Hindawi
Security and Communication Networks
Volume 2022, Article ID 7258236, 13 pages
https://doi.org/10.1155/2022/7258236

Research Article

Dynamic Task Offloading for NOMA-Enabled Mobile Edge Computing with Heterogeneous Networks

Kaixin Li,[1] Jiajie Xu,[1] Hua Xing,[1] Ying Chen,[1] and Jiwei Huang[2]

[1]*School of Computer Science, Beijing Information Science and Technology University, Beijing 100101, China*
[2]*Jiwei Huang Is with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China*

Correspondence should be addressed to Ying Chen; chenying@bistu.edu.cn

Received 19 June 2022; Accepted 22 August 2022; Published 20 September 2022

Academic Editor: Xiaolong Xu

With the rapid development of the Internet of Things (IoT), more and more computation-intensive tasks are generated by IoT devices. Due to their own limitations, IoT devices cannot process all tasks locally, and some tasks need to be offloaded to edge servers for processing. In addition, nonorthogonal multiple access (NOMA) technology allows multiple IoT devices to share the same frequency resource. IoT devices can use NOMA technology to transmit data to increase the data transmission rate. In this article, we study the problem of NOMA-enabled dynamic task offloading in heterogeneous networks. We formulate a stochastic optimization problem to minimize system energy consumption. Using stochastic optimization techniques, we transform this problem into a deterministic optimization problem and decompose it into five sub-problems to solve. At the same time, we propose a NOMA-enabled dynamic task offloading (NDTO) algorithm. Then, we mathematically analyze the performance of the NDTO algorithm. We conduct a series of parameter analysis experiments and comparative experiments, and the results verify the performance of the NDTO algorithm.

1. Introduction

In recent years, with the rapid development of Internet of Things (IoT) technology, various IoT devices have appeared in people's lives [1, 2]. At the same time, more and more computation-intensive applications are emerging, such as virtual reality. These computation-intensive applications generate a large number of computation-intensive tasks as they work. However, due to the limited computing resources and battery capacity of IoT devices, these computation-intensive tasks cannot be fully handled [3–5]. To cope with this situation, mobile cloud computing (MCC) came into being [6–8]. However, offloading tasks to cloud computing centers for processing will introduce significant delays [9–11]. The emergence of mobile edge computing (MEC) provides a new solution [12–14]. At the same time, in order to cope with the problem of data traffic explosion brought by the increasing number of IoT devices, a two-layer heterogeneous small cell network (SCN) is proposed [15]. The SCN is usually composed of macro base station (MBS) and small base station (SBS). Both the MBS and the SBS are generally deployed with an MEC server for providing services. SCN can expand network coverage and further improve user experience quality. Therefore, SCN will play an important role in the future [16].

With the rapid increase of IoT devices and various computation-intensive applications, the traditional orthogonal multiple access (OMA) technology can no longer meet the increasing communication demands. Non-orthogonal multiple access (NOMA) technology is a promising technology that can alleviate the problem of scarcity of spectrum resources [17–19]. NOMA uses power multiplexing technology that allows multiple devices to share the same spectrum resources. Thus, NOMA can greatly improve the resource utilization efficiency of the wireless access network [20, 21].

Since both the computing resources of IoT devices and the computing resources that MEC can provide are limited,

it is challenging to design a suitable task offloading scheme to maximize the use of available computing resources [22–24]. First, the generation and arrival of tasks are dynamic. Second, the quality of the wireless channel is time-varying. Whether the wireless channel state is good or bad will have a great impact on task offloading. At the same time, the statistics of these factors are difficult to predict. Therefore, it is urgent to design a task offloading scheme that can dynamically adapt to changes in wireless channels and task arrivals.

In this article, we study the NOMA-enabled dynamic task offloading problem in MEC. We work to minimize the long-term average energy consumption of the system. Meanwhile, we design a NOMA-enabled dynamic task offloading (NDTO) algorithm using stochastic optimization techniques. Moreover, we evaluate the performance of the NDTO algorithm through a series of experiments. Our contributions are as follows:

(1) We study the problem of dynamic task offloading for SBSs and devices based on NOMA. Both SBSs and MBS are deployed with MEC servers. IoT devices can offload tasks to SBS for processing through NOMA technology. At the same time, the SBS can also offload the collected tasks to the MBS for processing. Our goal is to minimize the long-term average energy consumption of the system.

(2) Since task arrival and channel state are time-varying. Therefore, we transform the above problem into a deterministic optimization problem using stochastic optimization techniques. Then, we decompose it into multiple sub-problems to solve. Meanwhile, we design the NDTO algorithm to solve this problem.

(3) We verify the effectiveness of the NDTO algorithm through parameter analysis and comparative experiments. Experimental results demonstrate the effectiveness of the NDTO algorithm in minimizing system energy consumption and maintaining task queue stability.

The remainder of this article is as follows. In Section 2, we present the system model and problem formulation. A NOMA-enabled dynamic task offloading algorithm is proposed in Section 3. In Section 4, we analyze the NDTO algorithm mathematically. In Section 5, we experimentally verify the performance of the NDTO algorithm. In Section 6, we discuss related works. In Section 7, we conclude this study.

2. System Model and Problem Formulation

2.1. System Model. In this study, we consider a system with one MBS and multiple SBSs. Each SBS is deployed with an MEC server to serve IoT devices. In this work, IoT devices can offload tasks to SBS for processing. In our model, there are S SBSs, which are defined as $\mathcal{S} = \{1, 2, \ldots, s, \ldots, S\}$. Each SBS has a fixed service area, and the service areas of any two SBSs do not overlap. Among them, there are N_s IoT devices located in the service area of SBS s, which is represented as $\mathcal{N}^s = \{1, 2, \ldots, i, \ldots, N_s\}$. For the convenience of description, we denote the IoT device i within the service area of SBS s as $U_{s,i}$. We consider a discrete slot model denoted as $\mathcal{T} = \{0, 1, \ldots, T-1\}$. At the same time, $t \in \mathcal{T}$, and the length of each slot is τ. The main symbols are listed in Table 1.

2.2. Task and Communication Model. In this study, each IoT device maintains two task queues, namely the local computing queue and the local offloading queue. The tasks in the local computing queue are processed locally by IoT devices. The tasks in the local offloading queue are offloaded to the SBS for processing. In each time slot, the size of the task generated by the IoT device $U_{s,i}$ is denoted as $A_{s,i}(t)$. $I_{s,i}(t)$ is an offload decision, indicating whether newly arrived tasks are placed on the local computing queue or the local offloading queue. When $I_{s,i}(t) = 0$, the newly arrived task is placed in the local computing queue, otherwise, it is placed in the local offloading queue.

IoT devices communicate with SBS via NOMA wireless network. $h_{s,i}(t)$ represents the channel gain from the IoT device $U_{s,i}$ to the SBS s. We assume that the channel gain of IoT devices within the SBS's service area is in nonincreasing order, i.e., $h_{s,1}(t) \geq h_{s,2}(t) \geq \cdots \geq h_{s,i}(t) \geq \cdots \geq h_{s,N_s}(t)$. Using the SIC technique, the signal is iteratively decoded in order of the IoT device's channel gain [25]. Therefore, we can get the signal-to-noise ratio when the IoT device $U_{s,i}$ communicates with the SBS s as follows:

$$\gamma_{s,i}(t) = \frac{P_{s,i}(t)\left|h_{s,i}(t)\right|^2}{\sum_{j=i+1}^{N_s} P_{s,j}(t)\left|h_{s,j}(t)\right|^2 + \sigma^2}, \tag{1}$$

where $P_{s,i}(t)$ represents the transmit power of the IoT device $U_{s,i}$ and σ^2 represents the channel noise power.

Therefore, the achievable data transmission rate when the IoT device $U_{s,i}$ communicates with the SBS s is expressed as follows:

$$R_{s,i}(t) = B_s \log_2\left(1 + \gamma_{s,i}(t)\right), \tag{2}$$

where B_s represents the channel bandwidth allocated by SBS s to IoT devices.

Orthogonal frequency division multiple access technology is used when SBS communicates with MBS. $g_s(t)$ represents the channel gain when the SBS s communicates with the MBS. $P_s(t)$ represents the transmit power of the SBS s. Therefore, we can get the achievable data transmission rate when the SBS s communicates with the MBS as follows:

$$R_s(t) = B_m \log_2\left(1 + \frac{P_s(t) g_s(t)}{\sigma^2}\right), \tag{3}$$

where B_m represents the channel bandwidth allocated by MBS to SBSs.

2.3. Queuing Model. Each IoT device maintains a local computing queue and a local offloading queue. The number of CPU cycles consumed by the IoT device $U_{s,i}$ to process 1 bit of data is $\varphi_{s,i}$. The amount of tasks processed locally by the

Table 1: Notations.

Symbol	Description
$\mathcal{S}$	The set of SBSs
$\mathcal{N}^s$	A set of IoT devices within the service area of SBS s.
τ	The length of time slot
$P_{s,i}(t)$	The transmit power of $U_{s,i}$
$h_{s,i}(t)$	The channel power gain of IoT device $U_{s,i}$
$f^l_{s,i}(t)$	The CPU cycle frequency of the IoT device $U_{s,i}$
$g_s(t)$	The channel power gain of SBS s
$P_s(t)$	The transmit power of SBS s
$R_{s,i}(t)$	The data offloading rate of IoT device $U_{s,i}$
$A_{s,i}(t)$	The amount of tasks generated by IoT device $U_{s,i}$
$D^l_{s,i}(t)$	The amount of tasks processed locally by IoT device $U_{s,i}$
$D^o_{s,i}(t)$	The amount of offloaded tasks for IoT device $U_{s,i}$
$\mu^l_{s,i}(t)$	The amount of tasks from the IoT device $U_{s,i}$ processed locally by SBS s
$\mu^o_{s,i}(t)$	The amount of tasks from the IoT device $U_{s,i}$ that SBS s offloads to the MBS
$Q^l_{s,i}(t)$	The task backlog of the local computing queue of IoT device $U_{s,i}$
$Q^o_{s,i}(t)$	The task backlog of the local offloading queue of IoT device $U_{s,i}$
$H_{s,i}(t)$	The task queue backlog that IoT device $U_{s,i}$ offloads to SBS s
$U_{s,i}$	The IoT device i within the service area of SBS s

IoT device in the time slot t is denoted as $D^l_{s,i}(t)$, and the specific calculation method is as follows:

$$D^l_{s,i}(t) = \frac{f^l_{s,i}(t)\tau}{\varphi_{s,i}}, \tag{4}$$

where $f^l_{s,i}(t)$ represents the CPU cycle frequency of the IoT device $U_{s,i}$.

We define the maximum CPU cycle frequency of the IoT device $U_{s,i}$ as $f^{l,\max}_{s,i}$. Thus, we can get:

$$f^l_{s,i}(t) \le f^{l,\max}_{s,i}. \tag{5}$$

We use $Q^l_{s,i}(t)$ to denote the task backlog of the local computing queue of IoT device $U_{s,i}$. Then, the evolution of $Q^l_{s,i}(t)$ can be obtained as follows:

$$Q^l_{s,i}(t+1) = \max\left\{Q^l_{s,i}(t) - D^l_{s,i}(t), 0\right\} + \left(1 - I_{s,i}(t)\right)A_{s,i}(t). \tag{6}$$

The amount of tasks offloaded by IoT device $U_{s,i}$ in time slot t is denoted as $D^o_{s,i}(t)$. Because the data transmission rate of IoT devices has an upper limit, the amount of offloaded tasks should satisfy:

$$D^o_{s,i}(t) \le R_{s,i}(t)\tau. \tag{7}$$

We define $Q^o_{s,i}(t)$ as the task backlog of the local offloading queue of IoT device $U_{s,i}$. Then, we give the evolution of $Q^o_{s,i}(t)$ as follows:

$$Q^o_{s,i}(t+1) = \max\left\{Q^o_{s,i}(t) - D^o_{s,i}(t), 0\right\} + I_{s,i}(t)A_{s,i}(t). \tag{8}$$

Each SBS maintains a task queue for each IoT device in its service area to store tasks offloaded from IoT devices. In each time slot, the amount of tasks from the IoT device $U_{s,i}$ processed locally by the SBS is $\mu^l_{s,i}(t)$. Then, $\mu^o_{s,i}(t)$ represents the amount of tasks from the IoT device $U_{s,i}$ that the SBS offloads to the MBS.

We define $H_{s,i}(t)$ to represent the task queue backlog that IoT device $U_{s,i}$ offloads to SBS s. Then, the evolution of $H_{s,i}(t)$ is as follows:

$$H_{s,i}(t+1) = \max\left\{H_{s,i}(t) - \mu^l_{s,i}(t) - \mu^o_{s,i}(t), 0\right\} + D^o_{s,i}(t). \tag{9}$$

2.4. Energy Consumption Model. For each IoT device, its energy consumption includes two parts, namely local computing energy consumption and local offloading energy consumption. We know that the local computing energy consumption of an IoT device is closely related to its CPU frequency [26]. We denote the energy consumption of IoT device $U_{s,i}$ locally calculated in time slot t as $E^l_{s,i}(t)$, as follows:

$$E^l_{s,i}(t) = \xi_{s,i} f^{l2}_{s,i}(t)\varphi_{s,i}D^l_{s,i}(t), \tag{10}$$

where $\xi_{s,i}$ denotes the effective switched capacitance of the CPU.

The offloading energy consumption of IoT devices is closely related to its transmit power. The energy consumption generated by IoT device $U_{s,i}$ offloading tasks to SBS s in time slot t is denoted by:

$$E^o_{s,i}(t) = P_{s,i}(t)\frac{D^o_{s,i}(t)}{R_{s,i}(t)}. \tag{11}$$

In each time slot, the energy consumption generated by SBS includes two parts, one part is computing energy consumption, and the other part is offloading energy consumption. During time slot t, the energy consumption generated by the tasks from the IoT device $U_{s,i}$ processed by the SBS s is denoted as:

$$C^l_{s,i}(t) = l_1\mu^l_{s,i}(t), \tag{12}$$

where l_1 denotes the energy consumption coefficient of the MEC server in SBS s [27].

Then, the energy consumption generated by SBS s offloading tasks to MBS is denoted as:

$$C_{s,i}^{o}(t) = P_s(t)\frac{\mu_{s,i}^{o}(t)}{R_s(t)}. \tag{13}$$

All tasks offloaded by SBS will be handled by MBS. We set the energy consumption generated by the MBS to process the tasks from the IoT device U offloaded by the SBS s as:

$$C_{s,i}^{M}(t) = l_2\mu_{s,i}^{o}(t), \tag{14}$$

where l_2 is the energy consumption coefficient of MBS.

Therefore, in a time slot t, the total energy consumption generated by the system is as follows:

$$E(t) = \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[E_{s,i}^{l}(t) + E_{s,i}^{o}(t) + C_{s,i}^{l}(t) + C_{s,i}^{o}(t) + C_{s,i}^{M}(t)\right]. \tag{15}$$

2.5. Problem Formulation. Within each slot, we make a decision to determine whether newly arrived tasks should be executed locally or offloaded to the SBS for processing. Our goal is to minimize the long-term average energy consumption of the system, as follows:

$$\begin{aligned}
&\min_{I(t),f^{l}(t),D^{o}(t),\mu^{l}(t),\mu^{o}(t)} \overline{E} = \lim_{T\longrightarrow\infty}\frac{\sum_{t=0}^{T-1}E\{E(t)\}}{T}\\
&\quad C1\text{: } I_{s,i}(t) \in \{0,1\},\\
&\quad C2\text{: } 0 \le f_{s,i}^{l}(t) \le f_{s,i}^{l,\max},\\
&\quad C3\text{: } 0 \le D_{s,i}^{o}(t) \le R_{s,i}(t)\tau,\\
&\text{s.t.}\quad C4\text{: } D_{s,i}^{o}(t) \le Q_{s,i}^{o}(t),\\
&\quad C5\text{: } 0 \le \sum_{i=1}^{N_s}\mu_{s,i}^{l}(t) \le \frac{f_{s,i}^{e}(t)\tau}{\varphi_e}, \quad \forall s \in \mathcal{S},\\
&\quad C6\text{: } 0 \le \mu_{s,i}^{o}(t) \le R_s(t)\tau, \quad \forall s \in \mathcal{S}.
\end{aligned} \tag{16}$$

There are certain challenges to solving this problem. First, the generation and arrival of tasks are random. Second, the wireless channel state is also constantly changing. Therefore, next, we use stochastic optimization techniques to solve this problem.

3. NOMA-Enabled Dynamic Task Offloading Algorithm Design

In this section, we introduce stochastic optimization techniques to solve problem (16). Using stochastic optimization techniques, we design a NOMA-enabled Dynamic Task Offloading (NDTO) algorithm [28]. The NDTO algorithm can make optimal offloading decisions without the need for future statistics. Then, we decompose the optimization problem into five sub-problems to solve.

3.1. Problem Transformation. In order to maintain the stability of the task queue, we define the Lyapunov function as follows:

$$L(\Theta(t)) = \frac{1}{2}\sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[Q_{s,i}^{l,2}(t) + Q_{s,i}^{o,2}(t) + H_{s,i}^{2}(t)\right], \tag{17}$$

where $\Theta(t) = [\mathbf{Q^l}(\mathbf{t}), \mathbf{Q^o}(\mathbf{t}), \mathbf{H}(\mathbf{t})]$ indicates the queue backlog vector.

Then, the Lyapunov drift function is given as follows:

$$\Delta(\Theta(t)) = \mathbf{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\}. \tag{18}$$

Our goal is to keep the task queue stable while minimizing energy consumption. To this end, we give the Lyapunov drift plus penalty function as follows:

$$\Delta_V(\Theta(t)) = \Delta(\Theta(t)) + V\mathbf{E}\{E(t)|\Theta(t)\}, \tag{19}$$

where V is the trade-off parameter.

We can minimize energy consumption by minimizing the upper bound of (19). The following theorem gives the upper bound of (19).

Theorem 1. *The upper bound of the drift plus penalty function is shown below.*

$$\begin{aligned}
\Delta(\Theta(t)) + V\mathbf{E}\{E(t)|\Theta(t)\} \le Z + \mathrm{VE}\{E(t)|\Theta(t)\} &+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s}Q_{s,i}^{l}(t)\left[\left(1 - I_{s,i}(t)\right)A_{s,i}(t) - D_{s,i}^{l}(t)\right]|\Theta(t)\right\}\\
&+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s}Q_{s,i}^{o}(t)\left[I_{s,i}(t)A_{s,i}(t) - D_{s,i}^{o}(t)\right]|\Theta(t)\right\}\\
&+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s}H_{s,i}(t)\left[D_{s,i}^{o}(t) - \mu_{s,i}^{l}(t) - \mu_{s,i}^{o}(t)\right]|\Theta(t)\right\},
\end{aligned} \tag{20}$$

where Z is a constant.

Proof. From $([a-b]^+ + c)^2 \le a^2 + b^2 + c^2 + 2a(c-b)$, we can get:

$$\begin{aligned} Q_{s,i}^{l,2}(t+1) &\le Q_{s,i}^{l,2}(t) + \left[\left(1 - I_{s,i}(t)\right)A_{s,i}(t)\right]^2 \\ &+ D_{s,i}^{l,2}(t) + 2Q_{s,i}^{l}(t)\left[\left(1 - I_{s,i}(t)\right)A_{s,i}(t) - D_{s,i}^{l}(t)\right], \end{aligned} \tag{21}$$

$$\begin{aligned} Q_{s,i}^{o,2}(t+1) &\le Q_{s,i}^{o,2}(t) + \left[I_{s,i}(t)A_{s,i}(t)\right]^2 \\ &+ D_{s,i}^{o,2}(t) + 2Q_{s,i}^{o}(t)\left[I_{s,i}(t)A_{s,i}(t) - D_{s,i}^{o}(t)\right]. \end{aligned} \tag{22}$$

$$\begin{aligned} H_{s,i}^{2}(t+1) &\le H_{s,i}^{2}(t) + \left[\mu_{s,i}^{l}(t) + \mu_{s,i}^{o}(t)\right]^2 \\ &+ D_{s,i}^{o,2}(t) + 2H_{s,i}(t)\left[D_{s,i}^{o}(t) - \mu_{s,i}^{l}(t) - \mu_{s,i}^{o}(t)\right]. \end{aligned} \tag{23}$$

Then, combining (17) and (18), we can get:

$$\begin{aligned} \Delta_V(\Theta(t)) \le Z + V\mathbf{E}\{E(t)|\Theta(t)\} &+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s} Q_{s,i}^{l}(t)\left[\left(1 - I_{s,i}(t)\right)A_{s,i}(t) - D_{s,i}^{l}(t)\right]|\Theta(t)\right\} \\ &+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s} Q_{s,i}^{o}(t)\left[I_{s,i}(t)A_{s,i}(t) - D_{s,i}^{o}(t)\right]|\Theta(t)\right\} \\ &+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s} H_{s,i}(t)\left[D_{s,i}^{o}(t) - \mu_{s,i}^{l}(t) - \mu_{s,i}^{o}(t)\right]|\Theta(t)\right\}, \end{aligned} \tag{24}$$

where $Z = (1/2)\sum_{s=1}^{S}\sum_{i=1}^{N_s}\left\{(A_{s,i}^{\max}(t)^2) + (f_{s,i}^{l,\max}(t)\tau/\varphi_{s,i})^2 + 2(R_{s,i}^{\max}(t)\tau)^2 + [f_{s,i}^{e}(t)\tau/\varphi_e + R_s^{\max}(t)\tau]^2\right\}$. □

3.2. NOMA-Enabled Dynamic Task Offloading Algorithm. In this section, we propose the NDTO algorithm to minimize the upper bound of (19). We define $\mathcal{X}(t) = \{I(t), f^l(t), D^o(t), \mu^l(t), \mu^o(t)\}$. Since Z is a constant, the upper bound minimization problem can be formulated as follows:

$$\min_{\mathcal{X}(t)} \left\{ \begin{array}{l} \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[Q_{s,i}^{l}(t)\left(1 - I_{s,i}(t)\right)A_{s,i}(t) + Q_{s,i}^{o}(t)I_{s,i}(t)A_{s,i}(t)\right] \\ \quad + \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[V\xi_{s,i}f_{s,i}^{l,3}(t)\tau - Q_{s,i}^{l}(t)\dfrac{f_{s,i}^{l}(t)\tau}{\varphi_{s,i}}\right] \\ + \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[VP_{s,i}(t)\dfrac{D_{s,i}^{o}(t)}{R_{s,i}(t)} - Q_{s,i}^{o}(t)D_{s,i}^{o}(t) + H_{s,i}(t)D_{s,i}^{o}(t)\right] \\ \quad + \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[Vl_1\mu_{s,i}^{l}(t) - H_{s,i}(t)\mu_{s,i}^{l}(t)\right] \\ + \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[\mathrm{VP}_s(t)\dfrac{\mu_{s,i}^{o}(t)}{R_s(t)} - H_{s,i}(t)\mu_{s,i}^{o}(t) + Vl_2\mu_{s,i}^{o}(t)\right], \end{array} \right\}$$

$$
\begin{aligned}
&C1\text{: } I_{s,i}(t) \in \{0,1\},\\
&C2\text{: } 0 \le f_{s,i}^{l}(t) \le f_{s,i}^{l,\max},\\
&C3\text{: } 0 \le D_{s,i}^{o}(t) \le R_{s,i}(t)\tau,\\
s.\text{t.}\quad &C4\text{: } D_{s,i}^{o}(t) \le Q_{s,i}^{o}(t),\\
&C5\text{: } 0 \le \sum_{i=1}^{N_s} \mu_{s,i}^{l}(t) \le \frac{f_{s,i}^{e}(t)\tau}{\varphi_e}, \quad \forall s \in \mathcal{S},\\
&C6\text{: } 0 \le \mu_{s,i}^{o}(t) \le R_s(t)\tau, \quad \forall s \in \mathcal{S}.
\end{aligned}
\tag{25}
$$

In Problem (25), its decision variables are $I(t)$, $f^{l}(t)$, $D^{o}(t)$, $\mu^{l}(t)$, and $\mu^{o}(t)$. Because the decision variables are decoupled, we can decompose Problem (25) into five corresponding sub-problems. Below, we will give the optimal solution for each sub-problem.

3.3. Offloading Decision. By separating out the part related to $I_{s,i}(t)$ in Problem (25), the offloading decision sub-problem is given as follows:

$$
\begin{aligned}
&\min_{I(t)} \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[Q_{s,i}^{l}(t)\left(1 - I_{s,i}(t)\right)A_{s,i}(t) + Q_{s,i}^{o}(t)I_{s,i}(t)A_{s,i}(t)\right],\\
&s.\text{t.}\quad I_{s,i}(t) \in \{0,1\}.
\end{aligned}
\tag{26}
$$

Problem (26) is a zero-one integer programming problem. Therefore, we give the optimal solution for the offloading decision as follows:

$$
I_{s,i}^{*}(t) = \begin{cases} 0, & Q_{s,i}^{o}(t) \ge Q_{s,i}^{l}(t),\\ 1, & \text{otherwise.} \end{cases}
\tag{27}
$$

3.4. CPU Frequency for IoT Devices. The optimal CPU frequency of IoT devices in each time slot can be obtained by solving the following sub-problems:

$$
\begin{aligned}
&\min_{f^{l}(t)} \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[V\xi_{s,i}f_{s,i}^{l,3}(t)\tau - Q_{s,i}^{l}(t)\frac{f_{s,i}^{l}(t)\tau}{\varphi_{s,i}}\right],\\
&s.\text{t.}\quad 0 \le f_{s,i}^{l}(t) \le f_{s,i}^{l,\max}.
\end{aligned}
\tag{28}
$$

Problem (28) is a convex optimization problem. Its first derivative function is $3V\xi_{s,i}f_{s,i}^{l,2}(t)\tau - Q_{s,i}^{l}(t)\tau/\varphi_{s,i}$. When its first derivative function is equal to 0, we can get $f_{s,i}^{l}(t) = \sqrt{(Q_{s,i}^{l}(t)/3V\xi_{s,i}\varphi_{s,i})}$. Then, we can get its second derivative function as $6V\xi_{s,i}f_{s,i}^{l}(t)\tau \ge 0$. Thus, the optimal solution of $f_{s,i}^{l}(t)$ as follows:

$$
f_{s,i}^{l,*}(t) = \begin{cases} \sqrt{\dfrac{Q_{s,i}^{l}(t)}{3V\xi_{s,i}\varphi_{s,i}}}, & 0 \le \sqrt{\dfrac{Q_{s,i}^{l}(t)}{3V\xi_{s,i}\varphi_{s,i}}} \le f^{*},\\ f^{*}, & \text{otherwise,} \end{cases}
\tag{29}
$$

where $f^{*} = \min\left\{Q_{s,i}^{l}(t)\varphi s, i/\tau, f_{s,i}^{l,\max}\right\}$.

3.5. Offloading Computation for IoT Devices. The sub-problem of offloading computation for IoT devices is described as follows:

$$
\begin{aligned}
&\min_{D^{o}(t)} \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[\frac{\mathrm{VP}_{s,i}(t)}{R_{s,i}(t) - Q_{s,i}^{o}(t) + H_{s,i}(t)}\right]D_{s,i}^{o}(t).\\
&s.\text{t.}\quad \begin{aligned} &0 \le D_{s,i}^{o}(t) \le R_{s,i}(t)\tau,\\ &D_{s,i}^{o}(t) \le Q_{s,i}^{o}(t). \end{aligned}
\end{aligned}
\tag{30}
$$

Problem (30) is a linear programming problem. We define $W_{s,i}(t) = (\mathrm{VP}_{s,i}(t)/R_{s,i}(t)) - Q_{s,i}^{o}(t) + H_{s,i}(t)$. Thus, the offloading computation $D_{s,i}^{o}(t)$ of the IoT device $U_{s,i}$ is calculated as follows:

$$
D_{s,i}^{o,*}(t) = \begin{cases} \min\left\{R_{s,i}(t)\tau, Q_{s,i}^{o}(t)\right\}, & W_{s,i}(t) \le 0,\\ 0, & \text{otherwise.} \end{cases}
\tag{31}
$$

3.6. SBS Local Computation Allocation. The SBS local computation allocation sub-problem is formulated as follows:

$$
\begin{aligned}
&\min_{\mu^{l}(t)} \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[Vl_1 - H_{s,i}(t)\right]\mu_{s,i}^{l}(t).\\
&s.\text{t.}\quad 0 \le \sum_{i=1}^{N_s}\mu_{s,i}^{l}(t) \le \frac{f_{s,i}^{e}(t)\tau}{\varphi_e}, \quad \forall s \in \mathcal{S}.
\end{aligned}
\tag{32}
$$

Problem (32) can be viewed as a knapsack problem, where the weights of $\mu_{s,i}^{l}(t)$ are $Vl_1 - H_{s,i}(t)$. The specific steps to solve this problem are given as follows:

(1) We initialize the amount of tasks that SBS s can handle in the time slot t as $C_s^l = f_{s,i}^e(t)\tau/\varphi_e$.

(2) For each SBS, the IoT devices in its service area are sorted according to the value of $Vl_1 - H_{s,i}(t)$ from low to high.

(3) The SBS s sequentially assigns the size of the SBS local computation to the IoT device $U_{s,i}$ according to the sorting results, as follows:

$$\mu_{s,i}^{l,*}(t) = \begin{cases} \min\left\{H_{s,i}(t), C_s^l\right\}, & i = i^*, \quad Vl_1 - H_{s,i}(t) \le 0, \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

where $i^* = \arg\min_{i \in \mathcal{N}^s}\left\{Vl_1 - H_{s,i}(t)\right\}$.

(4) Updating the amount of tasks that SBS s can still process in the time slot t is $C_s^l = C_s^l - \mu_{s,i}^{l,*}(t)$.

3.7. SBS Offloading Computation Allocation. The SBS Offloading computation allocation sub-problem is formulated as follows:

$$\min_{\mu_i^o(t)} \sum_{s=1}^{S}\sum_{i=1}^{N_s}\left[\frac{VP_s(t)}{R_s(t)} - H_{s,i}(t) + Vl_2\right]\mu_{s,i}^{o}(t), \tag{34}$$

$$s.t. \quad 0 \le \sum_{i=1}^{N_s}\mu_{s,i}^{o}(t) \le R_s(t)\tau, \quad \forall s \in \mathcal{S}.$$

We define $Y_{s,i}(t) = \mathrm{VP}_s(t)/R_s(t) - H_{s,i}(t) + Vl_2$. Similar to solving Problem (32), the specific steps to solve this problem are given as follows:

(1) We initialize the amount of tasks that SBS s can offload in time slot t as $C_s^o = R_s(t)\tau$.

(2) For each SBS, the IoT devices in its service area are sorted according to the value of $Y_{s,i}(t)$ from low to high.

(3) The SBS s sequentially assigns the size of the SBS local computation to the IoT device $U_{s,i}$ according to the sorting results, as follows:

$$\mu_{s,i}^{o,*}(t) = \begin{cases} \min\left\{H_{s,i}(t) - \mu_{s,i}^{l,*}(t), C_s^o\right\}, & i = i^*, Y_{s,i}(t) \le 0, \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

where $i^* = \arg\min_{i \in \mathcal{N}^s}\left\{Y_{s,i}(t)\right\}$.

(4) Updating the amount of tasks that SBS s can still offload in time slot t is $C_s^o = C_s^o - \mu_{s,i}^{o,*}(t)$.

Details of the NDTO algorithm are given in Algorithm 1.

4. Analysis of NOMA-Enabled Dynamic Task Offloading Algorithm

In this section, we analyze the performance of the NDTO algorithm mathematically.

We give the following lemma to demonstrate the performance of the NDTO algorithm.

Lemma 1. *No matter how the task arrival rate λ changes, there is always an optimal decision π^* that is independent of the queue length. From this, we can get:*

$$\begin{aligned} &E\left\{\left\{\overline{E}^{\pi^*}(t)\right\} = E^*(\lambda),\right. \\ &E\left\{\left(1 - I_{s,i}^{\pi^*}(t)\right)A_{s,i}^{\pi^*}(t)\right\} \le E\left\{D_{s,i}^{l,\pi^*}(t)\right\}, \\ &E\left\{D_{s,i}^{o,\pi^*}(t)\right\} \le E\left\{\mu_{s,i}^{l,\pi^*}(t) + \mu_{s,i}^{o,\pi^*}(t)\right\}, \end{aligned} \tag{36}$$

where $E^*(\lambda)$ is minimum energy consumption with λ.

Proof. Caratheodory's theorem can be used to prove Lemma 1. Here, we will not repeat them. □

It is worth noting that the task arrival rate is capped. Therefore, the upper limit of the system energy consumption is denoted as $\widehat{E}$, and the lower limit is denoted as $\check{E}$. We define $\overline{L} = \lim_{T\to\infty}(1/T)\sum_{s=1}^{S}\sum_{i=1}^{N_s}[Q_{s,i}^l(t) + Q_{s,i}^o(t) + H_{s,i}(t)]$. According to Lemma 1, we give Theorem 2 to analyze the performance of NDTO. □

Theorem 2. *For any value of V, when the task arrival rate is $\lambda + \epsilon$, according to NDTO, we have:*

$$E^{\mathrm{NDTO}} \le E^* + \frac{Z}{V}, \tag{37}$$

$$\overline{L} \le \frac{Z + V(\widehat{E} - \check{E})}{\epsilon}. \tag{38}$$

Proof. Based on Lemma 1, for any decision π and task arrival rate $\lambda + \epsilon$, we can get:

$$\begin{aligned} &\mathbf{E}\left\{\overline{E}^{\pi}(t)\right\} = E^*(\lambda + \epsilon), \\ &\mathbf{E}\left\{\left(1 - I_{s,i}^{\pi}(t)\right)A_{s,i}^{\pi}(t)\right\} + \epsilon \le \mathbf{E}\left\{D_{s,i}^{l,\pi}(t)\right\}, \\ &\mathbf{E}\left\{I_{s,i}^{\pi}(t)A_{s,i}^{\pi}(t)\right\} + \epsilon \le \mathbf{E}\left\{D_{s,i}^{o,\pi}(t)\right\}, \\ &\mathbf{E}\left\{D_{s,i}^{o,\pi}(t)\right\} + \epsilon \le \mathbf{E}\left\{\mu_{s,i}^{l,\pi}(t) + \mu_{s,i}^{o,\pi}(t)\right\}. \end{aligned} \tag{39}$$

Our NDTO algorithm is to minimize the value of (19). Thus, for decision π, we have:

Input: $Q^l_{s,i}(t) \leftarrow 0$, $Q^o_{s,i}(t) \leftarrow 0$, $H_{s,i}(t) \leftarrow 0$, $A_{s,i}(t)$
Output: $I_{s,i}(t)$, $f^l_{s,i}(t)$, $D^o_{s,i}(t)$, $\mu^l_{s,i}(t)$, $\mu^o_{s,i}(t)$
(1) **for all** $s \in \mathcal{S}$ **do**
(2) **for all** $i \in \mathcal{N}^s$ **do**
(3) Set the $I_{s,i}(t)$ according to (27).
(4) **end for**
(5) **end for**
(6) **for all** $s \in \mathcal{S}$ **do**
(7) **for all** $i \in \mathcal{N}^s$ **do**
(8) Calculate the value of $\sqrt{Q^l_{s,i}(t)/3V\xi_{s,i}\varphi_{s,i}}$.
(9) Set the $f^l_{s,i}(t)$ according to (29).
(10) **end for**
(11) **end for**
(12) **for all** $s \in \mathcal{S}$ **do**
(13) **for all** $i \in \mathcal{N}^s$ **do**
(14) Calculate the value of $\mathrm{VP}_{s,i}(t)/R_{s,i}(t) - Q^o_{s,i}(t) + H_{s,i}(t)$.
(15) Set the $D^o_{s,i}(t)$ according to (31).
(16) **end for**
(17) **end for**
(18) **for all** $s \in \mathcal{S}$ **do**
(19) Initialize $C^l_s = f^e_{s,i}(t)\tau/\varphi_e$.
(20) Sort IoT devices from low to high according to the value of $Vl_1 - H_{s,i}(t)$.
(21) **for all** $i \in \mathcal{N}^s$ **do**
(22) Set the $\mu^l_{s,i}(t)$ according to (33).
(23) $C^l_s = C^l_s - \mu^{l,*}_{s,i}(t)$.
(24) **end for**
(25) **end for**
(26) **for all** $s \in \mathcal{S}$ **do**
(27) Initialize $C^o_s = R_s(t)\tau$.
(28) Sort IoT devices from low to high according to the value of $\mathrm{VP}_s(t)/R_s(t) - H_{s,i}(t) + Vl_2$.
(29) **for all** $i \in \mathcal{N}^s$ **do**
(30) Set the $\mu^o_{s,i}(t)$ according to (35).
(31) $C^o_s = C^o_s - \mu^{o,*}_{s,i}(t)$.
(32) **end for**
(33) **end for**
(34) **for all** $s \in \mathcal{S}$ **do**
(35) **for all** $i \in \mathcal{N}^s$ **do**
(36) Update the values of queues $Q^l_{s,i}(t)$, $Q^o_{s,i}(t)$ and $H_{s,i}(t)$.
(37) **end for**
(38) **end for**

Algorithm 1: NOMA-enabled Dynamic Task Offloading (NDTO) Algorithm.

$$\begin{aligned}\Delta_V(\Theta(t)) \le{}& Z + V\mathbf{E}\{E(t)|\Theta(t)\} \\ &+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s} Q^l_{s,i}(t)\left[\left(1 - I_{s,i}(t)\right)A^{\pi}_{s,i}(t) - D^{l,\pi}_{s,i}(t)\right]\right\} \\ &+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s} Q^o_{s,i}(t)\left[I_{s,i}(t)A^{\pi}_{s,i}(t) - D^{o,\pi}_{s,i}(t)\right]\right\} \\ &+ \mathbf{E}\left\{\sum_{s=1}^{S}\sum_{i=1}^{N_s} H_{s,i}(t)\left[D^{o,\pi}_{s,i}(t) - \mu^{l,\pi}_{s,i}(t) - \mu^{o,\pi}_{s,i}(t)\right]\right\},\end{aligned} \tag{40}$$

Combining (39) and (40), we can get:

$$\begin{aligned}\mathbf{E}\{U(\Theta(t+1)) - U\Theta(t)\} + V\mathbf{E}(E(t)) \le{}& Z + \mathrm{VE}^*(\lambda + \epsilon) \\ &-\epsilon\sum_{s=1}^{S}\sum_{i=1}^{N_s}\mathbf{E}\left\{Q^l_{s,i}(t) + Q^o_{s,i}(t) + H_{s,i}(t)\right\}.\end{aligned} \tag{41}$$

Generally speaking, $L(\Theta(0)) = 0$. Taking expectations on both sides of (41) and giving the cumulative sum over time, we obtain:

$$\begin{aligned}V\sum_{t=1}^{T-1}\mathbf{E}(E(t)) \le{}& \mathrm{ZT} + \mathrm{VTE}^*(\lambda + \epsilon) \\ &-\epsilon\sum_{t=1}^{T-1}\sum_{s=1}^{S}\sum_{i=1}^{N_s}\mathbf{E}\left\{Q^l_{s,i}(t) + Q^o_{s,i}(t) + H_{s,i}(t)\right\}.\end{aligned} \tag{42}$$

Since $Q^l_{s,i}(t)$, $Q^o_{s,i}(t)$, $H_{s,i}(t)$ and ϵ are nonnegative, we have:

$$V\sum_{t=1}^{T-1}\mathbf{E}(E(t)) \le \mathrm{ZT} + \mathrm{VTE}^{*}(\lambda+\epsilon). \tag{43}$$

Then, divide both sides of (43) by VT. When $\epsilon \longrightarrow 0$, $T \longrightarrow \infty$, we can get (37).

With (42), we also get:

$$\epsilon\sum_{t=1}^{T-1}\sum_{s=1}^{S}\sum_{i=1}^{N_s}\mathbf{E}\left\{Q_{s,i}^{l}(t)+Q_{s,i}^{o}(t)+H_{s,i}(t)\right\} \le \mathrm{ZT} + \mathrm{VTE}^{*}(\lambda+\epsilon) - V\sum_{t=1}^{T-1}\mathbf{E}(E(t)). \tag{44}$$

Since $\mathbf{E}(E(t))$ is nonnegative, we have:

$$\epsilon\sum_{t=1}^{T-1}\sum_{s=1}^{S}\sum_{i=1}^{N_s}\mathbf{E}\left\{Q_{s,i}^{l}(t)+Q_{s,i}^{o}(t)+H_{s,i}(t)\right\} \le \mathrm{ZT} + \mathrm{VT}(\widehat{E}-\check{E}). \tag{45}$$

By dividing (45) by ϵT, when $T \longrightarrow \infty$, we get (38). □

5. Evaluation

In this section, we evaluate the performance of the NDTO algorithm through a series of experiments.

5.1. Experiment Setup. In the experiments, we consider a system with 100 IoT devices, 5 SBSs, and 1 MBS. Within each time slot, the task arrival rate satisfies a uniform distribution, i.e., $A_{s,i}(t) \sim U[0, 1.4]$ Mb. For each IoT device, the maximum CPU frequency is 1 GHz, and the transmit power is uniformly distributed, which is $P_{s,i}(t) \sim U[0.01, 0.1]w$. For each SBS, the transmit power is uniformly distributed, which is $P(t) \sim U[1, 5]w$. We set the wireless channel bandwidth to 10^6 Hz, the channel noise power to $10^{-13}w$, and the number of CPU cycles required to process 1-bit data to 1000. Furthermore, the effective switched capacitance is 10^{-27}, the time slot length is 1 s, and $l_1 = 10^{-4}$ J/b [29, 30].

5.2. Parameter Analysis

5.2.1. Impact of Parameter V. Figure 1 shows the effect of different parameter V on energy consumption. It can be seen that energy consumption decreases as V increases, which is consistent with the result of (37). This is because the greater the value of V, the greater the weight of the system energy consumption. Therefore, the NDTO algorithm will dynamically adjust the offloading strategy to reduce energy consumption. Figure 2 shows the effect of different parameter V on the queue length. It can be seen that the queue length increases as V increases, which is consistent with the result of (38). Combining Figures 1 and 2, it can be seen that the NDTO algorithm can achieve a trade-off between energy consumption and queue length.

5.2.2. Impact of Task Arrival Rate. Figures 3 and 4 show the effect of different task arrival rates on energy consumption and queue length. We set the task arrival rate to $\alpha \cdot A_{s,i}(t)$, where $\alpha =$ 0.8, 1.0, and 1.2. As can be seen from Figure 3, the

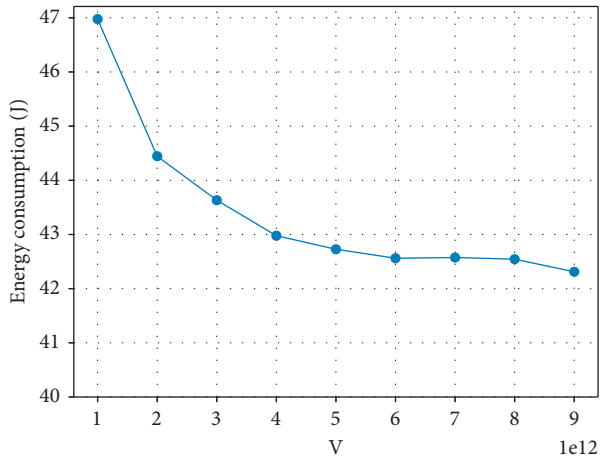

Figure 1: Energy consumption with different values of V.

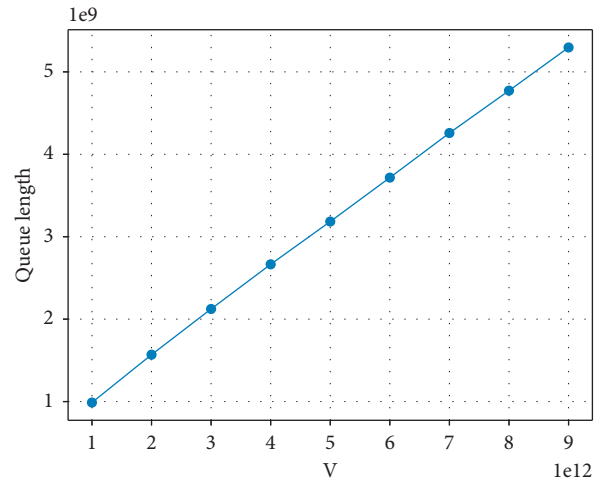

Figure 2: Queue length with different values of V.

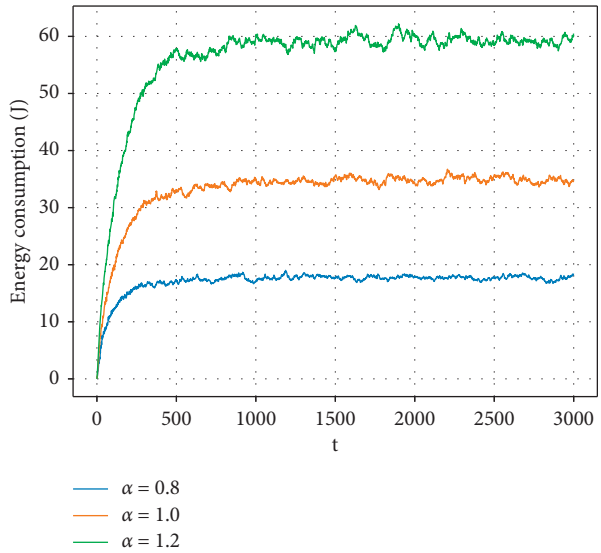

Figure 3: Energy consumption with different task arrival rates.

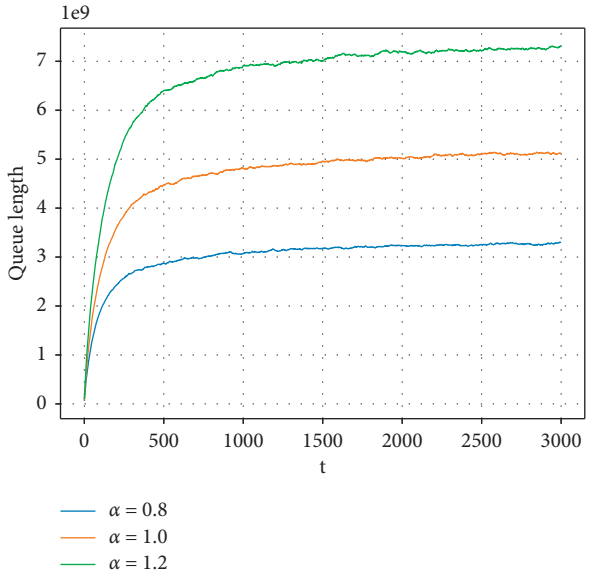

Figure 4: Queue length with different task arrival rates.

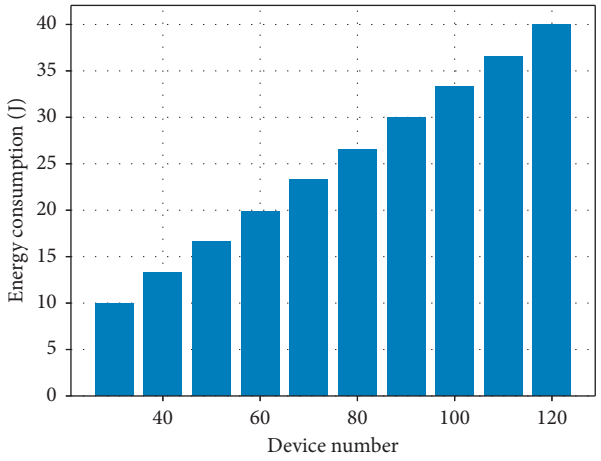

Figure 5: Energy consumption with different number of IoT devices.

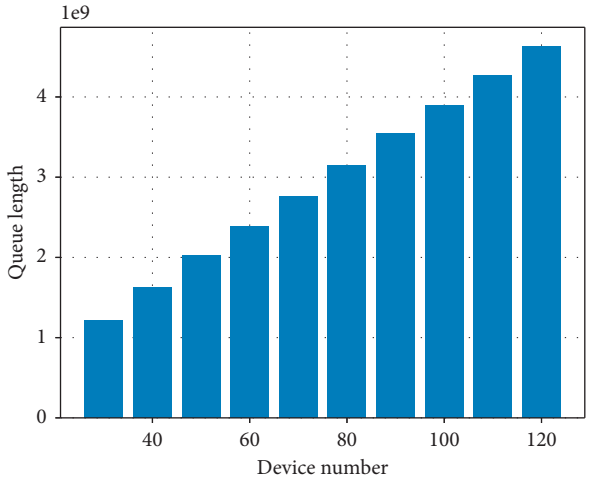

Figure 6: Queue length with different number of IoT devices.

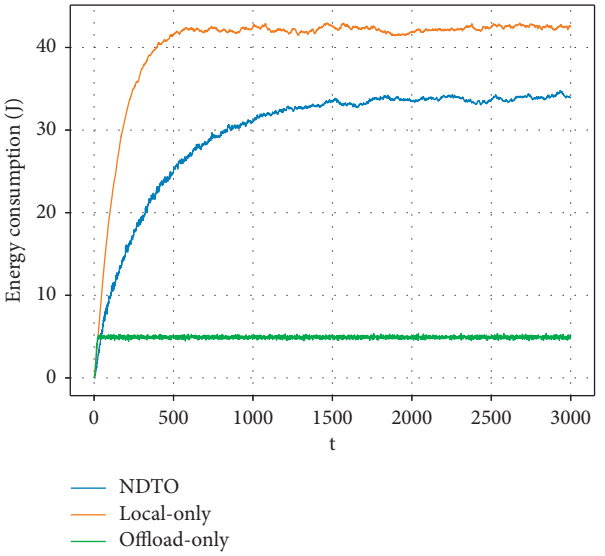

Figure 7: Energy consumption with different algorithms.

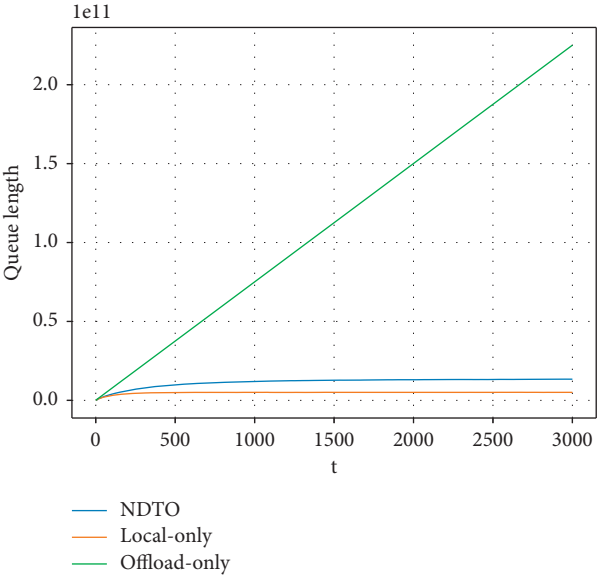

Figure 8: Queue length with different algorithms.

higher the task arrival rate, the higher the system energy consumption. An increase in the task arrival rate leads to an increase in computing tasks, and thus an increase in energy consumption. As can be seen from Figure 4, the higher the task arrival rate, the larger the queue length. Because the transmission capacity and computing power of IoT devices will not change due to the change of task arrival rate. Taken together, the NDTO algorithm can adapt to changes of the task arrival rate to achieve a trade-off between energy consumption and queue length.

5.2.3. Impact of Number of IoT Devices. Figures 5 and 6 show the effect of the number of IoT devices on energy

consumption and queue length. We set the number of IoT devices to increase from 30 to 120 in increments of 10 each. Figure 5 shows that energy consumption increases as the number of devices increases. Because the amount of generated tasks increases with the number of devices, energy consumption increases accordingly. But the computing power and transmission capacity of SBS are limited. Therefore, as shown in Figure 6, the queue length increases with the number of devices.

5.3. Comparison Experiment. We compare the NDTO algorithm with two other baseline algorithms to further evaluate the performance of NDTO.

(i) **Local-only Computation Algorithm.** The tasks generated by IoT devices are all placed on the local computing queue.

(ii) **Offload-only Computation Algorithm.** The tasks generated by IoT devices are all placed on the local offloading queue.

Figures 7 and 8 show the relationship between energy consumption and queue length for different algorithms, respectively. As can be seen from Figure 7, the energy consumption of the offload-only computation algorithm is the lowest, followed by the NDTO algorithm, and finally the local-only computation algorithm.

It can be seen from Figure 8 that the queue length of the offload-only computation algorithm increases linearly, which will lead to instability of the task queue. At the same time, we can see that the change in queue length of the NDTO algorithm and the local-only computation algorithm is basically the same. Taken together, our NDTO algorithm is more advantageous in minimizing system energy consumption and maintaining task queue stability.

6. Related Work

In recent years, with the development and popularization of IoT and wireless communication technologies, the number of IoT devices has exploded [31, 32]. In addition, the proliferation of mobile devices has facilitated the development of computation-intensive applications [33, 34]. The emergence of the MEC paradigm has attracted great attention from industry and academia. Mao et al. [35] proposed an online joint radio and computational resource management algorithm for multi-user MEC systems. The algorithm implements a trade-off between power consumption and execution delay. Chen et al. [36] studied the problem of energy-saving task offloading in MEC systems. Then, they propose an energy-efficient dynamic offloading algorithm. The algorithm can minimize the transmission energy consumption while limiting the queue length. Huang et al. [37] considered wirelessly powered MEC networks with the binary offload. They propose an online algorithm that adapts task offloading decisions and radio resource allocation to time-varying radio channel conditions. The above work only considers the situation that one base station provides services for multiple users, but does not consider the situation that one MBS covers multiple SCNs.

The problem of one MBS covering multiple SCNs has also been extensively studied. Guo et al. [38] studied the problem of energy-efficient computation offloading in MEC systems with SCN. Wherein each user equipment not only decided whether to perform offloading, but also decided to which base station to offload. Wang et al. [39] studied the problem of computational offloading and content caching in heterogeneous wireless SCNs. Then, they proposed an ADMM-based distributed algorithm to solve the proposed problem. The above work is based on OMA technology. But with the explosive growth of IoT devices, OMA technology can no longer meet user needs.

Therefore, NOMA technology is applied in SCN as a new generation of mobile communication technology. Yang et al. [40] considered the task offloading problem in NOMA-based small cell MEC networks. Meanwhile, a hybrid genetic hill climbing algorithm is proposed to minimize the weighted total cost of energy consumption and latency. Most of these works can be predicted based on task arrival or channel state. However, task arrival and channel state are affected by many factors, and it is unrealistic to achieve accurate prediction.

In our work, we study the problem of NOMA-based dynamic task offloading in heterogeneous networks. We specify stochastic problems to minimize the total energy consumption of the system. Meanwhile, we transform it into five sub-problems using stochastic optimization techniques. Finally, we design an NDTO algorithm to solve this problem.

7. Conclusion

In this article, we study the NOMA-enabled dynamic task offloading problem in MEC. We describe it as a stochastic optimization problem, where the goal is to minimize system energy consumption while keeping the task queue stable. Using stochastic optimization techniques, we decompose this problem into five deterministic sub-problems to solve. Then, an NDTO algorithm is designed to solve this problem. Meanwhile, we evaluate the performance of the NDTO algorithm through mathematical analysis and experiments. The experimental results show that the NDTO algorithm is effective in minimizing energy consumption while maintaining queue stability.

Data Availability

The data used to support the findings of this study can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China (no. 61902029), R&D Program of Beijing Municipal Education Commission (no.

KM202011232015), Project for Acceleration of University Classification Development (nos. 5112211036, 5112211037, and 5112211038). This work was supported in part by the National Natural Science Foundation of China, under Grants 61902029, 61972414, and 61973161, the Excellent Talents Projects of Beijing, under Grant 9111923401, the Scientific Research Project of Beijing Municipal Education Commission, under Grant KM202011232015, the Beijing Nova Program, under Grant Z201100006820082, the Beijing Natural Science Foundation, under Grant 4202066, and the Fundamental Research Funds for Central Universities, under Grant 2462018YJRC040.

References

[1] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8099–8110, 2020.

[2] J. Huang, Z. Tong, and Z. Feng, "Geographical poi recommendation for internet of things: a federated learning approach using matrix factorization," *International Journal of Communication Systems*, 2022.

[3] L. Qi, Y. Yang, X. Zhou, W. Rafique, and J. Ma, "Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6503–6511, 2022.

[4] A. Sultana, I. Woungang, A. Anpalagan, L. Zhao, and L. Ferdouse, "Efficient resource allocation in scma-enabled device-to-device communication for 5 g networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5343–5354, 2020.

[5] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Toffee: task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1634–1644, 2021.

[6] G. Qu, H. Wu, R. Li, and P. Jiao, "Dmro: a deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.

[7] J. Huang, C. Zhang, and J. Zhang, "A multi-queue approach of energy efficient task scheduling for sensor hubs," *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 242–247, 2020.

[8] Q. Li, L. Zhao, J. Gao, H. Liang, L. Zhao, and X. Tang, "Smdp-based coordinated virtual machine allocations in cloud-fog computing systems," *IEEE Internet of Things Journal*, vol. 5, no. 3, p. 1977, 2018.

[9] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "Eedto: an energy-efficient dynamic task offloading algorithm for blockchain-enabled iot-edge-cloud orchestrated computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2163–2176, 2021.

[10] J. Xu, D. Li, and W. Gu, "Uav-assisted task offloading for iot in smart buildings and environment via deep reinforcement learning," *Building and Environment*, vol. 2022.

[11] Y. Chen, W. Gu, and K. Li, "Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning," *International Journal of Communication Systems*, 2022.

[12] X. Xue, S. Wang, L. Zhang, Z. Feng, and Y. Guo, "Social learning evolution (sle): computational experiment-based modeling framework of social manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3343–3355, 2019.

[13] Y Xie, F.-X. Gui, W.-J. Wang, and C.-F Chien, "A two-stage multi-population genetic algorithm with heuristics for workflow scheduling in heterogeneous distributed computing environments," *IEEE Transactions on Cloud Computing*, vol. 1, p. 1, 2022.

[14] J. Huang, B. Lv, Y. Wu, Y. Chen, and X. Shen, "Dynamic admission control and resource allocation for mobile edge computing enabled small cell network," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, p. 1964, 2022.

[15] A. Nasser, O. Muta, M. Elsabrouty, and H. Gacanin, "Interference mitigation and power allocation scheme for downlink mimo–noma hetnet," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6805–6816, 2019.

[16] Y Xie, Y. Sheng, M. Qiu, and F. Gui, "An adaptive decoding biased random key genetic algorithm for cloud workflow scheduling," *Engineering Applications of Artificial Intelligence*, vol. 112, Article ID 104879, 2022.

[17] K. A. Hafeez, L. Zhao, J. W. Mark, S. Xuemin, and N. Zhisheng, "Distributed multichannel and mobility-aware cluster-based MAC protocol for vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 8, pp. 3886–3902, 2013.

[18] H. Dai, Y. Xu, G. Chen et al., "Rose: robustly safe charging for wireless power transfer," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2180–2197, 2022.

[19] R. Gu, Y. Chen, S. Liu et al., "Liquid: intelligent resource estimation and network-efficient scheduling for deep learning jobs on distributed gpu clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2808–2820, 2022.

[20] H. Dai, X. Wang, X. Lin et al., "Placing wireless chargers with limited mobility," *IEEE Transactions on Mobile Computing*, p. 1, 2021.

[21] R. Gu, K Zhang, Z. Xu et al., "Fluid: dataset abstraction and elastic acceleration for cloud-native deep learning training jobs," in *Proceedings of the 38th IEEE International Conference on Data Engineering*, pp. 2183–2196, 2022.

[22] L. Qi, C. Hu, X. Zhang et al., "Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4159–4167, 2021.

[23] Y. Chen, F. Zhao, Y. Lu, and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Science and Technology*, 2022.

[24] L. Qi, W. Lin, X. Zhang, W. Dou, X. Xu, and J. Chen, "A correlation graph based approach for personalized and compatible web apis recommendation in mobile app development," *IEEE Transactions on Knowledge and Data Engineering*, p. 1, 2022.

[25] Y. Chen, H. Xing, and Z. Ma, "Cost-efficient edge caching for noma-enabled iot services," China Communications, 2022.

[26] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[27] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.

[28] K. Li, J. Zhao, and J. Hu, "Dynamic energy efficient task offloading and resource allocation for noma-enabled iot in

smart buildings and environment," *Building and Environment*, 2022.

[29] Z. Wen, K. Yang, X. Liu, S. Li, and J. Zou, "Joint offloading and computing design in wireless powered mobile-edge computing systems with full-duplex relaying," *IEEE Access*, vol. 6, pp. 72786–72795, 2018.

[30] Y. Chen, F. Zhao, X. Chen, and Y. Wu, "Efficient multi-vehicle task offloading for mobile edge computing in 6g networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4584–4595, 2022.

[31] J. Huang, M. Wang, Y. Wu, Y. Chen, and X. Shen, "Distributed offloading in overlapping areas of mobile edge computing for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13837–13847, 2022.

[32] J. Li, H. Peng, Y. Cao et al., "Higher-order attribute-enhancing heterogeneous graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, p. 1, 2021.

[33] D. Zhou, X. Xue, and Z. Zhou, "Sle2:the improved social learning evolution model of cloud manufacturing service ecosystem," *IEEE Transactions on Industrial Informatics*, p. 1, 2022.

[34] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao, "Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4925–4934, 2021.

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

[36] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for internet of things," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1050–1060, 2021.

[37] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.

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

[39] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[40] L. Yang, S. Guo, L. Yi, Q. Wang, and Y. Yang, "Noscm: a novel offloading strategy for noma-enabled hierarchical small cell mobile-edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8107–8118, 2021.