Collaborative Cloud-Edge Computing with Mixed Wireless and Wired Backhaul Links: Joint Task Offloading and Resource Allocation

Daqing Zhang and Haifeng Sun(✉)

School of Computer Science and Technology, Southwest University of Science and Technology, Mianyang 621010, China
dr_hfsun@163.com

Abstract. Mobile Edge Computing (MEC) is a promising technology that provides computing services at the edge of wireless networks to reduce the latency and the energy consumption for Smart Mobile Devices (SMDs). Additionally, the Ultra-Dense Network (UDN) will play a key role in providing high transmission capacity for SMDs in 5G networks. In order to improve the edge cloud efficiency within limited communication and computing resources, this paper proposes a joint task offloading and resource allocation scheme collaborated between cloud computing and edge computing in the UDN. Since wireless backhaul is more economical than expensive wired backhaul deployments, we consider the mixed deployment of either wired or wireless backhaul between each Small Base Station (SBS) and the Macro Base Station (MBS) in UDN scenarios, then formulate an optimization problem to minimize the system-wide computation overhead, and apply the Linear Decreasing Weight Particle Swarm Optimization (LDWPSO) algorithm to solve the problem. Numerical experiments validate the effectiveness of our proposed scheme compared to other baseline schemes.

Keywords: Mobile Edge Computing · Ultra-Dense Network · task offloading · resource allocation · wireless backhaul

1 Introduction

With the rapid development of mobile communications and the Internet, the fifth generation (5G) communication technology has now entered the stage of full commercial deployment. The explosive growth of mobile data traffic, coupled with computation-intensive and delay-sensitive applications like augmented reality, face recognition and interactive games, poses a significant challenge for smart mobile devices (SMDs) with limited computation resources and battery capacity [1]. Traditionally, remote cloud centers have been relied upon to offload

This work was supported in part by NSFC of China under Grant 62261051.

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 561, pp. 403–420, 2024.
https://doi.org/10.1007/978-3-031-54521-4_22

resource-intensive applications from SMDs due to their high computing and storage capabilities. However, numerous SMDs and tremendous traffic load can result in network congestion and high latency. To address these challenges, the European Telecommunications Standards Institute (ETSI) proposed Mobile Edge Computing (MEC) to deploy cloud computing services at the edge of the mobile access network, allowing mobile devices to efficiently offload computing tasks to MEC servers deployed at wireless access points or base stations for quick response [4].

While MEC effectively reduces traffic pressure on the core network, the computing and storage capabilities of MEC become the main bottleneck compared to cloud computing. To address this challenge, a layered collaborative cloud-edge architecture can be adopted, where non-computing-intensive tasks are processed at edge servers to achieve high energy efficiency and low latency, while computing-intensive tasks are offloaded to cloud servers to utilize richer computing resources. Effective collaboration between the cloud and the edge is crucial for improving system performance [11].

The Ultra-Dense Network (UDN) is a dense network deployment that can provide wide-area coverage of local hotspots and enhance system capacity to achieve low latency and high reliability [7]. In the UDN, a large number of low-cost and low-power Small Base Stations (SBSs) for better wireless access links are deployed. Computing tasks generated from SMDs can be transmitted to the SBS through wireless access links, and continue to be transmitted to the Macro Base Station (MBS) integrated with an MEC server through either wired or wireless backhaul links. While wired backhaul links usually offer higher reliability and data transmission rates compared to wireless backhaul links, practical considerations such as difficulty in deployment and high maintenance costs must also be taken into account. Therefore, the choice of backhaul links depends on a variety of factors including the service requirements of mobile users, traffic load intensity, and the cost of building the backhaul links [10]. In scenarios where installing fiber optic backhaul links is not feasible, a mixed wired and wireless backhaul links approach can be employed, enabling the SBS to receive and send data traffic using the wired or wireless connection.

In this paper, we combine the layered architecture of cloud computing and edge computing as a collaborative cloud-edge system in the UDN, taking into account wired or wireless backhaul links between the SBSs and the MBS, and comprehensively consider factors affecting system performance, such as offloading decisions, bandwidth allocation, computing resource allocation, and transmission power allocation, to achieve optimal computation overhead in terms of both energy consumption and computing delay. We first establish a system model and introduce a collaborative offloading process between cloud computing and edge computing in the UDN, then formulate the energy consumption and the latency of each phase of the process on the local SMD, the SBS, and the MBS, respectively. Finally, we model the offloading problem as an optimization problem to minimize the system-wide computation overhead and solve the problem by applying the Linear Decreasing Weight Particle Swarm Optimization (LDW-PSO) algorithm. In summary, this paper makes the following contributions:

1) We study a collaborative cloud-edge system in the UDN with wide-area coverage, in which the SBSs provide relay services between SMDs and the MBS integrated with an MEC server, while the cloud servers are deployed at the cloud center for computing offloading services. Mixed wired and wireless backhaul links are deployed between the SBSs and the MBS, and the available spectrum is shared between the wireless access links and the wireless backhaul links.
2) Considering the varying performance and energy requirements of different SMDs, this paper models the computation overhead of each SMD as the weighted sum of energy consumption and computing delay, then proposes a joint optimization problem related to offloading decisions, bandwidth allocation, computing resource allocation, and transmission power allocation to minimize the system-wide computation overhead.
3) By applying the LDWPSO algorithm, we solve the non-convex optimization problem in our proposed collaborative cloud-edge scheme. Subsequently, we evaluate the scheme's performance against other baseline schemes through numerical simulations, which indicate that our proposed scheme outperforms the baseline schemes.

The rest of the paper is organized as follows. In Sect. 2, we review the related work. Section 3 describes the system model. In Sect. 4, we formulate the considered optimization problem. Section 5 describes the LDWPSO algorithm applied for solving the proposed problem. Section 6 performs the numerical experiments. We conclude our work in Sect. 7.

2 Related Work

In recent years, researchers have made significant strides in solving the problem of computing offloading in MEC, in order to allow users to fully experience the performance benefits of MEC in 5G networks. MEC is a versatile technology that can be applied to various rich application scenarios, and it can effectively overcome the limitations of centralized cloud computing. Optimizing task offloading and resource allocation has become a key area of research for both academia and industry.

Aiming at the task offloading and resource allocation problems of MEC, there have been many related studies. Tran *et al.* decomposed the resource allocation problem for multi-user and multi-server MEC systems into convex and quasi-convex problems, by using the KKT condition and the dichotomy method to solve it, maximizing the weighted sum of all users' offloading benefits [13]. Furthermore, The academic community has also conducted extensive research on collaborative offloading of MEC and cloud computing systems. Ren *et al.* studied the joint optimization of computing and communication resources for MEC and cloud computing systems, and formulated a multi-user, multi-relay, and multi-server scenario, aiming to minimize system delay [11]. In addition, in MEC networks, computing collaboration is considered as a technique to further improve users' quality of experience (QoE), and has been extensively studied

by the academic community recently. Cao *et al.* proposed an offloading method for communication and computing collaboration based on MEC systems, and suggested a four-slot transmission protocol to improve energy efficiency under delay constraints [2]. The authors in [16] proposed a three-stage Time Division Multiple Access (TDMA) protocol to minimize delay in multi-assistant scenarios, and solved the relaxed convex problem using the Lagrangian dual decomposition technique with the ellipsoid method.

In the field of task offloading and resource optimization for the UDN combined with MEC, several related studies have been conducted. However, when a large number of users choose to offload tasks, bandwidth constraints severely limit the scalability of MEC. To address this problem, the authors in [8] jointly optimized offloading decisions, transmission power allocation, and sub-channel allocation to minimize the energy consumption, utilizing the coordinate descent method to update offloading decisions and allocating sub-channels with the Hungarian and greedy algorithms. In [10], the authors introduced a new wireless backhaul framework to achieve dense cell deployment and efficient data transmission, but did not dynamically manage limited transmission power and computing resources of local devices. Yang *et al.* considered a multi-cell MEC system based on Non-Orthogonal Multiple Access (NOMA) and proposed a hybrid genetic hill-climbing algorithm to minimize the weighted sum of energy consumption and delay, but they did not optimize communication and computing resources [17].

This paper differs from the aforementioned literature by studying a scenario that involves collaborative cloud-edge computing in the UDN and covers mixed wired and wireless backhaul. We analyze the joint optimization problem of offloading decisions, bandwidth allocation, computing resource allocation, and transmission power allocation and propose a solution using the LDWPSO algorithm to determine the optimal strategy.

3 System Model

The UDN under consideration in this paper consists of one MBS equipped with an MEC server and S SBSs of the same type, as shown in Fig. 1. We assume that each SMD has already been associated with a SBS through user association strategies [9]. To facilitate the analysis, the link between the SMD and the SBS is called the wireless access link, while the wired and wireless links between SBSs and the MBS are respectively referred to as the wired backhaul link and the wireless backhaul link, and the link between the MBS and the cloud is referred to as the backhaul link. The set of all SBSs is denoted as $\mathbb{S} = \{1, 2, \ldots, S\}$. Let $\mathbb{S}_0 = \{1, \ldots, M\}$ and $\mathbb{S}_1 = \{M + 1, \ldots, S\}$ represent the subsets of SBSs connected to the wireless backhaul links and the wired backhaul links, respectively, and M represent the total number of SBSs accessing the wireless backhaul link. Suppose U_s SMDs locate in the s-th SBS cell, let $\mathbb{U}_s = \{1, 2, \ldots, U_s\}, s \in \mathbb{S}$ denote the set of SMDs, and $u_{s,k} \in \mathbb{U}_s$ represent the k-th SMD. The computing resources of the MEC server and the cloud server can be allocated to SMDs through virtual machine technology.

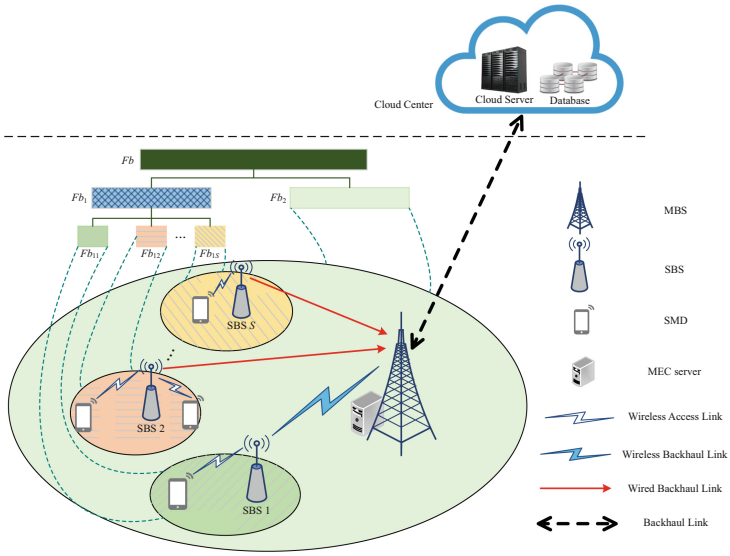

Fig. 1. System model.

Assuming that the SBSs and the MBS share the same frequency spectrum (as shown in Fig. 1), the system's total frequency band Fb is divided into two parts: Fb_1 and Fb_2. Fb_1 is allocated for SBSs, and Fb_2 is allocated for the MBS. Notably, Fb_1 is equally allocated to each SBS, then each SBS's frequency band is evenly allocated to its associated SMDs, while the MBS's frequency band is evenly allocated to the connected SBSs by the wireless backhaul links. The widths of frequency bands Fb, Fb_1, and Fb_2 are B, λB, and $(1-\lambda)B$, respectively, where $0 \leq \lambda \leq 1$ is the frequency band partitioning factor. The model cancels out inter-tier interference, eliminates intra-tier interference, and completely avoids intra-cell interference [18].

Suppose each SMD has a computing task, and the input-bits of the task for the SMD $u_{s,k}$ is denoted as $L_{s,k}$. Assume that the computing task belongs to data-partitioned oriented tasks and can be divided arbitrarily, regardless of the internal content, such as virus scanning tasks and GZip tasks [14]. Tasks can be executed in three layers including the local device of the SMD, the MEC server, and the cloud server respectively. Let $l^u_{s,k}$, $l^m_{s,k}$ and $l^c_{s,k}$ respectively denote the task input-bits executed on the local SMD $u_{s,k}$, on the MEC server, and on the cloud server. Then we have

$$L_{s,k} = l^u_{s,k} + l^m_{s,k} + l^c_{s,k}, \ \forall s \in \mathbb{S}, k \in \mathbb{U}_s. \tag{1}$$

The entire process of task computation and offloading in the system can be categorized into three phases: the local processing phase, the SBS processing phase and the MBS processing phase. In the upcoming sections, we will provide

a detailed analysis of the computation latency, the communication latency, and the energy consumption for each phase.

3.1 Local Processing Phase

The local processing phase involves the local SMD $u_{s,k}$ performing self-computing on the partial task of $l^u_{s,k}$ bits and offloading partial task of $l^m_{s,k} + l^c_{s,k}$ bits to the SBS.

Local Computing. Let $c^u_{s,k}$ represent the number of CPU cycles required by the SMD $u_{s,k}$ to compute each task input-bit, and the computation frequency (in CPU cycles/s) of the SMD $u_{s,k}$ as $f^u_{s,k} \geq 0$, which is limited by the maximum frequency $F^u_{s,k}$. Then the computing time of $l^u_{s,k}$ is

$$t^{u,comp}_{s,k} = \frac{l^u_{s,k} c^u_{s,k}}{f^u_{s,k}}. \tag{2}$$

To calculate the energy consumption of SMDs when performing tasks locally, we use the widely adopted model of the energy consumption per computing cycle as $\varepsilon = \kappa {f^u_{s,k}}^2$, where κ is the energy coefficient depending on the chip architecture [3]. Therefore, the energy consumption of the SMD $u_{s,k}$ to execute the $l^u_{s,k}$ bits locally is given by:

$$E^{comp}_{s,k} = l^u_{s,k} c^u_{s,k} \varepsilon = l^u_{s,k} c^u_{s,k} \kappa (f^u_{s,k})^2. \tag{3}$$

Computation Offloading to SBS. The SMD $u_{s,k}$ offloads the remaining $l^m_{s,k} + l^c_{s,k}$ task bits to the SBS s through a wireless access link. The transmission power of the SMD $u_{s,k}$ is represented by $p_{s,k} \geq 0$, and the maximum transmission power is represented by $P_{s,k}$. The data transmission rate (in bits/s) from the SMD $u_{s,k}$ to the SBS s is given by the equation:

$$r_{s,k} = \frac{B\lambda}{SU_s} \log_2(1 + \frac{p_{s,k} h_{s,k}}{N_s}), \tag{4}$$

where $h_{s,k} \geq 0$ is the channel gain from the SMD $u_{s,k}$ to the SBS s, N_s is the Additive White Gaussian Noise (AWGN) power of the SBS s.

Using the equation (4), we can derive the offloading delay and energy consumption from the SMD $u_{s,k}$ to the SBS s:

$$t^{u,trans}_{s,k} = \frac{l^m_{s,k} + l^c_{s,k}}{r_{s,k}}, \tag{5}$$

$$E^{trans}_{s,k} = t^{u,trans}_{s,k} p_{s,k} = \frac{p_{s,k}(l^m_{s,k} + l^c_{s,k})}{r_{s,k}}. \tag{6}$$

3.2 SBS Processing Phase

After receiving the $l_{s,k}^m + l_{s,k}^c$ task bits from the SMD $u_{s,k}$, the SBS s offloads the task segment to the MBS using a wired or wireless backhaul link in the SBS processing phase.

When the SBS s accesses the MBS through a wireless backhaul link, the transmission power of the SBS s is represented by p_s. The data transmission rate from the SBS s to the MBS is given by:

$$r_s = \frac{B(1-\lambda)}{M} \log_2(1 + \frac{p_s h_s}{N_0}), \forall s \in \mathbb{S}_0, \tag{7}$$

where $h_s \geq 0$ represents the channel gain from the SBS s to the MBS, N_0 represents the AWGN power at the MBS.

Since the transmission delay of the wired backhaul link is typically much smaller than that of the wireless backhaul link, the transmission delay of the wired backhaul link can be ignored [8]. Therefore, we can obtain the transmission delay for the SBS s to send $l_{s,k}^m + l_{s,k}^c$ task bits to the MBS:

$$t_{s,k}^m = \begin{cases} \dfrac{l_{s,k}^m + l_{s,k}^c}{r_s}, & \text{if } s \in \mathbb{S}_0, \\ 0, & \text{if } s \in \mathbb{S}_1. \end{cases} \tag{8}$$

3.3 MBS Processing Phase

The processing phase of the MBS can be split into two parts. In the first part, the MEC server computes the $l_{s,k}^m$ task bits received from the SMD $u_{s,k}$. In the second part, the MBS offloads the remaining $l_{s,k}^c$ bits to the cloud server for computation.

Computing at MEC Server. Let $c_{s,k}^m$ denote the number of CPU cycles required by the MEC server to calculate each task input-bit from the SMD $u_{s,k}$, and let $f_{s,k}^m \geq 0$ specify the computing resources allocated by the MEC server to the SMD $u_{s,k}$. The execution delay of computing $l_{s,k}^m$ task bits on the MEC server can then be expressed as

$$t_{s,k}^{m,comp} = \frac{l_{s,k}^m c_{s,k}^m}{f_{s,k}^m}. \tag{9}$$

Given the limited resources at the edge, it is necessary to satisfy the computing resources constraint, which can be expressed as

$$\sum_{s=1}^{S} \sum_{k=1}^{U_s} f_{s,k}^m \leq F^m, \ \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{10}$$

where F^m is the maximum computational frequency of the MEC server.

Offloading to the Cloud Server. The MBS is connected to the cloud server through a backhaul link, which is typically characterized by high bandwidth and shared by multiple SMDs. Following the assumptions made in [11], we consider a fixed resource scheduling policy. Accordingly, we can denote W as the backhaul communication capacity for each SMD. Therefore, similar to the offloading delay in (5), the backhaul transmission delay $t_{s,k}^{m,trans}$ is proportional to the size of data to be offloaded, as

$$t_{s,k}^{m,trans} = \frac{l_{s,k}^{c}}{W}, \tag{11}$$

where W^{-1} can be interpreted as the required time for the backhaul link to transmit one-bit data.

Since the cloud has abundant computing resources in comparison to the edge, we can neglect the delay of executing these offloaded task bits [6]. Furthermore, the size of the output is small enough, then the download delay can be ignored [5].

4 Problem Formulation

As the MEC server and cloud servers have a reliable power supply, this paper mainly focuses on analyzing the energy consumption of the SMDs. Using the Eqs. (3) and (6), the total energy consumption of the SMD $u_{s,k}$ can be computed as

$$E_{s,k} = E_{s,k}^{comp} + E_{s,k}^{trans}. \tag{12}$$

Since the task of the SMD $u_{s,k}$ can be executed in parallel on the local SMD, the MEC server and the cloud server, the total time delay required for the SMD to complete the task is:

$$T_{s,k} = \max\{t_{s,k}^{u,comp}, t_{s,k}^{u,trans} + t_{s,k}^{m} + t_{s,k}^{m,comp}, t_{s,k}^{u,trans} + t_{s,k}^{m} + t_{s,k}^{m,trans}\}. \tag{13}$$

In a mobile edge computing system, the users' QoE is primarily determined by the time and the energy consumption required to complete tasks. Therefore, we define the computation overhead of the SMD $u_{s,k}$ as follows:

$$Z_{s,k} = \alpha_{s,k} E_{s,k} + (1 - \alpha_{s,k}) T_{s,k}, \tag{14}$$

in which $\alpha_{s,k} \in [0, 1]$.

Our objective is to minimize the system-wide computation overhead, which we define as a weighted sum of all SMDs computation overheads:

$$Z = \sum_{s=1}^{S} \sum_{k=1}^{U_s} \beta_{s,k} Z_{s,k}, \tag{15}$$

where $\beta_{s,k} \in (0, 1]$ specify the resource provider's preference towards the SMD $u_{s,k}$. The minimization problem of the system-wide computation overhead is then formulated as

$$\text{(P1)}: \quad \min_{\mathbf{L},\lambda,\mathbf{F},\mathbf{P}} Z \tag{16a}$$

$$\text{s.t.} \quad l^u_{s,k} \geq 0,\ l^m_{s,k} \geq 0,\ l^c_{s,k} \geq 0,\ \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{16b}$$

$$0 \leq \lambda \leq 1, \tag{16c}$$

$$0 \leq f^u_{s,k} \leq F^u_{s,k},\ \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{16d}$$

$$f^m_{s,k} \geq 0,\ \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{16e}$$

$$0 \leq p_{s,k} \leq P_{s,k},\ \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{16f}$$

$$\sum_{k=1}^{U_s} r_{s,k} \leq r_s,\ \forall s \in \mathbb{S}_0, \tag{16g}$$

$$(1)\ \textit{and}\ (10),$$

where the task offloading decisions is denoted by $\mathbf{L} = (l^u_{1,1}, l^m_{1,1}, l^c_{1,1}, \ldots, l^u_{S,U_S}, l^m_{S,U_S}, l^c_{S,U_S})$, the computing resource allocation policy by $\mathbf{F} = (f^u_{1,1}, f^m_{1,1}, \ldots, f^u_{S,U_S}, f^m_{S,U_S})$, and the transmission power allocation by $\mathbf{P} = (p_{1,1}, \ldots, p_{S,U_S})$, where U_S means the SMD locates in the S-th SBS cell. (16c) represents the lower and upper bounds of the frequency band partitioning factor. (16d) represents the constraints on the computational frequency of the SMDs. (16f) indicates the constraints on the transmission power of the SMDs. (16g) indicates that the total transmission rate of all SMDs associated with the SBS connected to the wireless backhaul link should be not greater than the transmission rate of the SBS.

5 Problem Solving

It is worth noting that the problem (P1) can be further simplified. The task size computed on the local device, on the MEC server and on the cloud server can be modeled as, respectively

$$l^u_{s,k} = \gamma^u_{s,k} L_{s,k}, \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{17}$$

$$l^m_{s,k} = \gamma^m_{s,k}(1 - \gamma^u_{s,k}) L_{s,k}, \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{18}$$

$$l^c_{s,k} = (1 - \gamma^m_{s,k})(1 - \gamma^u_{s,k}) L_{s,k}, \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{19}$$

where $\gamma^u_{s,k} \in [0,1]$ and $\gamma^m_{s,k} \in [0,1]$. In addition, we define the vectors $\mathbf{\Gamma^u} = (\gamma^u_{1,1}, \ldots, \gamma^u_{S,U_S})$ and $\mathbf{\Gamma^m} = (\gamma^m_{1,1}, \ldots, \gamma^m_{S,U_S})$, so the original problem (P1) can be transformed into the following equivalent optimization problem (P1-Eqv):

$$\text{(P1-Eqv)}: \quad \min_{\mathbf{\Gamma^u},\mathbf{\Gamma^m},\lambda,\mathbf{F},\mathbf{P}} Z \tag{20a}$$

$$\text{s.t.} \quad \gamma^u_{s,k} \in [0,1], \gamma^m_{s,k} \in [0,1], \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{20b}$$

$$(16c), (16d), (16e), (16f), (16g)\ \textit{and}\ (10).$$

The Particle Swarm Optimization (PSO) algorithm is a kind of swarm intelligent algorithm inspired by bird predation behaviors. It analogizes the optimization problem's search space to the birds' flight space and represents each bird as a particle. Each particle has its fitness value equals to the value of the objective function (20a), and particles follow the current optimal particle to search in the solution space.

5.1 Particle Encoding

To utilize the PSO algorithm to solve the problem (P1-Eqv), we represent the vector of particles as $\mathbf{I} = (1, 2, \ldots, I)$, and $\mathbf{\Gamma^u}$, $\mathbf{\Gamma^m}$, λ, $\mathbf{P}$ are encoded as $\mathbf{A_i}$, $\mathbf{B_i}$, c_i, $\mathbf{Q_i}$, $i \in \mathbf{I}$, where $\mathbf{A_i} = (a^i_{1,1}, \ldots, a^i_{S,U_S})$, and $a^i_{s,k}$ represents the proportion of the task at the SMD $u_{s,k}$ to be computed locally found by the particle i; $\mathbf{B_i} = (b^i_{1,1}, \ldots, b^i_{S,U_S})$, and $b^i_{s,k}$ represents the proportion dimensions of the task received by the MBS from the SMD $u_{s,k}$ to be computed on the MEC server found by the particle i; c_i is the frequency band partitioning factor found by the particle i; $\mathbf{Q_i} = (q^i_{1,1}, \ldots, q^i_{S,U_S})$, and $q^i_{s,k}$ is the transmission power of the SMD $u_{s,k}$ found by the particle i. $\mathbf{F}$ is further encoded as $\mathbf{D_i}$ and $\mathbf{E_i}$, where $\mathbf{D_i} = (d^i_{1,1}, \ldots, d^i_{S,U_S})$, and $d^i_{s,k}$ represents the computation frequency of the SMD $u_{s,k}$ found by the particle i; $\mathbf{E_i} = (e^i_{1,1}, \ldots, e^i_{S,U_S})$, and $e^i_{s,k}$ represents the computing resources allocated by the MEC server to the SMD $u_{s,k}$ found by the particle i.

5.2 Particle Velocity and Position Update

In the PSO algorithm, each particle has two attributes including the position and the velocity, where the position represents a solution found by the particle of the optimization problem, and the velocity shows how the solution evolves.

According to the constraint conditions (10), (16c)-(16f), and (20b), the initial position of the particle i is described as follows:

$$\begin{cases} a^{i,0}_{s,k} = rand(1), \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \\ b^{i,0}_{s,k} = rand(1), \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \\ c^0_i = rand(1), \\ d^{i,0}_{s,k} = rand(F^u_{s,k}), \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \\ e^{i,0}_{s,k} = rand(\frac{\beta_{s,k} F^m}{\sum\limits_{s=1}^{S} \sum\limits_{k=1}^{U_s} \beta_{s,k}}), \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \\ q^{i,0}_{s,k} = rand(P_{s,k}), \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \end{cases} \tag{21}$$

where $rand(a)$ randomly generates a number between 0 and a.

Then, the velocity of the particle i can be updated by

$$\begin{aligned} av^{i,t+1}_{s,k} =& \omega^t av^{i,t}_{s,k} + c_1 \eta^i_{s,k} (aj^{i,t}_{s,k} - al^{i,t}_{s,k}) + c_2 \hat{\eta}^i_{s,k} (ag^t_{s,k} - al^{i,t}_{s,k}), \\ & \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \end{aligned} \tag{22}$$

$$bv_{s,k}^{i,t+1} = \omega^t bv_{s,k}^{i,t} + c_1\eta_{s,k}^{i}(bj_{s,k}^{i,t} - bl_{s,k}^{i,t}) + c_2\hat{\eta}_{s,k}^{i}(bg_{s,k}^{t} - bl_{s,k}^{i,t}), \quad \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{23}$$

$$cv_i^{t+1} = \omega^t cv_i^t + c_1\xi_i(cj_i^t - cl_i^t) + c_2\hat{\xi}_i(cg^t - cl_i^t), \tag{24}$$

$$dv_{s,k}^{i,t+1} = \omega^t dv_{s,k}^{i,t} + c_1\eta_{s,k}^{i}(dj_{s,k}^{i,t} - dl_{s,k}^{i,t}) + c_2\hat{\eta}_{s,k}^{i}(dg_{s,k}^{t} - dl_{s,k}^{i,t}), \quad \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{25}$$

$$ev_{s,k}^{i,t+1} = \omega^t ev_{s,k}^{i,t} + c_1\eta_{s,k}^{i}(ej_{s,k}^{i,t} - el_{s,k}^{i,t}) + c_2\hat{\eta}_{s,k}^{i}(eg_{s,k}^{t} - el_{s,k}^{i,t}), \quad \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{26}$$

$$qv_{s,k}^{i,t+1} = \omega^t qv_{s,k}^{i,t} + c_1\eta_{s,k}^{i}(qj_{s,k}^{i,t} - ql_{s,k}^{i,t}) + c_2\hat{\eta}_{s,k}^{i}(qg_{s,k}^{t} - ql_{s,k}^{i,t}), \quad \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{27}$$

where $al_{s,k}^{i,t}$, $bl_{s,k}^{i,t}$, cl_i^t, $dl_{s,k}^{i,t}$, $el_{s,k}^{i,t}$ and $ql_{s,k}^{i,t}$ are the position coordinates of $a_{s,k}^i$, $b_{s,k}^i$, c_i, $d_{s,k}^i$, $e_{s,k}^i$ and $q_{s,k}^i$ at the t-th iteration; $av_{s,k}^{i,t}$, $bv_{s,k}^{i,t}$, cv_i^t, $dv_{s,k}^{i,t}$, $ev_{s,k}^{i,t}$ and $qv_{s,k}^{i,t}$ are the velocity vectors of $a_{s,k}^i$, $b_{s,k}^i$, c_i, $d_{s,k}^i$, $e_{s,k}^i$ and $q_{s,k}^i$ at the t-th iteration; ω^t represents an inertia weight at the t-th iteration; c_1 and c_2 denote the self-learning and social learning factors respectively; $\eta_{s,k}^i$, $\hat{\eta}_{s,k}^i$, ξ_i and $\hat{\xi}_i$ are random numbers between [0,1]; $aj^{i,t} = (aj_{1,1}^{i,t}, \ldots, aj_{S,U_S}^{i,t})$, $bj^{i,t} = (bj_{1,1}^{i,t}, \ldots, bj_{S,U_S}^{i,t})$, cl_i^t, $dj^{i,t} = (dj_{1,1}^{i,t}, \ldots, dj_{S,U_S}^{i,t})$, $ej^{i,t} = (ej_{1,1}^{i,t}, \ldots, ej_{S,U_S}^{i,t})$ and $qj^{i,t} = (qj_{1,1}^{i,t}, \ldots, qj_{S,U_S}^{i,t})$ are the historical optimal position coordinates of the particle i until the t-th iteration; $ag^t = (ag_{1,1}^t, \ldots, ag_{S,U_S}^t)$, $bg^t = (bg_{1,1}^t, \ldots, bg_{S,U_S}^t)$, cg^t, $dg^t = (dg_{1,1}^t, \ldots, dg_{S,U_S}^t)$, $eg^t = (eg_{1,1}^t, \ldots, eg_{S,U_S}^t)$ and $qg^t = (qg_{1,1}^t, \ldots, qg_{S,U_S}^t)$ are the historical optimal position coordinates in the solution space until the t-th iteration.

To balance the global and local search capabilities, a Linearly Decreasing Weight (LDW) strategy can be employed, which has better performance than using a fixed inertia weight [15]. Specifically, the update formula for the inertia weight is:

$$\omega^t = \omega_{max} - \frac{t(\omega_{max} - \omega_{min})}{T}, \tag{28}$$

where ω_{max} and ω_{min} are the maximum and minimum inertia weights, and T is the number of iterations. After updating the velocities of particles, the position coordinates of the particle i can be updated by

$$al_{s,k}^{i,t+1} = al_{s,k}^{i,t} + av_{s,k}^{i,t+1}, \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{29}$$

$$bl_{s,k}^{i,t+1} = bl_{s,k}^{i,t} + bv_{s,k}^{i,t+1}, \forall s \in \mathbb{S}, k \in \mathbb{U}s, \tag{30}$$

$$cl_i^{t+1} = al_i^t + cv_i^{t+1}, \tag{31}$$

$$dl_{s,k}^{i,t+1} = dl_{s,k}^{i,t} + dv_{s,k}^{i,t+1}, \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{32}$$

$$el_{s,k}^{i,t+1} = el_{s,k}^{i,t} + ev_{s,k}^{i,t+1}, \forall s \in \mathbb{S}, k \in \mathbb{U}_s, \tag{33}$$

$$ql_{s,k}^{i,t+1} = ql_{s,k}^{i,t} + qv_{s,k}^{i,t+1}, \forall s \in \mathbb{S}, k \in \mathbb{U}_s. \tag{34}$$

In summary, the specific process of using the LDWPSO algorithm to solve the optimization problem (P1-Eqv) is shown in Algorithm 1.

Algorithm 1: LDWPSO

Initialization:
  $t = 1$, and a large fitness value N
  Initialize the positions of I particles using Eq.(21).
  Set the current position of each particle as its historical optimal position.
  Find the global best particle of current population.
  Initialize the velocities of I particles.
While $t \leq T$ **Do**
  Update inertia weight using Eq.(28).
  Update the velocities of I particles using Eqs.(22)–(27).
  Update the positions of I particles using Eqs.(29)–(34).
  For each $i \in$ **I Do**
    If the position of the particle i satisfies the constraints of the problem (P1-Eqv) **then**
      calculate the fitness value of the particle i using Eq.(20a);
    else
      set its fitness value to N.
    End if
    If its current fitness value is less than its historical optimal fitness value **then**
      the historical optimal position is updated by its current position.
    End if
  End for
  Find the global best particle of current population.
  Update the iteration index: $t = t + 1$.
End while
Output:
  ag^T, bg^T, cg^T, dg^T, eg^T and qg^T as the solution to (P1-Eqv)

6 Results and Analysis

In this section, we will evaluate the performance of the proposed joint task offloading and resource allocation scheme by setting corresponding simulation parameters. We consider an outdoor environment of 300m $\times$ 300m, where the MBS is located at the center, 15 SBSs and K SMDs are randomly distributed within the area. Let d denote the distance between the transmitter and the receiver. The path-loss between any two nodes is given by $\beta_0(d/d_0)^{-\xi}$, where $\beta_0 = -60$db represents the path-loss at the reference distance of $d_0 = 10$m, and $\xi = 3$ is the path-loss exponent [12]. Table 1 lists other parameters in detail.

Table 1. Simulation Parameters

Parameter	Value	Parameter	Value
ω_{max}	0.9	ω_{min}	0.4
c_1	2	c_2	2
T	800	I	500
K	[15,120]	$L_{s,k}$	[1,10] Mbits
B	[5,50] MHz	$c^u_{s,k}$	10^3cycles/bit
$c^m_{s,k}$	10^3cycles/bit	κ	10^{-26}
N_s	-60 dbm	N_0	-70 dbm
p_s	20 dbm	$P_{s,k}$	20 dbm
m	4	W	5 Mbps
$F^u_{s,k}$	1 GHz	F^m	50 GHz
N	10^{10}	$\alpha_{s,k}$	0.6
$\beta_{s,k}$	1		

Based on these parameters, we conducted simulation experiments to compare the performance of the proposed collaborative cloud-edge scheme with three baseline schemes:

1) Local-Computing-Only: Tasks are not offloaded, and all SMDs only use local computing to execute their tasks.
2) Full-Offloading: All SMDs offload all their tasks to the cloud and the MEC server at the MBS.
3) Computing-without-Cloud: Tasks are not offloaded to the cloud server and are jointly executed by SMDs locally and the MEC server at the MBS.

Figure 2 shows the relationship between the system-wide computation overhead (i.e., total computation overhead) and the total SMD number K when $L = 5$ Mbits and $B = 30$ MHz. The experimental results demonstrate that the system-wide computation overhead of all schemes gradually increases with the increase of the number of SMDs. It was observed that the proposed collaborative cloud-edge scheme is superior to other schemes. This is because the collaborative cloud-edge scheme not only has local and edge collaboration, but also takes the advantage of the larger computing resources of the cloud server. Moreover, when the number of SMDs is less than 30, the computation overhead of computing-without-cloud is close to that of full-offloading. This is because when the number of SMDs is relatively small, the computing resources that SMDs obtain from the MEC server as well as the allocated bandwidth are sufficient, while when the number of SMDs is greater than 60, the performance of local-computing-only is better than that of full-offloading, and the gap between them gradually increases. This is because the computing resources of the MEC server and the system bandwidth are not enough to meet the needs of multiple devices simultaneously.

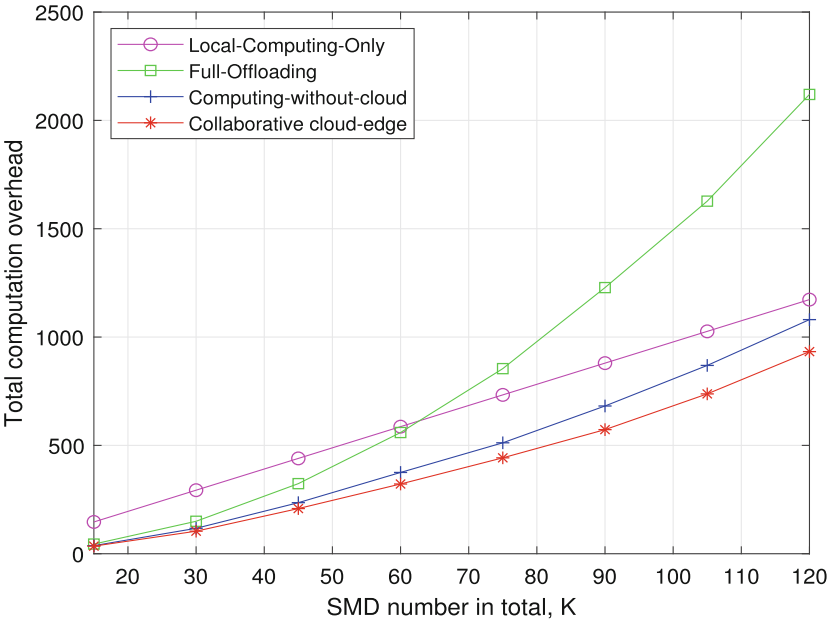

Fig. 2. Computation overhead versus SMD number.

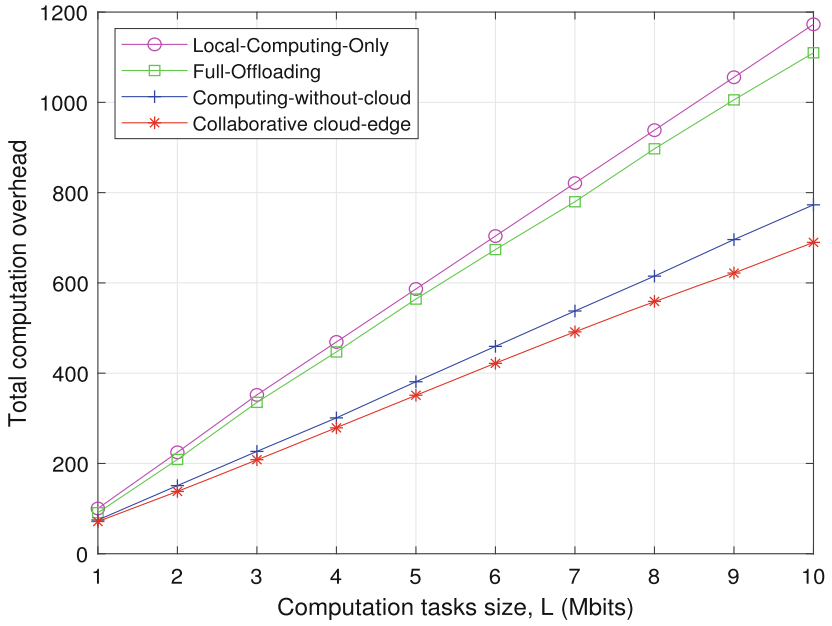

Fig. 3. Computation overhead versus computation task size.

Figure 3 illustrates the variation of computation overhead for the four schemes with the increase of the computation tasks size when $K = 60$ and $B = 30\,\text{MHz}$. The results show that the overhead of all schemes increases as the computation task size grows. Among these schemes, the overhead of the proposed collaborative cloud-edge scheme is the smallest. This is because the proposed scheme support offload partial of the task to the MEC server and cloud server for execution. When the computation task size is small, the performance of local-computing-only is similar to the other schemes, as the CPU execution frequency of the local device is sufficient to complete the computation task. It can be observed that the overhead of full-offloading is higher than that of computing-without-cloud, and this gap gradually increases with the increase of the computation task size. This indicates that with the increase of the computation task size, the energy consumption and the latency required for communication (data transmission) will also increase.

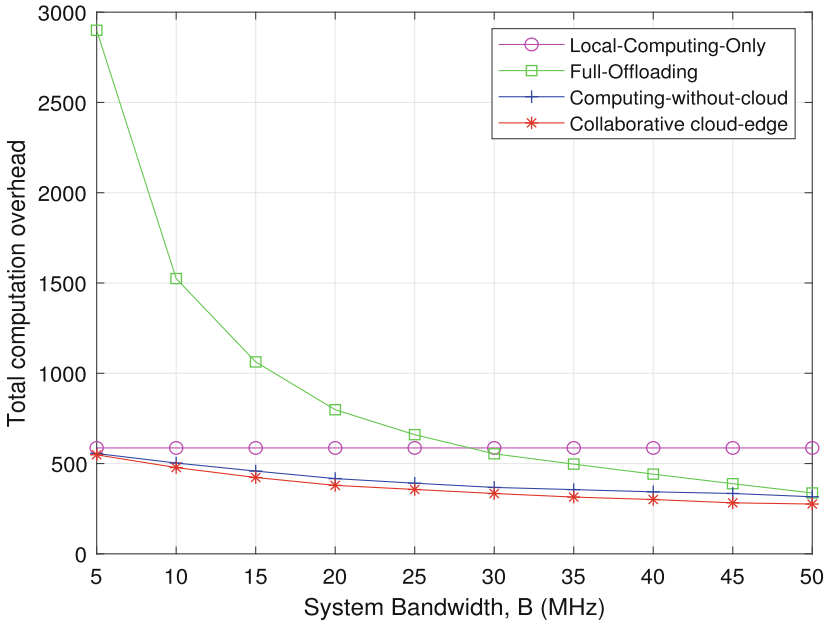

Fig. 4. Computation overhead versus system bandwidth.

Figure 4 illustrates the relationship between the system-wide computation overhead and system bandwidth when $K = 60$ and $L = 5\,\text{Mbits}$. As can be seen, the computation overhead of local-computing-only remains constant as system bandwidth increases, while the overhead of other schemes decreases. This is because local-computing-only is independent of offloading. We can again observe that the proposed collaborative cloud-edge scheme generates the minimum overhead compared to the other baseline schemes. When the system bandwidth

keeps low, full-offloading performs more worse than the other schemes for long offloading delay. When the system bandwidth is large enough (i.e., great than 28 MHz), full-offloading is better than local-computing-only. Especially, when the system bandwidth is relatively large enough, such as 50 MHz, the overhead of full-offloading is almost as low as that of the proposed scheme. This is reasonable because when the system bandwidth is large enough, the task offloading time is quite small.

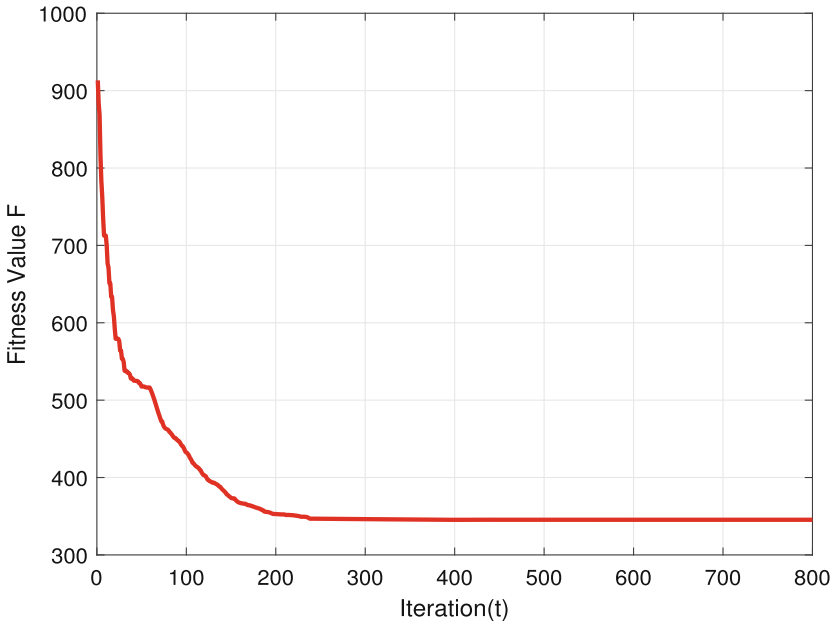

Fig. 5. Convergence analysis.

Figure 5 evaluates the convergence of the LDWPSO algorithm for the proposed scheme with $K = 60$, $L = 5$ Mbits, and $B = 30$ MHz by varying the number of iterations to observe the system-wide fitness value. It can be observed that the algorithm converges quickly in the first 150 iterations and the fitness value remains unchanged after 250 iterations with the global optimal solution found which means the LDWPSO algorithm can constantly search for the global optimal solution in the early stage of the algorithm and has good local search capabilities in the later stage.

7 Conclusions

With the advent of the 5G era, the requirements for delay and energy consumption in mobile communication networks have become increasingly strict.

To meet the QoE demands of SMDs, this paper explores the problem of task offloading and resource allocation based on collaborative cloud-edge system in the UDN. Under the deployment of mixed wired and wireless backhaul links, a system model for computing offloading is constructed, and the computation time, offloading delay, and energy consumption are given for the local processing, the SBS processing, and the MBS processing, respectively. Using the LDWPSO algorithm, the optimization problem including the computing offloading decision, frequency band partitioning factor, CPU execution frequency of the SMDs and the MEC server, and the transmission power of the SMDs is solved, and the optimal system-wide computation overhead is achieved in the entire task offloading process. Simulation results show that our proposed collaborative cloud-edge offloading scheme effectively reduces the system-wide computation overhead and improves the overall system performance compared to the other three baseline solutions.

References

1. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: a survey. IEEE Internet Things J. **5**(1), 450–465 (2018). https://doi.org/10.1109/JIOT.2017.2750180
2. Cao, X., Wang, F., Xu, J., Zhang, R., Cui, S.: Joint computation and communication cooperation for energy-efficient mobile edge computing. IEEE Internet Things J. **6**(3), 4188–4200 (2018)
3. Chen, X.: Decentralized computation offloading game for mobile cloud computing. IEEE Trans. Parallel Distrib. Syst. **26**(4), 974–983 (2014)
4. Dong, S., Li, H., Qu, Y., Zhang, Z.: Survey of research on computation unloading strategy in mobile edge computing. Comput. Sci. **46**(11), 32–40 (2019)
5. Elgendy, I.A., Zhang, W.Z., He, H., Gupta, B.B., Abd El-Latif, A.A.: Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms. Wireless Netw. **27**(3), 2023–2038 (2021)
6. Gao, Z., Hao, W., Han, Z., Yang, S.: Q-learning-based task offloading and resources optimization for a collaborative computing system. IEEE Access **8**, 149011–149024 (2020)
7. Ge, X., Tu, S., Mao, G., Wang, C.X., Han, T.: 5G ultra-dense cellular networks. IEEE Wirel. Commun. **23**(1), 72–79 (2016)
8. Haibo, Z., Hu, L., Shanxue, C., Xiaofan, H.: Computing offloading and resource optimization in ultra-dense networks with mobile edge computation. J. Electr. Inf. Technol. **41**(05), 1194–1201 (2019)
9. Oo, T.Z., Tran, N.H., Saad, W., Niyato, D., Han, Z., Hong, C.S.: Offloading in HetNet: a coordination of interference mitigation, user association, and resource allocation. IEEE Trans. Mob. Comput. **16**(8), 2276–2291 (2016)
10. Pham, Q.V., Le, L.B., Chung, S.H., Hwang, W.J.: Mobile edge computing with wireless backhaul: joint task offloading and resource allocation. IEEE Access **7**, 16444–16459 (2019)
11. 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)

12. Sun, H., Wang, J., Peng, H., Song, L., Qin, M.: Delay constraint energy efficient cooperative offloading in MEC for IoT. In: Gao, H., Wang, X., Iqbal, M., Yin, Y., Yin, J., Gu, N. (eds.) Collaborative Computing: Networking, Applications and Worksharing. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 349, pp. 671–685. Springer, Cham (2020)
13. Tran, T.X., Pompili, D.: Joint task offloading and resource allocation for multi-server mobile-edge computing networks. IEEE Trans. Veh. Technol. **68**(1), 856–868 (2018)
14. Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J.: Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans. Commun. **64**(10), 4268–4282 (2016)
15. Xiaojing, Y., Qingju, J., Xinke, L.: Center particle swarm optimization algorithm. In: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), pp. 2084–2087. IEEE (2019)
16. Xing, H., Liu, L., Xu, J., Nallanathan, A.: Joint task assignment and resource allocation for d2d-enabled mobile-edge computing. IEEE Trans. Commun. **67**(6), 4193–4207 (2019)
17. 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 (2020)
18. Zhou, T., Yue, Y., Qin, D., Nie, X., Li, X., Li, C.: Joint device association, resource allocation, and computation offloading in ultradense multidevice and multitask IoT networks. IEEE Internet Things J. **9**(19), 18695–18709 (2022)