notation	Definition
Q_n	Computation task of device n
L_n	Data size for all tasks of the device n
$T_n^{\max}$	The maximum delay allowed for the device n
Q_n^m	The computing task m of device n
L_n^m	Data size of the computing task m of device n
$T_m^{\max}$	The maximum delay allowed for the computing task m of device n
x_n^m	$x_n^m = 1$ if the device n process its task m locally. Otherwise, $x_n^m = 0$
y_n^m	$y_n^m = 1$ if the device n offloads its task m to the edge nodes. Otherwise, $y_n^m = 0$
z_n^m	$z_n^m = 1$ if the device n offloads its task m to the cloud server. Otherwise, $z_n^m = 0$
W_n	The device n's channel bandwidth
n_0	the spectral power density of the background noise
$P^{\max}$	The device's maximum transmit-power
t_n	The NOMA transmission-duration
$E_n^{\max}$	The device n's energy-budget
β	Weight between energy consumption and processing delay in the system cost
f_n^l	Local computing rate of the device n
f_{nm}^i	The computing rate of the edge server i
f_n^c	The cloud computation capability
t_n^{trans}	Average backhaul delay

We express the optimal solution as $V_{(t_n)}^*$. Problem (P_1) aims at finding the minimum value of $V_{(t_n)}^*$ under the given t_n. Note that different from Problem (SCM) before, the value of t_n is fixed in constraints (21), (22),and (23)in Problem (P_1).

Radio resource allocation (P_2): Here we tackle the optimization NOMA-transmission t_n. First, after obtaining $V_{(t_n)}$ by solving problem (P_1) for each given t_n, we minimize $V_{(t_n)}$ by continuing to adjust $t_{NOMA}^{cur} \leq \min\left\{T_n^{\max}, \frac{L_n}{f_n^l}, \frac{L_n^m}{f_{nm}^i}\right\}$ and find the best $V_{(t_n)}^*$, we can further solve the original problem (SCM) by solving the following subproblem:

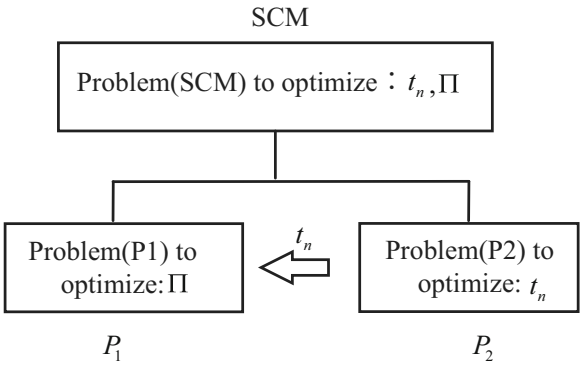

Fig. 2 Problem decomposition

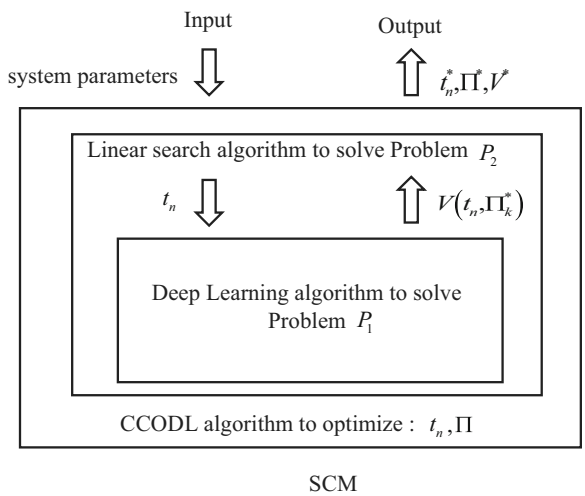

Fig. 3 Layered structure of our proposed algorithm

$$(P_2): \min V^*_{(t_n)}$$

$$s.t. 0 \leq t_n \leq T_n^{\max}. \tag{26}$$

We emphasize that the hierarchical structure of our problem will enable us to solve the original problem (SCM) optimally, as long as both the subproblem (P_1) and the subproblem (P_2) can be solved accurately. The details of the algorithm are shown in the remainder of this section.

4.1.1 Proposed algorithm to slove problem(P_1)

In this section, a task offloading algorithm based on distributed deep learning is proposed to obtain the optimal decision of the problem(P_1). Given the NOMA-transmission time, in order to solve the problem(P_1), our goal is to find the offloading policy function π to generate the best offloading decision to reduce system cost and balance the energy consumption and delay of the device.

As shown in Fig. 4, the algorithm is mainly composed of two parts: offloading decision generation and deep learning. Specifically, given a transmission duration t_n and the input task workloads of all devices, we find an offloading policy function π to generate the optimal offloading decision of the problem P_1. This policy function can be expressed as: $\Pi^* = \pi_\theta\left(L_n^m\right)$. Here, we approximately express π by a parameterized function based on DNN, and θ represents the learning parameter used in the DNN module [35]. The generation of the offloading decision relies on the use of K DNNs, which is characterized by its embedded learning parameters θ.

Specifically, given the transmission duration t_n, for each input task workload L_n^m, K DNNs are used to efficiently generate K candidate decisions. Then, the offloading action with the lowest system utility is chosen as the output, denoted as Π_k^*. Once the best offloading decision Π_k^* is obtained, we save it as a new entry of marked data $\left(L_n^m, \Pi_k^*\right)$ in a finite memory structure. When the memory is full, the oldest data entry will be discarded. After collecting a certain number of new samples, we select a batch of samples from the training memory and use them according to the experience replay technique [36, 37] to train the DNNs (i.e. update learning parameters θ) to further improve accuracy. Overall, the DNN iteratively learns from the best decision pairs $\left(L_n^m, \Pi_k^*\right)$ and generates better offloading decisions output as time progresses.

In practice, we use the experience replay technique in the proposed framework to train DNNs, which has reduced complexity than using the entire set of data samples and the random sampling fastens the convergence by reducing the correlation in the training samples. The parameters θ of the DNN are updated by applying the Adam algorithm [38] to reduce the averaged cross-entropy loss, as $L(\theta_k) = -x^{\mathrm{T}} \log \pi_{\theta_k}(L) - (1-x)^{\mathrm{T}} \log\left(1 - \pi_{\theta_k}(L)\right)$. Adam optimization algorithm is an extension of random gradient descent algorithm, which is suitable for solving optimization problems with large-scale data and parameters, and can achieve efficient computing. Adam algorithm designs independent adaptive learning rate for different parameters by computing the first-order moment estimation and second-order moment estimation of gradient.The detailed update procedure of the Adam algorithm is omitted here for brevity. Subsequently, the offloading policy is also updated and used to generate new offloading decisions.

With the learning value $\Pi^* = \pi_{\theta_k}\left(L_n^m\right)$, the corresponding optimal offloading decision is obtained.

4.1.2 Proposed Algorithm to Slove Problem(P_2)

Then, we continue to adjust $t_{NOMA}^{cur} \leq \min\left\{T_n^{\max}, \frac{L_n}{f_n^l}, \frac{L_n^m}{f_{nm}^i}\right\}$ by using linear search algorithm to find the best system cost. Compared with directly learning the whole problem(SCM) solution set, this method can effectively improve the accuracy and efficiency of the DNN module. Using the input of task data of different samples and by adjusting different transmission durations t_n, this iteration is repeated, and the learning strategy π of DNN is gradually improved.

4.2 Collaborative Computing Offloading Algorithm Based on Deep Learning(CCODL)

In this section, a collaborative computing offloading algorithm based on distributed deep learning is proposed to achieve efficient offloading. The CCODL algorithm can run independently on the local device, and the local device will decide whether each task is executed locally, at the edge node or on the cloud server. our CCODL algorithm implements K DNNs to

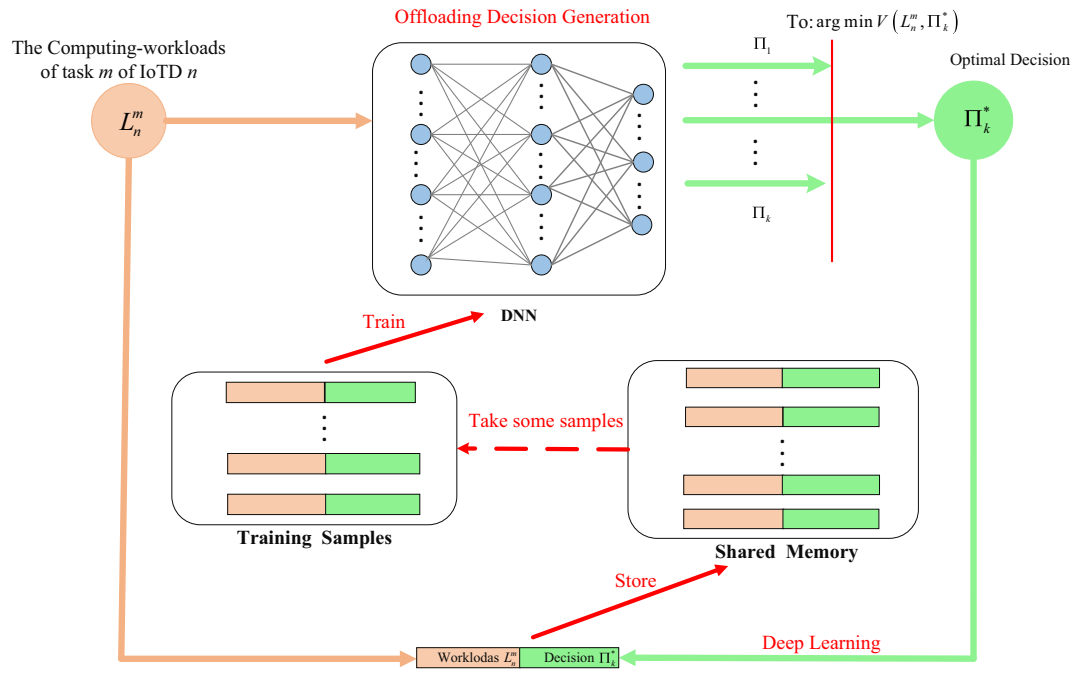

Fig. 4 Architecture of deep learning-based offloading algorithm

learn the optimal offloading decision according to different task workloads using historical experience. Our goal is to train DNNs to learn the best offloading decision (according to the experience data) so as to balance devices' energy consumption and delay.

The details of the CCODL algorithm are shown in Algorithm 1. It is implemented in TensorFlow [39]. The steps are detailed below:

1. Initialize K DNNs with random learning parameter values. The memory is initially empty. By selecting a $K >= 2$, the algorithm can converge to a better offloading decision. Given a current NOMA transmission time $t_n^{cur} = \Delta$, set a small step $\Delta = 1s$.
2. Repeatedly input different task data sizes into K DNNs. For each input data, based on subproblem P_1, we can gain the optimal offloading decision $\Pi_k^* = \arg\min V_{(t_n^{cur})}^*$ from K offloading decisions.
3. Obtain the best offloading decision Π_k^*. We save it as a new entry of marked data $\left(L_n^m, \Pi_k^*\right)$ in a finite memory structure. When this memory is full, the oldest data entry will be discarded. These generated tag data are then used to train all K DNNs and to update the learning parameter value θ. Finally, we find the global optimal solution giving the current time t_n.
4. Until now, we can use the CCODL algorithm to find any $V_{(t_n)}^*$. At this point, we proceed to solve the optimal transmission time, which will lead to the lowest system cost. Note that our subproblem (P_2) involves only a single variable $t_n^{cur} \leq \min\left\{T_n^{\max}, \frac{L_n}{f_n^l}, \frac{L_n^m}{f_{nm}^i}\right\}$. Therefore, one method to solve the subproblem is to use a small step length to perform a linear search.

5 Performance Evaluation

5.1 Simulation Settings

In this section, we use numerical results to verify the convergence of our proposed algorithm and show the performance advantages of our proposed NOMA-enabled multiple access computing offloading. For the simulations, we built the network model and implemented the proposed algorithm in Python-3.6 together with TensorFlow-1.8.0 and Pandas-0.24 on a server powered by an Intel Core i5-10400F CPU and 16 GB of memory. The network topology

Algorithm 1 CCODL algorithm.

Input:

Input different workloads L_n^m at time t

Output:

Optimal offloading decision Π_k^* at time t, optimal NOMA-transmission t_n^*, and the best system cost $V_{(t_n^*)}^*$.

Initialization of the K DNNs with random parameters $\theta_k, k \in K$ and emptying of the memory structure.
Set step-size $\Delta = 1s$ as a small number. Set $CBV = \infty$ and $CBS = \emptyset$.
Set the NOMA-transmission duration $t_n^{cur} = \Delta$.
for $t = 1, 2, ..., G$ **do**
  Replicate different workloads L_t to all K DNNs.
  Generate the k-th offloading strategy Π_k from the k-th DNN in a parallel way as $\Pi_k = \pi_{\theta_{k,t}}\left(L_n^m\right)$
  while $t_n^{cur} \leq \min\left\{T_n^{\max}, \frac{L_n}{f_n^l}, \frac{L_n^m}{f_{nm}^i}\right\}$ **do**
    Compute $V_{(t_n^{cur})}^*$ for all $\{\Pi_k\}_{k \in K}$ by solving (P_1).
    if $V_{(t_n^{cur})}^* < CBV$ **then**
      Set $CBV = V_{(t_n^{\text{cur}})}^*$ and $CBS = t_n^{cur}$.
      Update $t_n^{cur} = t_n^{cur} + \Delta$.
    end if
  end while
  $V_{(t_n^{cur})}^* = CBV, t_n^* = CBS$.
  Select the best offloading decision as the output $\Pi_k^* = \arg\min V_{(t_n^{cur})}^*$
  Store $\left(L_n^m, \Pi_k^*\right)$ into the memory structure
  Randomly Sample K batches of training data from the memory structure
  Train the DNNs and update $\theta_{k,t}$
  Select the best NOMA-transmission duration t_n^* when the offloading decision Π_k^* is present.
end for

consists of the $1000m \times 1000m$ square area with a cloud server, 3 edge servers and 3 IoTDs. The cloud center is located in the network center with coordinates (0,0), and 3 edge servers, e.g., each edge server in a warehouse, has a coverage radius of $500m$. In addition, IoTDs are randomly distributed around edge servers and the channel power gains from the IoTD n to the edge servers are generated according to the distance model of [40], the random channel power gains used here are $\{g_{ni}\}_{n \in N, i \in I} = \left\{1.8185 \times 10^{-7}, 1.7793 \times 10^{-7}, 1.7793 \times 10^{-7}\right\}$. We set $T_n^{\max} = 12s, P^{\max} = 20W$, and $E^{\max} = 20Joul$. We adopt a fully connected DNN consisting of one input layer, two hidden layers, and one output layer. We set training interval as 10, training batch size as 128, memory size as 1024, and learning rate for Adam optimizer as 0.01. The relevant additional parameters used in the simulation experiment are summarized in Table 3. To evaluate the performance of the proposed algorithm, we compared it with several existing schemes. Details are as follows.

5.2 Algorithm Convergence

Figures 5 and 6 depict the convergence of the proposed algorithm with various DNN learning rates and transmission time t_n. From Fig. 5, it can see that the system cost decreases as the number of learning steps increases until it converges within a specific range. This is because the raw task data of different samples are inputted directly for training without any quantization. In addition, the higher the learning rate is, the faster the convergence rate of CCODL. However, when the learning rate increases, we are likely to obtain the local optimal solution rather than the global optimal solution. Therefore, we choose an appropriate learning rate according to the specific situation.

Table 3 The parameter settings of simulation experiment

Parameter	Value
The number of IoTDs /N	3
The number of Tasks /M	9
The data size of task /$Mbits$	[10,30]$Mbits$
The computational capability of IoTDs /f_n^l	$1\sim 5Mbits/s$
The computational capability of ESs i /f_{nm}^i	$10\sim 20Mbits/s$
The computational capability of Cloud /f_n^c	$150Mbits/s$
The task completion deadline /$T_n^{\max}$	$12s$
Channel Bandwidth /W_n	$10\sim 20MHz$
Average backhaul delay /t_c^{trans}	$5s$
IoT device's CPU power consumption /ρ_n	$0.5Joul/s$
The spectral power density of the background noise/n_0	10^{-10}
Weight of energy consumption and task completion /β	$1.5J/s$

Figure 6 shows the convergence performance of our algorithm at different transmission durations t_n. We observe that the convergence speed of the algorithm is not linear with the NOMA-transmission t_n. Specifically, When $t_n < 3s$, the devices need to use a large transmit-power for sending the offloaded workloads to edge servers or the cloud center. As a result, only a small part of the devices' computation requirements can be offloaded. On the other hand, using a very large $t_n > 3s$ will lead to a significant delay in the NOMA-transmission, and thus the system cost is again very large. So we should choose the optimal transmission duration $t_n = 3s$.

Figure 7 shows the impact of NOMA-transmission duration on system cost under different bandwidths. The optimal t_n is different under different bandwidths. The basic principle of solving this subproblem is now explained. In this algorithm, we perform linear search on the pair in a small step $\Delta = 1s$. Specifically, we observe the system cost and their convergence rate

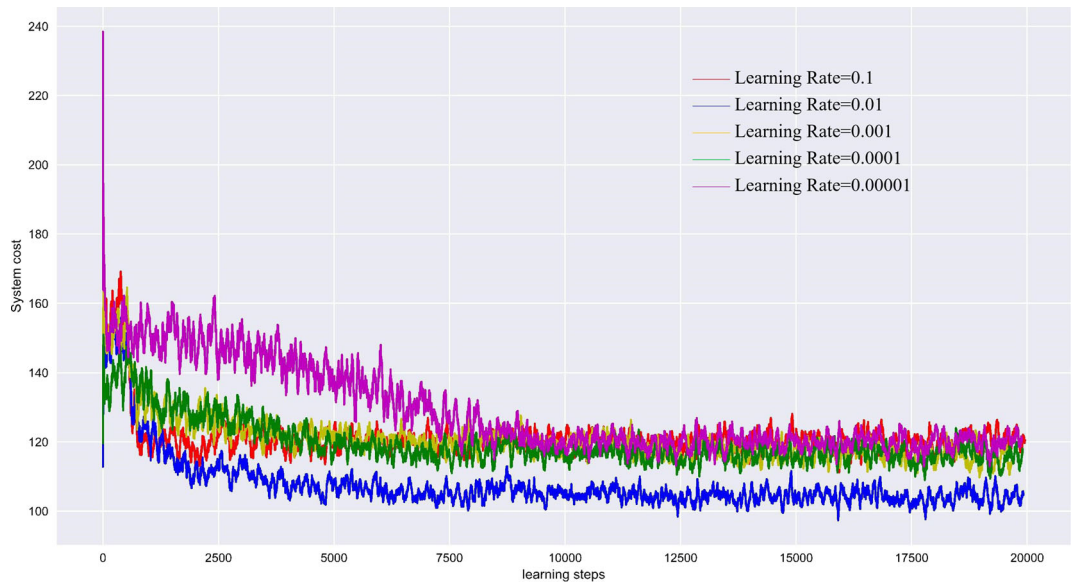

Fig. 5 Convergence performance under different learning rates

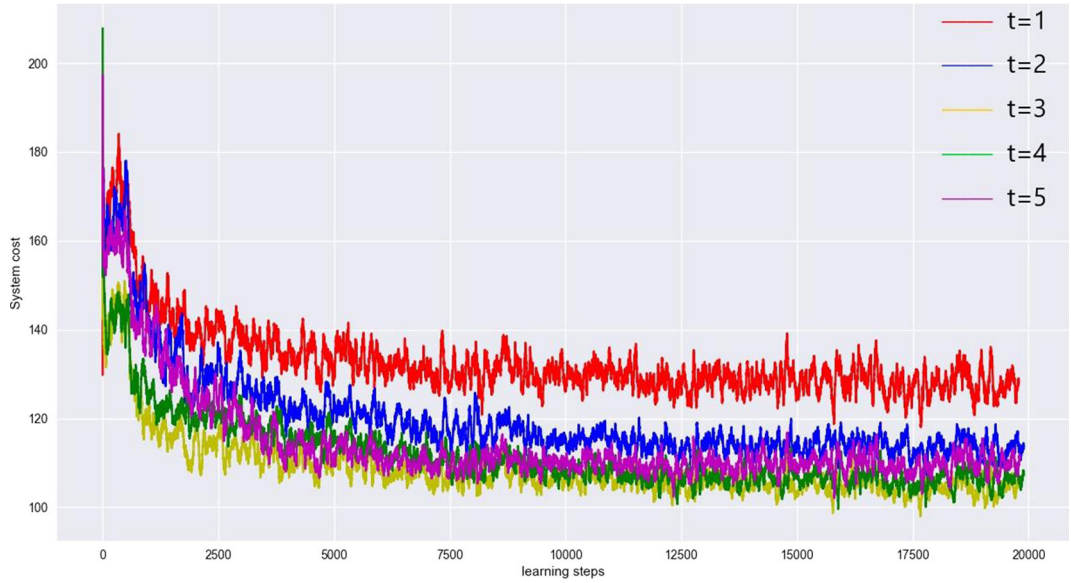

Fig. 6 System cost versus learning steps under different t

under different t_n. A too large or too small t_n will not be beneficial to minimize the total cost of the system. Therefore, we need to select the optimal to send the offloaded tasks to edge servers and the cloud server under different bandwidths.

5.3 Performance Comparisons

In the experiment, we compare the performance of cloud-edge-end collaborative offloading based on the frequency division multiple access (FDMA) scheme with the other two offloading schemes to establish the effectiveness of our proposed collaborative offloading framework.

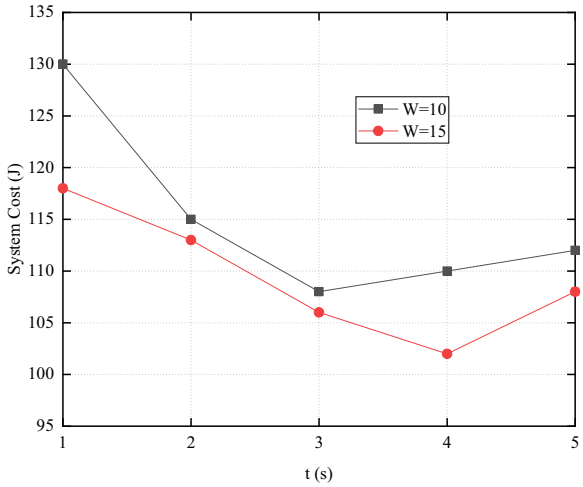

Fig. 7 System cost versus t under different W_n

Cloud-edge-end: Based on our proposed algorithm using FDMA, the tasks to be processed on each device simultaneously will be processed locally, edge servers or the cloud server.

Local-only scheme: Based on our proposed algorithm using FDMA, all tasks on each device are processed locally on the device.

Edge-only scheme: Based on our proposed algorithm using FDMA, all tasks will be offloaded to the edge servers for processing.

As shown in Fig. 8, when the algorithm reaches convergence, it can observe that the system cost of cloud-edge-end collaborative network is the lowest compared to the other schemes. When all tasks are processed locally, the computing power of the device is limited, which will lead to the increased energy consumption of the devices. It is shown that the proposed algorithm stably converges to its optimal value, providing approximately 25.2% and 79.2% lower system cost than schemes such as only using edge computing and fully local processing. Therefore, this shows the effectiveness of our proposed collaborative framework.

Next, we will show the performance advantages of the proposed NOMA-enabled multi-access computa-

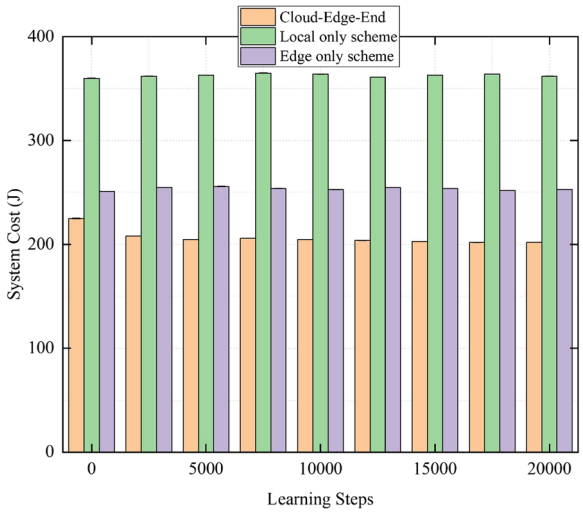

Fig. 8 System cost versus learning steps under $W_n = 15MHz$

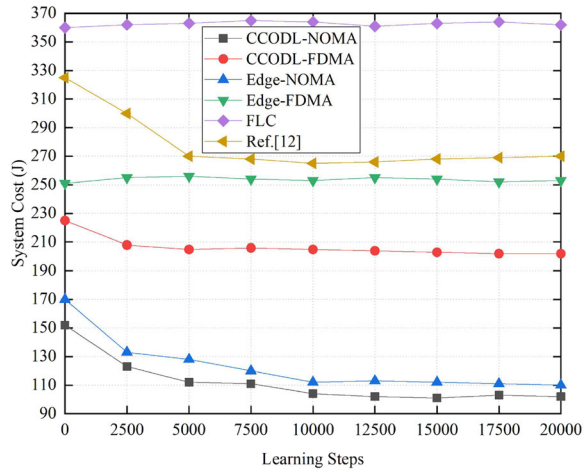

Fig. 9 System cost versus learning steps under different algorithms

tion offloading. To do so, we compared it with several existing schemes. Details are as follows:

Proposed scheme (CCODL algorithm): The offloading tasks of the device are optimally partitioned locally, to the edge server and the cloud according to our proposed algorithm using NOMA (CCODL-NOMA) and FDMA (CCODL-FDMA).

Edge on scheme: Based on our proposed algorithm, all tasks are offloaded to the edge using NOMA (Edge-NOMA) and FDMA (Edge-FDMA).

Fully local computing (FLC): All the computation tasks of devices are executed locally.

As shown in Fig. 9, when the algorithm reaches convergence, we can clearly see that the system cost of CCODL-NOMA is the lowest compared with the algorithm in [12] and other schemes, From the figure, we can see that the total system cost overhead with CCODL-NOMA is approximately 93.4%, 9.8%, and 253.9% less than those with CCODL-FDMA, Edge-NOMA and FLC, respectively. This shows the effectiveness of our proposed algorithm.

In order to analyze the influence of the weighting factor β of energy consumption and task completion on the system cost, we compare the changes of system cost by changing the value of the weighting factor β, and the results are shown in the Fig. 10. As illustrated in the figure, as β increases, the total system cost of all schemes increases. The larger the weighting factor, the more attention is paid to the delay, and more computing resources are needed for processing, so the energy consumption is higher. However, its improvement will significantly reduce the execution time of computing tasks and greatly reduce the delay of entire computing offloading.Therefore, the weight factor is adjusted for different scenarios to obtain the lowest system cost. For example, for some applications that are more sensitive to delay, it is possible to appropriately increase energy consumption in exchange for lower delay to ensure the quality of user experience. In addtion, our proposed algorithm outperforms [12] and other schemes.

Finally, we compare the performance of our proposed algorithm with FDMA under different W_n. Specifically, in the FDMA-based scheme, IoTD

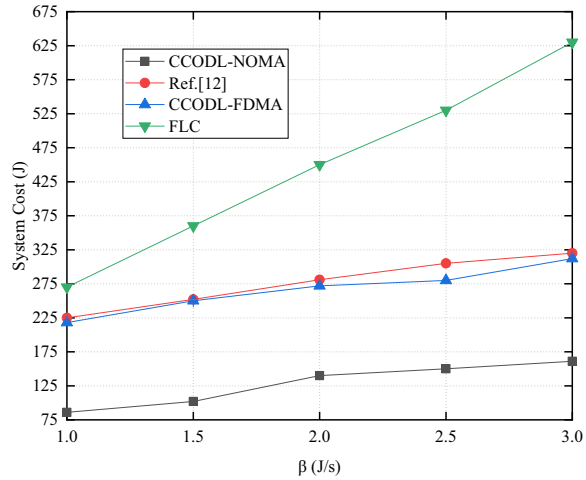

Fig. 10 System cost under different β for different offloading algorithms

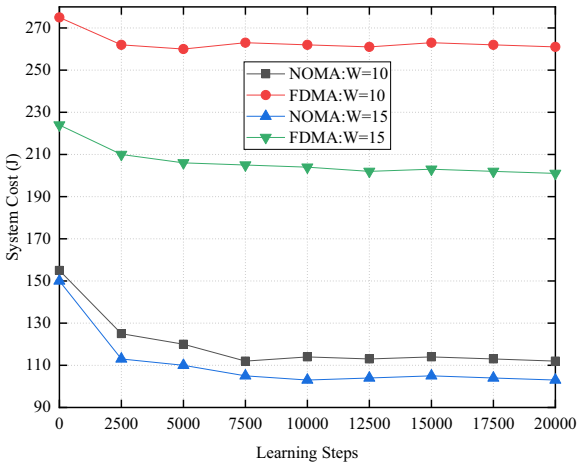

Fig. 11 Comparison between our NOMA-enabled computation offloading scheme and the FDMA-enabled offloading scheme

divides its total bandwidth W_n into I subchannels (the bandwidth of each subchannel is $\frac{w_n}{I}$). Therefore, the IoTD uses a subchannel to send its offloaded workloads to each edge server in parallel. However, the FDMA-based scheme produces a low spectral efficiency in comparison with the NOMA-enabled scheme. Specifically, we tested two cases: $W = 10MHz$ and $W = 15MHz$. For the sake of clear comparison, our algorithm has basically converged when the learning step reaches 2000, and we have obtained the system cost for the NOMA-based scheme and the FDMA-based scheme in each test case. As shown in Fig. 11, our NOMA-based scheme is always better than the FDMA-based scheme; e.g., when $W_n = 15MHz$, we can see that the system cost overhead with CCODL-NOMA is approximately 95.1% less than that with CCODL-FDMA, which demonstrates the effectiveness of NOMA over FDMA in IoMT. In addition, the results show that the system cost will be relatively low when the bandwidth increases.

6 Conclusion

In this paper, we integrate cloud-edge-end computing network and NOMA to propose a novel network communication model, which can realize the balance between high resource consumption and high communication cost in the process of task offloading in IoMT. It can provide medical users with energy-efficient and low-delay services according to different medical service levels and performance requirements. Specifically, considering the energy consumption, transmission delay, and quality of service, we jointly optimize the computing offloading decision and its radio resource allocations for NOMA-transmission to reduce the system cost (the weighted sum of consumed energy and delay) on IoMT of cloud-edge-end supported by NOMA. In addition, although the joint optimization problem is non-convex, we use its hierarchical structure to propose a collaborative computing offloading algorithm based on deep learning to find the optimal offloading strategy. This strategy can directly offload the real-time services of medical terminal devices to edge nodes and cloud servers for processing. Numerical results show that compared with the traditional orthogonal multiple access algorithm, our proposed algorithm has reliable convergence and near-optimal performance in reducing energy consumption and task completion delay.

However, there are still some deficiencies and a lack of consideration in the scheme. For example, the scheme does not consider real-time offloading in a dynamic environment. In future research work, we will consider the mobility of users, such as the sudden shutdown of user equipment or the addition of new equipment.

Acknowledgements This research was supported by the following projects:The National Natural Science Foundation of China (61972073), the Key Program of Natural Science Foundation of Hebei Province of China (F2019201290), the Natural Science Foundation of Hebei Province of China (F2018201153).

Data Availability Statement The raw/processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Habibzadeh, H., Dinesh, K., Shishvan, O.R., Boggiodandry, A., Sharma, G., Soyata, T.: A survey of healthcare internet of things (hiot): A clinical perspective. IEEE Internet Things J **7**(1), 53–71 (2020)

2. Arzo, S.T., Naiga, C., Granelli, F., Bassoli, R., Devetsikiotis, M., Fitzek, F.H.P.: A theoretical discussion and survey of network automation for iot: Challenges and opportunity. IEEE Internet Things J **8**(15), 12021–12045 (2021)
3. Qiu, Y., Zhang, H., Long, K.: Computation offloading and wireless resource management for healthcare monitoring in fog-computing-based internet of medical things. IEEE Internet Things J **8**(21), 15875–15883 (2021)
4. Park, C., Lee, J.: Mobile edge computing-enabled heterogeneous networks. IEEE Trans. Wirel. Commun. **20**(2), 1038–1051 (2021)
5. Ding, Y., Li, K., Liu, C., Li, K.: A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing. IEEE Trans. Parallel Distributed Syst. **33**(6), 1503–1519 (2022)
6. Kai, C., Zhou, H., Yi, Y., Huang, W.: Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability. IEEE Trans. Cogn. Commun. Netw. **7**(2), 624–634 (2021)
7. Ren, J., Yu, G., He, Y., Li, G.Y.: Collaborative cloud and edge computing for latency minimization. IEEE. Trans Veh. Technol. **68**(5), 5031–5044 (2019)
8. Sheng, M., Wang, Y., Wang, X., Li, J.: Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server. IEEE Trans. Commun. **68**(3), 1524–1537 (2020)
9. Alamu, O., Iyaomolere, B., Abdulrahman, A.: An overview of massive MIMO localization techniques in wireless cellular networks: Recent advances and outlook. Ad Hoc Networks **111**, 102353 (2021)
10. Ning, Z., Dong, P., Kong, X., Xia, F.: A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things. IEEE Internet Things J **6**(3), 4804–4814 (2019)
11. Subramaniam, E.V.D., Krishnasamy, V.: A novel energy estimation model for constraint based task offloading in mobile cloud computing. J. Ambient Intell. Humaniz. Comput. **11**(11), 5477–5486 (2020)
12. Huang, L., Feng, X., Feng, A., Huang, Y., Qian, L.P.: Distributed deep learning-based offloading for mobile edge computing networks. Mobile Networks and Applications, pp. 1–8. https://doi.org/10.1007/s11036-018-1177-x (2018)
13. Bi, J., Yuan, H., Duanmu, S., Zhou, M., Abusorrah, A.: Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization. IEEE Internet Things J **8**(5), 3774–3785 (2021)
14. Hu, D., Huang, G., Tang, D., Zhao, S., Zheng, H.: Joint task offloading and computation in cooperative multicarrier relaying-based mobile-edge computing systems. IEEE Internet Things J **8**(14), 11487–11502 (2021)
15. Yang, G., Hou, L., He, X., He, D., Chan, S., Guizani, M.: Offloading time optimization via markov decision process in mobile-edge computing. IEEE Internet Things J **8**(4), 2483–2493 (2021)
16. Ale, L., Zhang, N., Fang, X., Chen, X., Wu, S., Li, L.: Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning. IEEE Trans. Cogn. Commun. Netw. **7**(3), 881–892 (2021)
17. Zhou, S., Jadoon, W.: The partial computation offloading strategy based on game theory for multi-user in mobile edge computing environment. Comput. Networks **178**, 107334 (2020)
18. Huang, L., Bi, S., Zhang, Y.A.: Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Trans. Mob. Comput. **19**(11), 2581–2593 (2020)
19. Song, Z., Liu, Y., Sun, X.: Joint task offloading and resource allocation for noma-enabled multi-access mobile edge computing. IEEE Trans. Commun. **69**(3), 1548–1564 (2021)
20. Liu, B., Liu, C., Peng, M.: Resource allocation for energy-efficient MEC in noma-enabled massive iot networks. IEEE J. Sel. Areas Commun. **39**(4), 1015–1027 (2021)
21. Du, J., Liu, W., Lu, G., Jiang, J., Zhai, D., Yu, F.R., Ding, Z.: When mobile-edge computing (MEC) meets nonorthogonal multiple access (NOMA) for the internet of things (iot): System design and optimization. IEEE Internet Things J **8**(10), 7849–7862 (2021)
22. Qian, L., Wu, Y., Jiang, F., Yu, N., Lin, B.: Noma assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial internet of things. IEEE Transactions on Industrial Informatics **PP**(99), 1–1 (2020)
23. Fang, F., Xu, Y., Ding, Z., Shen, C., Peng, M., Karagiannidis, G.K.: Optimal resource allocation for delay minimization in NOMA-MEC networks. IEEE Trans. Commun. **68**(12), 7867–7881 (2020)
24. Wu, Y., Qian, L.P., Ni, K., Zhang, C., Shen, X.: Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading. IEEE J. Sel. Top. Signal Process. **13**(3), 392–407 (2019)
25. Wu, Y., Ni, K., Zhang, C., Qian, L.P., Tsang, D.H.K.: Noma-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation. IEEE Trans. Veh. Technol. **67**(12), 12244–12258 (2018)
26. Yang, L., Guo, S., Yi, L., Wang, Q., Yang, Y.: NOSCM: A novel offloading strategy for noma-enabled hierarchical small cell mobile-edge computing. IEEE Internet Things J **8**(10), 8107–8118 (2021)
27. Tuong, V., Truong, T.P., Nguyen, T., Noh, W., Cho, S.: Partial computation offloading in noma-assisted mobile-edge computing systems using deep reinforcement learning. IEEE Internet Things J **8**(17), 13196–13208 (2021)
28. Thai, M., Lin, Y., Lai, Y., Chien, H.: Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading. IEEE Trans. Netw. Serv. Manag. **17**(1), 227–238 (2020)
29. Wang, Y., Tao, X., Zhang, X., Zhang, P., Hou, Y.T.: Cooperative task offloading in three-tier mobile computing networks: An ADMM framework. IEEE Trans. Veh. Technol. **68**(3), 2763–2776 (2019)
30. Sun, C., Li, H., Li, X., Wen, J., Xiong, Q., Wang, X., Leung, V.C.M.: Task offloading for end-edge-cloud orchestrated computing in mobile networks. In: 2020 IEEE Wireless Communications and Networking Conference, WCNC 2020, Seoul, Korea (South), May 25-28, 2020, IEEE, pp. 1–6 (2020)

31. Ren, H., Liu, K., Dai, P., Li, Y., Xie, R., Guo, S.: Adaptive task scheduling via end-edge-cloud cooperation in vehicular networks. In: Yu, D., Dressler, F., Yu, J. (eds.) Wireless Algorithms, Systems, and Applications - 15th International Conference, WASA 2020, Qingdao, China, September 13-15, 2020, Proceedings, Part I, vol. 12384 of Lecture Notes in Computer Science, Springer, pp. 407–419 (2020)
32. Yuan, W., Li, P., Qian, H., Mao, X., Yang, Bo, H.: Optimal power allocation and scheduling for non-orthogonal multiple access relay-assisted networks. IEEE Transactions on Mobile Computing (2018)
33. Du, J., Zhao, L., Jie, F., Chu, X.: Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. IEEE Trans. Commun. **66**(4), 1594–1608 (2018)
34. Chen, M., Hao, Y.: Task offloading for mobile edge computing in software defined ultra-dense network. IEEE Journal on Selected Areas in Communications, pp 1–1 (2018)
35. Benditkis, D., Keren, A., Mor-Yosef, L., Avidor, T., Shoham, N., Tal-Israel, N.: Distributed deep neural network training on edge devices. In: Chen, S., Onishi, R., Ananthanarayanan, G., Li, Q. (eds.) Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, SEC 2019, Arlington, Virginia, USA, November 7-9, 2019, pp. 304–306. ACM (2019)
36. Tao, X., Hafid, A.S.: Deepsensing: A novel mobile crowdsensing framework with double deep q-network and prioritized experience replay. IEEE Internet Things J **7**(12), 11547–11558 (2020)
37. Cha, H., Park, J., Kim, H., Bennis, M., Kim, S.: Proxy experience replay: Federated distillation for distributed reinforcement learning. IEEE Intell. Syst. **35**(4), 94–101 (2020)
38. Jais, I.K.M., Ismail, A.R., Nisa, S.Q.: Adam optimization algorithm for wide and deep neural network. Knowl. Eng. Data Sci. **2**(1), 41–46 (2019)
39. Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., Bileschi, S., Terry, M., Nicholson, C., Gupta, S.N., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viégas, F.B., Wattenberg, M.: Tensorflow.Js: Machine Learning for the Web and Beyond. In: Talwalkar, A., Smith, V., Zaharia, M. (eds.) Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019 (2019). mlsys.org
40. Rui, Z.: Optimal dynamic resource allocation for multi-antenna broadcasting with heterogeneous delay-constrained traffic. IEEE Journal of Selected Topics in Signal Processing **2**(2), 243–255 (2008)