Journal of Systems Architecture 143 (2023) 102979

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Lyapunov optimized energy-efficient dynamic offloading with queue length constraints

Jing Mei [a], Longbao Dai [a], Zhao Tong [a,*], Lianming Zhang [a], Keqin Li [b,c,1]

[a] College of Information Science and Engineering, Hunan Normal University in Changsha, Hunan, 410012, China
[b] College of Information Science and Engineering, Hunan University, and National Supercomputing Center in Changsha, Hunan, 410082, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Keywords:
Edge computing
Energy-efficient
Lyapunov optimization
Queue length constraint
System stability

ABSTRACT

With the advent of the Internet of Things (IoT), more computation-intensive applications are migrated to IoT devices. Whereas the battery with limited capacity and the processor with low computing power become the bottlenecks that limit its further development. Mobile edge computing (MEC) provides a promising solution to break through the bottlenecks. Many effective methods are proposed to guide how to offload tasks from IoT devices to MEC to enhance the computing capacity and prolong the battery life of devices. This paper investigates the task offloading problem for a multi-device single-MEC system whose status, such as task arrival rate and channel state, is dynamically changing over time and aims at minimizing the energy consumption of devices and maintaining the system stability in the long term. This problem requires lots of future information about the system, which brings a challenge since it is difficult to obtain future information. Moreover, to improve the system performance with respect to task response time, we define an individual queue length threshold for each IoT device such that the queue length of each device can stabilize around the predefined threshold. To address this problem, we first construct a virtual queue for each device to transform the queue length threshold constraint into the virtual queue stability constraint. Secondly, applying the Lyapunov optimization method, the original problem, which requires future system information, is transformed into a problem that only depends on the information of the current time. Thirdly, a dynamic energy-efficient task offloading algorithm is proposed to optimize the time-average energy consumption while maintaining the queue length constraint. This algorithm generates the offloading decision in real-time without requiring system statistical information. Lastly, simulations are conducted to analyze the effect of different parameters on performance. A group of comparisons are given, showing that the task queue length under the proposed method can be controlled effectively compared with the existing studies.

1. Introduction

Tthe rapid development of Internet of Things (IoT) technology enables billions of IoT devices capable of computation and communication, e.g., mobile devices, wearable devices, etc., to be connected to the Internet via cellular networks [1]. The variety of computation-intensive applications executed on the devices becomes ever-growing [2,3], and processing them always requires powerful computing capability and consumes quite a lot of energy. Whereas due to size limitations, the battery capacity of IoT devices is extremely limited. Moreover, the processor is always of low performance. For these two reasons, it is hard to satisfy the requirements of many applications on computing capability, e.g., augmented reality and image processing [4,5], and the device's battery life is extremely short. A promising solution to alleviate the two intrinsic problems of IoT devices is to offload a part of computation tasks to the nearby servers for processing. Mobile edge computing (MEC) provides such a technology that deploys server resources at the edge of network [6]. Via computation offloading, an IoT device can deal with tasks with high computing capacity requirements, and the battery life of devices is prolonged effectively.

In this paper, we focus on the computation partial offloading problem on a dynamic MEC system with limited resources, with the goal of reducing device energy consumption as much as possible while ensuring execution performance in the long run. There are several challenges to resolving this problem. First, the offloading problem is divided into

* Corresponding author.
E-mail addresses: jingmei@hunnu.edu.cn (J. Mei), 202120293781@hunnu.edu.cn (L. Dai), tongzhao@hunnu.edu.cn (Z. Tong), zlm@hunnu.edu.cn (L. Zhang), lik@newpaltz.edu (K. Li).
[1] Fellow, IEEE.

https://doi.org/10.1016/j.sysarc.2023.102979
Received 29 April 2023; Received in revised form 13 July 2023; Accepted 24 August 2023
Available online 26 August 2023

two sub-problems: task assignment and resource allocation. The purpose of task assignment is to determine how to divide and assign each device's tasks, how much is executed locally, and how much is executed remotely. The purpose of resource allocation is to determine how to allocate the limited resources for each device. The resource allocation decision depends on the number of tasks offloaded by each device, and the task assignment decision of each device depends on the amount of resources allocated to it. The solutions to the two sub-problems are inextricably linked, which brings great difficulty. Second, we aim at reducing the energy consumption of devices while also ensuring long-term execution performance. To achieve long-term energy consumption optimization, task assignment, and resource allocation decisions should be made by combining future states. For devices, the energy consumption of transmitting tasks is affected by the channel state greatly. A better channel condition leads to a higher transmission rate and a shorter transmission time. Consequently, the transmission energy consumption is lowered. In contrast, when the channel condition worsens, the energy efficiency decreases, and transmitting the same amount of tasks might consume more energy. A reasonable strategy is postponing task offloading until the wireless channel becomes better. Under this strategy, future system information regarding task load and resource state must be known in advance. Most existing studies assume that future information is known in advance and kept unchanged, or they can be predicted precisely according to historical statistics. However, due to the system dynamics, this information cannot be predicted precisely [7], which leads to another challenge. Hence, it is necessary to design an effective method to solve the long-term energy-efficient computation offloading problem in dynamic systems [8]. This paper applies the Lyapunov optimization method to solve this problem. Lyapunov optimization is widely used in the optimal control of dynamic systems to optimize the target performance in the long term while maintaining system stability. It can transform the original problem, which requires much future information, into a novel problem that only depends on the information at the current time. Via problem transformation, the optimal offloading decisions can be made in real-time such that the objective performance is optimized in the long run and the system stability is maintained as well. Many studies have applied Lyapunov optimization to solving energy-efficient computation offloading problems, and each of them stressed different aspects. However, most existing studies leveraging Lyapunov optimization only require maintaining the system stability but do not care how long the queue length is at the stable state. In this case, the task execution performance, which significantly depends on the task queue length, would be inferior since the queue length of devices might be very long when the system reaches stability. Motivated by this shortage, we introduce a queue length threshold for each device. Under the queue length constraint, the performance concerning task response time can be guaranteed effectively. Meanwhile, we can control and adjust the system performance artificially by adjusting the threshold settings. Our significant contributions are summarized as follows:

- We consider the computation offloading problem for a dynamic MEC system where the amount of arrived tasks and the wireless channel state change over time. Our goal is to minimize the energy consumption of all devices in the long run. An individual threshold is defined to constrain the queue length of each device to guarantee long-term execution performance concerning task response time. The threshold settings make the task execution performance adjustable.
- We formulate the energy-efficient task offloading problem as a time-average energy minimization problem with queue length and resource constraints. To leverage the Lyapunov optimization method, we construct a virtual queue for each device and transform the queue length constraint into the virtual queue stability constraint. After that, the Lyapunov optimization method is applied to transform the original problem, which relies on future information, into a real-time optimization problem only depending on the information of the current time. An iterative two-stage heuristic algorithm is proposed to solve the real-time optimization problem. By the proposed algorithm, the offloading schemes, i.e., the number of tasks to be processed for each device, the ratio of tasks executed locally and offloaded to MEC, and the resource allocation scheme, are determined in real-time.
- We give a theoretical analysis of the performance of the proposed method and do a series of experiments to observe the effect of different parameters on the performance in terms of queue length and energy consumption. Besides, we conduct a group of simulations to verify the adaptability of the method to dynamic performance requirements by changing the queue length threshold at run time. Lastly, a group of comparative experiments is given to verify the performance improvement of our algorithm compared with three strategies.

The proposed method can be applied to edge computing in agriculture and can help improve agricultural production efficiency, decision support, and resource management. By deploying sensors and IoT devices, bringing edge computing to agriculture can enable smart agriculture. Sensors can collect environmental data from farmland in real-time, such as soil moisture, temperature, and light, and then process and analyze the data through edge devices. There is no clear deadline for processing these data, but only a certain period of time to correspond to. This enables real-time farm monitoring, precise irrigation control, and fine-grained crop management.

The organization of the article is given as follows. Section 2 summarizes the related work on offloading optimization in edge computing. Section 3 describes the models used in this paper, and then our time-average energy-efficient offloading problem is rigorously defined. In Section 4, we introduce the details of applying Lyapunov optimization to solve this problem and propose an algorithm QC-EEDOA. In Section 5, the performance of QC-EEDOA is analyzed theoretically. In Section 6, several groups of experiments are performed to evaluate the performance of QC-EEDOA, and the performance comparison between four strategies is presented. Section 7 concludes the article finally.

2. Related work

The energy-aware offloading optimization problem in MEC has been widely studied from different aspects [9,10].

Bi et al. [6] investigated the full offloading on a multi-user MEC system powered by wireless power transfer (WPT), and aimed at maximizing the weighted computation rate. Dinh et al. [11] focused on the offloading of a multi-server MEC system with one DVFS-enabled device to optimize the task execution latency and energy consumption of the device. Tao et al. [12] studied the energy-efficient and performance-guaranteed partial offloading problem for a multi-user single-MEC system. Liu et al. [13] also focused on partial computation offloading for a multi-user multi-MEC environment. It aims at maximizing the deadline-satisfying ratio. An offloading policy is proposed based on game theory to find the Pareto optimal solution. Li [14] defined two offloading problems with energy and time constraints, respectively. Combinatorial optimization is utilized to solve the problems, and a two-stage method is proposed, which jointly generates the offloading decision and decides the computation speeds of UEs and the communication speeds. Tong et al. [15] introduced the Stackelberg game to describe the relationship between MEC servers and end-users (EUs) and proposed uniform pricing and differentiated pricing algorithms to solve the task offloading problem for EUs. However, this article does not consider the energy consumption of followers in the Stackelberg game.

Above works can achieve good performance in terms of energy consumption or execution efficiency. However, most of the above studies assume that task arrival rate and channel state are known prior or can be predicted based on historical statistics. In actuality, the task arrival

rate is dynamic and the wireless channel quality is affected by many factors. Hence they are hard to be predicted precisely.

To address the above shortages, different stochastic optimization strategies were applied in recent studies, which can make online decisions without requiring any prior information. Among these techniques, Lyapunov optimization is one of the most common ones [16–25]. Islam et al. [26] provided a comprehensive review of MEC task offloading schemes proposed by various researchers. It discusses the issues, challenges, and future research directions in the field of task offloading to MEC servers. Chen et al. [17] studied the task offloading problem in a multi-user MEC system. This paper considered the dynamical task arrival rate and the time-varying channel condition and proposed a stochastic energy-efficient optimization algorithm such that the long-term energy consumption is minimized while the queue state keeps stable. This paper offloads all tasks in the queue to MEC and does not take advantage of local computing capacity, which is unreasonable. Lin et al. [18] utilized the Lyapunov optimization method to design a real-time offloading algorithm for addressing the business congestion arising from the spatiotemporal dynamics of randomly generated mobile user demand tasks, but it does not take into account the energy consumption of the system. Sun et al. [20] investigated the offloading problem for a single user in an MEC-enabled ultra-dense network where multiple BSs are densely deployed. A dynamic mobility management scheme is proposed to reduce the average delay of tasks and guarantee to satisfy the energy consumption constraint of devices in the long run. Chen et al. [21] studied the computation peer offloading among small-cell base stations (SBSs) in load-unbalanced small-cell networks. Because of the system dynamics, an online peer offloading strategy is proposed. By applying the strategy, the long-term system performance can be improved efficiently on the premise of ensuring the individual long-term energy consumption constraint. Liang et al. [22] investigated service function chaining in edge core networks and proposes a profit-driven heuristic search algorithm to optimize the average latency of all service function chains in the edge core network. Wu et al. [23] proposed a latency-aware energy-saving online offloading algorithm that adaptively offloads more tasks when the network quality is good while ensuring that tasks do not violate deadlines during periods of poor network quality. Similar to [17], this article also does not take advantage of local computing capacity. Zhen et al. [25] considered the system dynamics, including computing resources, the radio environment, and the battery power, and proposed a dynamic optimization scheme based on Lyapunov optimization. Many works also studied dynamic offloading for MEC systems where devices are energy harvesting [27–29]. However, most dynamic task offloading algorithms only aim to maintain system stability in terms of queue length. Hence, the task queue length of the system, as well as the response time of tasks, are uncontrollable. Our work is motivated by this shortage.

3. The models

In this section, we first introduce the related models. Based on the models, the energy-efficient computation offloading problem can be formulated and studied rigorously.

3.1. System model

The MEC system with one base station (BS) is considered in this paper, which is depicted in Fig. 1. An MEC server deployed at the BS provides services to nearby IoT devices. Let $\mathcal{N} = \{1, 2, \dots, n\}$ be the index set of the n IoT devices within the coverage of the BS. Due to the dynamic nature of the system, we divide time into a series of time slots $\mathcal{T} = \{0, 1, \dots, T-1, \dots\}$. The length of each slot is identical and denoted as τ. It is assumed that the system status keeps static during such a small time slot. Table 1 lists the key notations used in this paper.

Each IoT device i generates new tasks during each time slot. Let $A_i(t)$ (in bits) denote the number of tasks that arrive at device i in slot t. Notice that the bits of the tasks generated by the devices are bit-wise independent so that partial offloading schemes can be applied on each device [16,17]. After tasks arrive, they are firstly stored in a task buffer queue and wait to be processed. The tasks in the queue are processed in the first-come-first-served (FCFS) discipline. Let $q_i(t)$ be the initial queue length of device i in slot t, $D_i^l(t)$ and $D_i^o(t)$ be the number of tasks executed locally and offloaded to MEC in slot t respectively, and $D_i(t)$ be the total amount of tasks processed in slot t. We have $D_i(t) = D_i^l(t) + D_i^o(t)$. It is obvious that each device cannot process more tasks than what it has, so

$$D_i(t) = D_i^l(t) + D_i^o(t) \le q_i(t) + A_i(t), \forall i \in \mathcal{N} \tag{1}$$

holds always.

Table 1
Definitions of mathematical notations.

Notation	Definition
$A_i(t)$	The amount of tasks which arrive at device i in slot t
B	The bandwidth of each sub-channel
$D_i^l(t)$	The amount of tasks executed locally of device i in slot t
$D_i^o(t)$	The amount of tasks offloaded to MEC of device i in slot t
$e_i^l(t)$	The energy consumption for local computation of device i in time slot t
$e_i^o(t)$	The transmit energy consumption of device i in slot t
f_i	The computation speed (in Hz) of device i
$h_i(t)$	The channel power gain of device i in slot t
N_0	The noise power spectral density
P_i	The computing power consumption (in Watts) of device i in time slot t
P_{tra}^i	The transmit power (in Watts) of device i
q_{cons}^i	The queue length threshold for each device i
$q_i(t)$	The initial queue length of device i at slot t
$r_i(t)$	The communication speed (in bits/s) from device i to MEC in time slot t
$S(t)$	The number of sub-channels which are available during time slot t
$\kappa_i(t)$	The duration of a sub-channel allocated to device i for offloading in time slot t

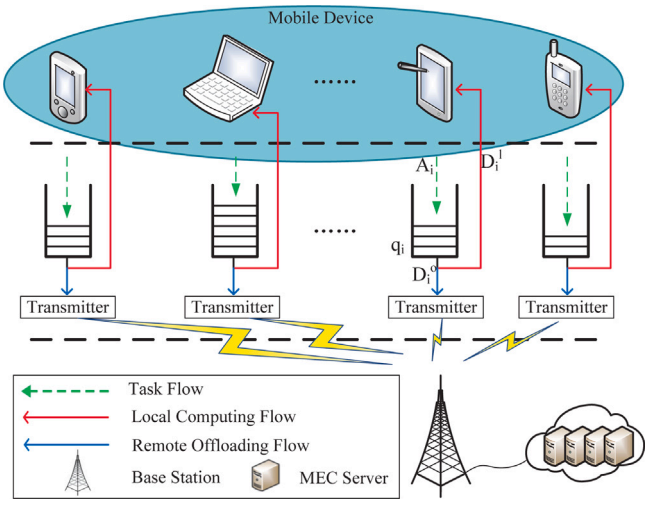


Fig. 1. Task offloading model.

3.2. Computation and energy models

The delay and energy models are introduced firstly for local computation and task transmission [17].

Local Computation. For generality, the IoT devices considered in this paper are heterogeneous. Let f_i be the computation speed (in Hz) of device i, and C_i (in cycles/bit) be a known constant which describes the number of CPU cycles needed to process one bit of task for device i. Then, the time required to process $D_i^l(t)$ bits tasks in device i is $T_i(t) = D_i^l(t)C_i/f_i$. Since the tasks should be processed within current time slot, that is, $T_i(t) \le \tau$, we have

$$0 \le D_i^l(t) \le \frac{\tau f_i}{C_i}, \forall i \in \mathcal{N} \tag{2}$$

that means, the tasks allocated to device i in slot t cannot exceed its computing capacity.

Let P_i be the computing power consumption (in Watts) of device i. Generally, P_i depends on the chip architecture of the device, and it is proportional to CPU frequency, which is $P_i = \xi f_i^3$ where ξ is a coefficient depending on chip architecture [30–32]. Then, the energy consumption of device i for local computation is

$$e_i^l(t) = P_i T_i(t) = \xi f_i^2 C_i D_i^l(t).$$

Task Transmission. Let P_{tra}^i be the transmit power (in Watts) of device i. The communication speed (in bits/s) from device i to MEC is

$$r_i(t) = B\log_2\left(1 + \frac{P_{tra}^i h_i(t)}{BN_0}\right),$$

where B denotes the bandwidth of each sub-channel, $h_i(t)$ is the channel power gain in slot t [33], and N_0 is the noise power spectral density [14,34,35].

For generality, we assume that the channel resources are dynamically changing over different time slots in this paper. Let $S(t)$ be the number of sub-channels which are available during time slot t. To achieve optimal performance, the channel resources should be allocated to different devices properly. Define the channel allocation decisions as $\boldsymbol{\kappa}(t) = \{\kappa_1(t), \dots, \kappa_n(t)\}$, where $\kappa_i(t)$ represents the duration of a sub-channel allocated to device i for offloading. Then, the task amount that device i can offload during slot t is

$$D_i^o(t) = r_i(t)\kappa_i(t).$$

Because a device cannot offload more tasks than what it has, $\kappa_i(t)$ should satisfy

$$\kappa_i(t) \le \frac{q_i(t)+A_i(t)}{r_i(t)}, \forall i \in \mathcal{N}. \tag{3}$$

Consider each IoT device operates in narrow-band [1], hence each device can only access a sub-channel at a time, so we have

$$0 \le \kappa_i(t) \le \tau, \forall i \in \mathcal{N}. \tag{4}$$

Combining (3) with (4), we obtain

$$0 \le \kappa_i(t) \le \min\left\{\frac{q_i(t)+A_i(t)}{r_i(t)}, \tau\right\}, \forall i \in \mathcal{N}. \tag{5}$$

Similar to [17,36], we considered that different devices can access a sub-channel at different times during one slot. Hence, the total duration of channel resources allocated to all devices cannot exceed the total communication length of all sub-channels, that is,

$$\sum_{i=1}^{n} \kappa_i(t) \le S(t)\tau. \tag{6}$$

For device i, the transmit energy consumption in slot t is

$$e_i^o(t) = P_{tra}^i(t)\kappa_i(t).$$

Based on the above calculation, the total energy consumed by all devices is

$$e(t) = \sum_{i=1}^{n}\left(\xi f_i^2 C_i D_i^l(t) + P_{tra}^i \kappa_i(t)\right). \tag{7}$$

In this paper, the computing capacity of the MEC server is assumed to be enough, and we do not consider the allocation of the computing resources of the MEC server for two reasons. First, the communication resources are the main bottleneck limiting the benefits of task offloading compared with the MEC computation resources. Second, this work is to minimize the energy consumption of devices, and the energy consumption is not affected by the computing capacity of MEC and the MEC computing resource allocation schemes. Hence, the above assumption is reasonable, which is also adopted in many existing works such as [17,37,38].

3.3. Queue state transition model

Recall that $q_i(t)$ is the initial queue length of device i at slot t. According to our problem, the queue length in the next slot is

$$q_i(t+1) = \max\{q_i(t) + A_i(t) - D_i(t), 0\}, \tag{8}$$

where $q_i(0) = 0$ for each device i.

In this paper, we try to maintain the queue stability of all IoT devices and stabilize each device's queue backlog below a given threshold in the long term. Hence, the average queue backlog across multiple time slots should be bounded for each device i, that is,

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\{q_i(t)\} \le q_{cons}^i, \forall i \in \mathcal{N}, \tag{9}$$

where q_{cons}^i is the queue length threshold for each device i.

The reasons for introducing such a queue backlog threshold for each device are explained as follows. Under the queue stability constraint, the average queue backlog is uncontrollable. When the system reaches stability, the queue backlog might be huge. According to Little's law [39], the average queuing delay is proportional to queue backlogs. Under a large queue backlog, the performance in terms of task response time will deteriorate significantly. By introducing a queue backlog threshold, we can not only minimize the task response time but also control the system performance effectively by adjusting the threshold values.

3.4. Energy optimization problem

Next, we give a formal definition of our energy optimization problem with resource constraint and queue length constraint.

According to the model description, it is obvious that the quality of wireless channels is dynamically changing, and the amount of arriving tasks are dynamic over time. Consequently, the total energy consumed by all devices fluctuates too. Therefore, the energy consumption should be averaged over a long time scale, which can be expressed as

$$\bar{e} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\{e(t)\}.$$

From Eq. (7), we know that the energy consumption in each slot t is varying and depends on the task offloading decision $(D_i^l(t), D_i^o(t))$, where $D_i^o(t)$ depends on the channel allocation decision $\kappa_i(t)$ further. To achieve minimal energy consumption, the proper decisions in terms of task offloading and resource allocation should be determined for each time slot, and our energy optimization problem is formulated as

$$\textbf{(P1)} \quad \min_{\boldsymbol{D}^l(t),\boldsymbol{\kappa}(t)} \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\{e(t)\},$$
$$\text{s.t.} \quad (1), (2), (5), (6), (9),$$

where $\boldsymbol{D}^l(t) = \{D_1^l(t), \dots, D_n^l(t)\}$.

Analyzing problem (**P1**), it is a stochastic optimization problem. Solving this problem requires the future system information, including $A_i(t)$, $h_i(t)$, and $S(t)$, etc. However, the statistical information is generally hard to predict accurately in real systems, so solving this problem offline is of great challenge. This paper proposes a Lyapunov optimized online task offloading strategy to address the problem. The algorithm makes decisions only depending on current information of devices and channel conditions but has no requirements on the statistical information for a long time.

4. Lyapunov optimized energy-efficient dynamic offloading

In this section, the original stochastic optimization problem is first transformed into a real-time decision problem by applying Lyapunov optimization. After that, an energy-efficient dynamic offloading algorithm with queue length constraint (QC-EEDOA) is designed.

4.1. Problem transformation

To transform our problem, a virtual queue Q is constructed firstly for the queue length constraint [40], which is shown as

$$Q_i(t+1) = \max\{Q_i(t) + q_i(t+1) - q^i_{cons}, 0\}, \tag{10}$$

where $Q_i(t)$ is the virtual queue backlog in slot t. Noticed that Q is different from q, since q is the actual queue of devices, while Q is a virtual queue. The virtual queue backlog $Q_i(t)$ reflects the degree that q_i exceeds the threshold q^i_{cons} by the end of time slot t, and $Q_i(0) = 0$ for all devices. When the queue length of q exceeds the threshold q_{cons} at t, the queue backlog of Q increases, and vice versa. The specific transformation process is depicted in Fig. 2. Hence, the long-term queue length constraint (9) is transformed into a stability constraint of the queue backlog of Q. Until that, the Lyapunov optimization theory can be utilized to solve this problem.

Denoting $\Theta(t) \triangleq \{Q_1(t), \ldots, Q_n(t)\}$, we define the Lyapunov function as

$$L(\Theta(t)) \triangleq \frac{1}{2}\sum_{i=1}^{n} Q_i(t)^2,$$

and the Lyapunov drift is defined as

$$\Delta(\Theta(t)) \triangleq E[L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)].$$

After introducing the virtual queue Q, our goal becomes to find an energy-efficient task offloading policy while maintaining the virtual queue stability. To achieve the dual goals, we incorporate virtual queue stability into energy consumption with a weight of V, and define a *Lyapunov drift-plus-penalty* function as

$$\Delta(\Theta(t)) + Ve(t),$$

where V is a positive constant which is to trade off energy consumption against virtual queue backlog.

Now, the original optimization problem (**P1**) can be transformed as

$$\textbf{(P2)} \quad \min_{\boldsymbol{D}^I(t),\boldsymbol{\kappa}(t)} \ \Delta(\Theta(t)) + Ve(t),$$
$$\text{s.t.} \quad (1), (2), (5), (6).$$

Analyzing problem (**P2**), we find that the task offloading decision in slot t still depends on the information of next time slot. To address this problem, we calculate the upper bound of $\Delta(\Theta(t)) + Ve(t)$ as Theorem 1.

Theorem 1. *If $A_i(t)$ is upper bounded by A_i^{max} over time, the drift-plus-penalty value under any task offloading algorithm satisfies*

$$\Delta(\Theta(t)) + Ve(t) \le B_1 + B_2 + Ve(t) + \sum_{i=1}^{n}\left(\frac{D_i(t)^2}{2} + \left(q^i_{cons} - q_i(t) - A_i(t) - Q_i(t)\right)D_i(t)\right), \tag{11}$$

where

$$B_1 = \frac{1}{2}\sum_{i=1}^{n}\left((A_i^{max})^2 + (q^i_{cons})^2\right),$$

and

$$B_2 = \sum_{i=1}^{n}\left(\frac{1}{2}q_i(t)^2 + A_i^{max}q_i(t) + Q_i(t)\left(q_i(t) + A_i^{max} - q^i_{cons}\right)\right).$$

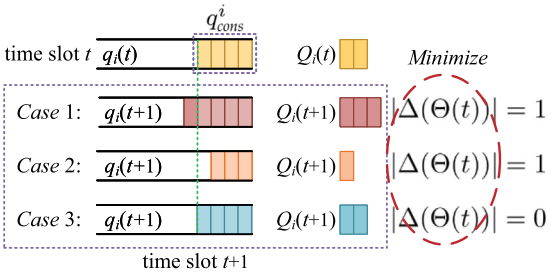


Fig. 2. Transformation process of virtual queue Q.

Proof. Taking the square of (10) and exploiting $(\max\{a, 0\})^2 \le a^2$, we have

$$\begin{aligned} &Q_i(t+1)^2 - Q_i(t)^2 \\ &\le (Q_i(t) + q_i(t+1) - q^i_{cons})^2 - Q_i(t)^2 \\ &= (q_i(t+1) - q^i_{cons})^2 + 2Q_i(t)(q_i(t+1) - q^i_{cons}) \\ &= q_i(t+1)^2 - 2q_i(t+1)q^i_{cons} + 2Q_i(t)q_i(t+1) \\ &\quad + (q^i_{cons})^2 - 2Q_i(t)q^i_{cons}. \end{aligned} \tag{12}$$

Taking square on (8), we have

$$\begin{aligned} &q_i(t+1)^2 \le (q_i(t) - D_i(t) + A_i(t))^2 \\ &= (q_i(t) - D_i(t))^2 + A_i(t)^2 + 2A_i(t)(q_i(t) - D_i(t)) \\ &= q_i(t)^2 - 2q_i(t)D_i(t) + D_i(t)^2 + A_i(t)^2 \\ &\quad + 2A_i(t)q_i(t) - 2A_i(t)D_i(t). \end{aligned} \tag{13}$$

In addition, because $D_i(t) \le q_i(t) + A_i(t)$, we can also obtain

$$-2q_i(t+1)q^i_{cons} = 2D_i(t)q^i_{cons} - 2q^i_{cons}(q_i(t) + A_i(t)), \tag{14}$$

and

$$2Q_i(t)q_i(t+1) = 2Q_i(t)(q_i(t) + A_i(t)) - 2Q_i(t)D_i(t). \tag{15}$$

Substituting (13), (14), (15) to (12), we have

$$\begin{aligned} &Q_i(t+1)^2 - Q_i(t)^2 \\ &\le D_i(t)^2 + 2\left(q^i_{cons} - q_i(t) - A_i(t) - Q_i(t)\right)D_i(t) \\ &\quad + q_i(t)^2 + A_i(t)^2 + 2A_i(t)q_i(t) - 2q^i_{cons}\left(q_i(t) + A_i(t)\right) \\ &\quad + 2Q_i(t)\left(q_i(t) + A_i(t) - q^i_{cons}\right) + (q^i_{cons})^2 \\ &\le D_i(t)^2 + 2\left(q^i_{cons} - q_i(t) - A_i(t) - Q_i(t)\right)D_i(t) \\ &\quad + q_i(t)^2 + (A_i^{max})^2 + 2A_i^{max}q_i(t) \\ &\quad + 2Q_i(t)(q_i(t) + A_i^{max} - q^i_{cons}) + (q^i_{cons})^2. \end{aligned}$$

Let

$$B_1 = \frac{1}{2}\sum_{i=1}^{n}\left((A_i^{max})^2 + (q^i_{cons})^2\right),$$

and

$$B_2 = \sum_{i=1}^{n}\left(\frac{1}{2}q_i(t)^2 + A_i^{max}q_i(t) + Q_i(t)\left(q_i(t) + A_i^{max} - q^i_{cons}\right)\right).$$

The objective function (11) can be upper bounded as

$$\Delta(\Theta(t)) + Ve(t) \le B_1 + B_2 + Ve(t) + \sum_{i=1}^{n}\left(\frac{D_i(t)^2}{2} + \left(q^i_{cons} - q_i(t) - A_i(t) - Q_i(t)\right)D_i(t)\right).$$

The lemma is proven. □

Till now, we can transform the original problem (problem (**P2**)) and turn to minimize its upper bound. By doing so, the average energy consumption can be reduced effectively while the queue length constraint of each IoT device can be guaranteed. Since B_1 is a constant during all time slots and B_2 is fixed at a specific time slot, B_1 and B_2 can be

discarded from the objective function and the problem is constructed as

$$\textbf{(P3)} \min_{\boldsymbol{D}(t),\boldsymbol{D}^l(t),\boldsymbol{\kappa}(t)} V\sum_{i=1}^{n}\left(\xi f_i^2 C_i D_i^l(t) + P_{tra}^i(t)\kappa_i(t)\right) + \sum_{i=1}^{n}\left(\frac{D_i(t)^2}{2} + \left(q_{cons}^i - q_i(t) - A_i(t) - Q_i(t)\right)D_i(t)\right),$$
$$\text{s.t.} \quad (1), (2), (5), (6).$$

Since $D_i(t) = D_i^l(t) + D_i^o(t) = D_i^l(t) + r_i(t)\kappa_i(t)$, we have $D_i^l(t) = D_i(t) - r_i(t)\kappa_i(t)$. Substituting $D_i^l(t)$ into (**P3**), we have

$$\textbf{(P4)} \min_{\boldsymbol{D}(t),\boldsymbol{\kappa}(t)} \sum_{i=1}^{n}\psi_i(t)\kappa_i(t) + \sum_{i=1}^{n}\left(\frac{D_i(t)^2}{2} + \omega_i(t)D_i(t)\right),$$
$$\text{s.t.} \quad (1), (2), (5), (6),$$

where $\psi_i(t) = V P_{tra}^i(t) - V\xi f_i^2 C_i r_i(t)$ and $\omega_i(t) = q_{cons}^i - q_i(t) - A_i(t) - Q_i(t) + V\xi f_i^2 C_i$.

4.2. Energy-efficient dynamic offloading algorithm

Analyzing problem (**P4**), $D_i(t)$ and $k_i(t)$ are the optimization variables in the optimization problem. Since (**P4**) exists the dynamic coupling between The number of tasks and the offloading duration of IoT device, finding the optimal values of decision variables is difficult. Thus, an energy-efficient dynamic offloading algorithm with queue length constraint (QC-EEDOA) is proposed.

This algorithm can minimize the *drift plus penalty's* upper bound. Through the observation of (**P4.2**), we find that the range of $D_i(t)$ depends on the value of $k_i(t)$. Accordingly, the objective optimization problem can be decomposed into two sub-problems, allowing each to be addressed independently. Next, we notice that the portion of (**P4**) associated with the variable $k_i(t)$. Thus, the problem (**P4**) is transformed into (**P4.1**), and we apply the knapsack theory to find a sub-optimal solution of $k_i(t)$ in (**P4.1**). After solving the problem (**P4.1**), the problem (**P4.2**) is equal to (**P4.2**).

$$\textbf{(P4.1)} \quad \min_{\boldsymbol{\kappa}(t)} \sum_{i=1}^{n}\psi_i(t)\kappa_i(t),$$
$$\text{s.t.} \quad (5), (6),$$

where $\psi_i(t) = V P_{tra}^i(t) - V\xi f_i^2 C_i r_i(t)$.

$$\textbf{(P4.2)} \quad \min_{\boldsymbol{D}(t)} \sum_{i=1}^{n}\left(\frac{D_i(t)^2}{2} + \omega_i(t)D_i(t)\right),$$
$$\text{s.t.} \quad r_i(t)\kappa_i(t) \le D_i(t) \le \min\left\{r_i(t)\kappa_i(t) + \frac{\tau f_i}{C_i}, q_i(t) + A_i(t)\right\}, \quad \forall i \in \mathcal{N}, \tag{17a}$$

where $\omega_i(t) = q_{cons}^i - q_i(t) - A_i(t) - Q_i(t) + V\xi f_i^2 C_i$.

The steps of determining the offloading decision per time slot are given as follows:

(1) Solving problem (**P4.1**) to find a solution for $\boldsymbol{\kappa}(t)$;
(2) Updating the value range of $D_i(t)$ (see (17a)), and solving problem (**P4.2**) to find a solution for $\boldsymbol{D}(t)$;
(3) Adjusting the value of $\boldsymbol{\kappa}(t)$ and repeating step 2 until the solution cannot be optimized.

Solution of Problem (**P4.1**): Similar to [17], problem (**P4.1**) can be considered as a divisible knapsack problem in which $S(t)\tau$ is the knapsack capacity, $\min\{(q_i(t) + A_i(t))/r_i(t), \tau\}$ is the size of each item, and $\psi_i(t)$ can be considered as the unit value of each item. To obtain the optimal solution of the divisible knapsack problem, the principle is to fulfill the knapsack with the smallest negative $\psi_i(t)$ in priority. The detailed process is described in Alg. 1. In the algorithm, all devices are sorted in the ascending order of $\psi_i(t)$ firstly. And then, the device with the smallest negative $\psi_i(t)$ is selected and allocated with the most available offloading duration. The process is repeated until the unit value of the selected device is positive or the knapsack is empty.

Algorithm 1: Determining $\boldsymbol{\kappa}(t)$

Input: device parameters such as $P_{tra}^i(t)$, ξ_i, f_i, C_i; channel parameters such as B, $h_i(t)$, N_0, $S(t)$; queue state such as $q_i(t)$;
Output: the solution of $\boldsymbol{\kappa}(t)$;

```
for all i ∈ 𝒩 do
    Calculate r_i(t) and ψ_i(t);
    κ_i(t) ← 0;
end
Sort all devices in the ascending order of ψ_i(t);
restK ← S(t)τ;
while restK > 0 do
    i ← pop the index of the first device with the smallest ψ_i(t);
    if ψ_i(t) < 0 then
        κ_i(t) ← min{restK, min{(q_i(t)+A_i(t))/r_i(t), τ}};
        restK ← restK − k_i(t);
    else
        break;
    end
end
```

Solution of Problem (**P4.2**): After determining the value of $\boldsymbol{\kappa}(t)$ in the first step, the value ranges of $D_i(t)$ can be fixed. Since the values of $D_i(t)$ are non interacting, which are only related to $\kappa_i(t)$, it is easy to determine $D_i(t)$ for each device i as follows. Let

$$\mathcal{R}_i(D_i(t)) = D_i(t)^2/2 + \omega_i(t)D_i(t), \tag{18}$$

it is a quadratic equation of $D_i(t)$ obviously. Taking the derivative with respect to $D_i(t)$, we have

$$\frac{\partial \mathcal{R}_i(D_i(t))}{\partial D_i(t)} = D_i(t) + \omega_i(t),$$

so the theoretical optimum is $D_i(t) = -\omega_i(t)$. Since the value range of $D_i(t)$ is limited as (17a), the actual optimum of $D_i(t)$ is discussed as the Alg. 2.

Algorithm 2: Determining $\boldsymbol{D}(t)$

Input: $\boldsymbol{\kappa}(t)$ determined by Alg. 1;
Output: the solution of $\boldsymbol{D}(t)$;

```
for all i ∈ 𝒩 do
    Calculate r_i(t) and ω_i(t);
    if −ω_i(t) < κ_i(t)r_i(t) then
        D_i(t) ← κ_i(t)r_i(t);
    end
    if κ_i(t)r_i(t) ≤ −ω_i(t) ≤ min{r_i(t)κ_i(t)+τf_i/C_i, q_i(t)+A_i(t)} then
        D_i(t) ← −ω_i(t);
    end
    if −ω_i(t) > min{r_i(t)κ_i(t)+τf_i/C_i, q_i(t)+A_i(t)} then
        D_i(t) ← min{r_i(t)κ_i(t)+τf_i/C_i, q_i(t)+A_i(t)};
    end
end
```

Adjustment of $\boldsymbol{\kappa}(t)$ and $\boldsymbol{D}(t)$: In Alg. 2, if $-\omega_i(t) < \kappa_i(t)r_i(t)$, $D_i(t)$ is set as $\kappa_i(t)r_i(t)$. In that case, the solution of $\kappa_i(t)$ and $D_i(t)$ can be optimized further.

Substituting $D_i(t) = \kappa_i(t)r_i(t)$ into $\mathcal{R}_i(D_i(t))$, we have

$$\mathcal{R}_i(\kappa_i(t)) = \frac{1}{2}r_i^2(t)\kappa_i^2(t) + \omega_i(t)r_i(t)\kappa_i(t).$$

Let

$$\mathcal{S}_i(\kappa_i(t)) = \psi_i(t)\kappa_i(t),$$

then both $\mathcal{R}_i$ and $\mathcal{S}_i$ are functions of $\kappa_i(t)$. According to Alg. 1, if $\kappa_i(t) > 0$, it has $\psi_i(t) < 0$. Taking the derivative of $\mathcal{R}_i$ and $\mathcal{S}_i$ with

respect to $\kappa_i(t)$, we have

$$\frac{\partial \mathcal{R}_i(\kappa_i(t))}{\partial \kappa_i(t)} = r_i^2(t)\kappa_i(t) + \omega_i(t)r_i(t) > 0,$$

and

$$\frac{\partial \mathcal{S}_i(\kappa_i(t))}{\partial \kappa_i(t)} = \psi_i(t) < 0.$$

Hence, $\mathcal{R}_i$ and S_i are monotonically increasing and decreasing with $\kappa_i(t)$, respectively.

If $r_i^2(t)\kappa_i(t)+\omega_i(t)r_i(t) > |\psi_i(t)|$, then the value of $\mathcal{R}_i(\kappa_i(t)) + \mathcal{S}_i(\kappa_i(t))$ decreases with the decrease of $\kappa_i(t)$ until $r_i^2(t)\kappa_i(t)+\omega_i(t)r_i(t) = |\psi_i(t)|$. Hence, we can conclude that the optimum of $\kappa_i(t)$ can be determined by

$$\kappa_i(t) = \min\left\{\frac{-\psi_i(t) - \omega_i(t)r_i(t)}{r_i^2(t)}, 0\right\}. \tag{19}$$

The details are given as Alg. (3).

Algorithm 3: Adjusting $\boldsymbol{\kappa}(t)$ and $\boldsymbol{D}(t)$

```
1  iteratively Input: κ(t) and D(t) determined by Alg. 1 and Alg. 2;
   Output: κ(t) and D(t) after adjustment;
2  do
3      isChanged ← false;
4      restK ← S(t)τ − Σκ(t);
5      for all i ∈ 𝒩 do
6          if r_i^2(t)κ_i(t)+ω_i(t)r_i(t) > |ψ_i(t)| then
7              Calculate κ_i(t) by Eq. (19);
8              Update D_i(t) based on κ_i(t) according to Alg. 2;
9              isChanged ← true;
10         end
11     end
12     restK' ← S(t)τ − Σκ(t);
13     if restK = 0 and restK' > 0 then
14         Allocate restK' to the devices having no channel resources
             according to Alg. 1;
15         Calculate the correspond κ_i(t) and D_i(t);
16         isChanged ← true;
17     end
18 while isChanged = true;
```

4.3. Algorithm complexity

Since we focus on the computation offloading problem of the dynamic system where many parameters change over time, our algorithm should make snap decisions on the offloading decision and resource allocation scheme per time slot. The time complexity of our algorithm is expressed in terms of the number of devices n. For Alg. 1, the complexity of parameter calculation is $O(n)$ (lines 1–4), the complexity of sorting is $O(n\log_2 n)$ (line 5), and the complexity of allocating channel resources is $O(n)$ (lines 7–14). Therefore, the time complexity of Alg. 1 is $O(n\log_2 n)$. In Alg. 2, there are only comparison and assignment operations, so the time complexity of Alg. 2 is $O(n)$. In Alg. 3, the values of κ_i and D_i are updated at most once for each device. Hence, its time complexity is $O(n)$. In total, the complexity of the proposed solution is $O(n\log_2 n)$. Besides, the space complexity of our proposed algorithm is relatively low. When the program runs on the computer, the memory changes in the system are very small. This is because the algorithm uses metadata to compute the optimal offloading policy, which is simple and efficient, rather than performing tasks. When running the algorithm, the CPU usage increased from 7% to 41%. The change is also acceptable.

5. Performance analysis

In this section, the theoretical analysis are conducted to verify the performance of the proposed offloading policy.

5.1. Queue length constraint

We first show that the solution derived by problem (**P4**) satisfies the long-term queue length constraint defined in (9). To do so, we first derive the following theorem.

Theorem 2. *Adopting the proposed algorithm, the virtual queue backlog Q_i is upper bounded for any device $i \in \mathcal{N}$ when the average system load in long term is within the system computing capacity.*

Proof. Taking the derivative of the objective function in problem (**P4**), the theoretical optimal value is achieved at $D_i(t) = -\omega_i(t)$. According to Alg. 2, the actual optimal value is discussed in three cases as follows.

Case 1. $D_i^*(t) = \kappa_i^*(t)r_i(t)$, that means $-\omega_i(t) < \kappa_i^*(t)r_i(t)$. In such a case, we have

$$q_i(t)+A_i(t)+Q_i(t) - q_{cons}^i - VP_iC_i/f_i < \kappa_i^*(t)r_i(t).$$

Hence, $Q_i(t)$ is bounded as

$$Q_i(t) < q_{cons}^i + VP_iC_i/f_i + \kappa_i^*(t)r_i(t). \tag{20}$$

Case 2. When $\kappa_i^*(t)r_i(t) \le -\omega_i(t) \le \min\{r_i(t)\kappa_i^*(t)+\tau f_i/C_i, q_i(t)+A_i(t)\}$, $D_i^*(t) = -\omega_i(t)$. In such a case, we have

$$\begin{aligned} Q_i(t) = {} & D_i^*(t) + q_{cons}^i + V\xi_i f_i^2 C_i - q_i(t) - A_i(t) \\ < {} & q_{cons}^i + VP_iC_i/f_i, \end{aligned} \tag{21}$$

since $D_i^*(t) \le q_i(t) + A_i(t)$.

Case 3. When $-\omega_i(t) > \min\{r_i(t)\kappa_i^*(t)+\tau f_i/C_i, q_i(t)+A_i(t)\}$, $D_i^*(t) = \min\{r_i(t)\kappa_i^*(t)+\tau f_i/C_i, q_i(t)+A_i(t)\}$. In such a case,

$$\begin{aligned} Q_i(t) > {} & q_{cons}^i + VP_iC_i/f_i - q_i(t) - A_i(t) \\ & + \min\left\{r_i(t)\kappa_i^*(t) + \frac{\tau f_i}{C_i}, q_i(t)+A_i(t)\right\}. \end{aligned}$$

According to Eq. (8), we have

$$q_i(t+1) = q_i(t) + A_i(t) - \min\left\{r_i(t)\kappa_i^*(t) + \frac{\tau f_i}{C_i}, q_i(t)+A_i(t)\right\},$$

which means that the system processes as many tasks as possible for device i at slot t. When the computing capacity of device i is great enough, $q_i(t+1)$ becomes smaller than q_{cons}^i, and according to Eq. (10), we have $Q_i(t+1) < Q_i(t)$ and the value of $-w_i(t)$ shows a downtrend. After limited time slots, there is a time slot t_0 such that $-w_i(t_0)$ satisfies the condition in case 1 or 2, then $Q_i(t)$ satisfies (20) and (21) again when $t \ge t_0$.

In all, $Q_i(t)$ is upper bounded under the given condition. This completes the proof. □

Based on Theorem 2, we further deduce the following theorem.

Theorem 3. *The constraint*

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} q_i(t) \le q_{cons}^i$$

is satisfied if the virtual queue backlog Q_i is upper bounded.

Proof. Suppose that the virtual queue backlog Q_i is upper bounded by $Q_i^{max} > 0$ for any device $i \in \mathcal{N}$. Then, we have

$$\lim_{t\to\infty} \frac{Q_i(t)}{t} \le \lim_{t\to\infty} \frac{Q_i^{max}}{t} = 0. \tag{22}$$

Eq. (10) can be rewritten as

$$\begin{aligned} Q_i(t+1) &= \max\{Q_i(t) + q_i(t+1) - q_{cons}^i, 0\} \\ &= \begin{cases} Q_i(t)+q_i(t+1)-q_{cons}^i, & \text{if } Q_i(t) \ge q_{cons}^i - q_i(t+1); \\ 0, & \text{if } Q_i(t) < q_{cons}^i - q_i(t+1). \end{cases} \end{aligned} \tag{23}$$

Reformatting (23), we have

$$\begin{aligned} &Q_i(t+1)-Q_i(t) \\ &=\begin{cases} q_i(t+1)-q_{cons}^i, & \text{if } Q_i(t)\geq q_{cons}^i-q_i(t+1), \\ -Q_i(t), & \text{if } Q_i(t)<q_{cons}^i-q_i(t+1), \end{cases} \\ &=\max\{q_i(t+1)-q_{cons}^i,-Q_i(t)\} \\ &\geq q_i(t+1)-q_{cons}^i. \end{aligned} \tag{24}$$

Taking the average of time slots 0 to $T-1$ on both sides of (24) and bringing T to ∞, we have

$$\lim_{T\to\infty}\frac{Q_i(T)}{T}\geq\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}q_i(t)-q_{cons}^i. \tag{25}$$

Combining (22) and (25), we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E(q_i(t))\leq q_{cons}^i.$$

The lemma is proven. □

Combining Theorems 2 and 3, we can conclude that the queue length constraints can be satisfied for all devices when the average system load in long term is within the system computing capacity.

5.2. Optimality gap analysis

To analyze the performance gap between the optimal solution of the original problem **(P1)** and the solution obtained from our problem **(P4)**, we first give a description on the optimal solution of **(P1)** as follows:

Lemma 4. *If the original problem **(P1)** has solutions, there must be an optimal $\boldsymbol{w}$-only policy $\boldsymbol{\omega}^*$ where the action taken in each slot t satisfies*

$$\begin{cases} e^{\boldsymbol{\omega}^*}(t)=e^*, \\ E\{q_i^{\boldsymbol{\omega}^*}(t)\}\leq q_{cons}^i, \end{cases}$$

*where $e^{\boldsymbol{\omega}^*}(t)$ denotes the energy consumption in slot t under the optimal $\boldsymbol{w}$-only policy, and e^* is the optimal objective value of problem **(P1)**, i.e.,*

$$e^*=\min\left(\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E\{e(t)\}\right).$$

Proof. The conclusion has been concluded in [41]. Hence, we omit the detailed proofs here. □

By applying Lemma 4, we give the upper bound of the time-average energy consumption in Theorem 5.

Theorem 5. *For the given V, the time-average energy consumption of QC-EEDOA satisfies*

$$\overline{e^{QC\text{-}EEDOA}}\leq e^*+\frac{C}{V},$$

where

$$\begin{aligned} C=\ & B_1+\sum_{i=1}^{n}\left(\frac{(q_i^{max})^2}{2}+\frac{(D_i^{max})^2}{2}+q_{cons}^iD_i^{max}\right) \\ &+nA_i^{max}(q_{cons}^i+Q_{cons}^i). \end{aligned}$$

Proof. Let $\boldsymbol{w}'$ be the task offloading strategy solved from problem **(P4)**. We have

$$\begin{aligned} &\Delta(\Theta(t))+Ve(t)\leq B_1+B_2+Ve^{\boldsymbol{w}'}(t) \\ &\quad+\sum_{i=1}^{n}\left(\frac{D_i^{\boldsymbol{w}'}(t)^2}{2}+\left(q_{cons}^i-q_i(t)-A_i(t)-Q_i(t)\right)D_i^{\boldsymbol{w}'}(t)\right), \end{aligned}$$

where $e^{\boldsymbol{w}'}(t)$ represents the energy consumption under strategy $\boldsymbol{w}'$, and $D^{\boldsymbol{w}'}(t)$ is the amount of tasks processed in slot t under $\boldsymbol{w}'$.

According to the problem description, $D_i(t)$ is obviously upper bounded. Let D_i^{max} denote the upper bound of $D_i(t)$, that is $D_i(t)\leq D_i^{max}$. Since $\boldsymbol{w}'$ is the optimal solution of **(P4)**, it can be concluded that

$$\begin{aligned} &\Delta(\Theta(t))+Ve(t)\leq B_1+B_2+Ve^{\boldsymbol{w}^*}(t) \\ &\quad+\sum_{i=1}^{n}\left(\frac{D_i^{\boldsymbol{w}^*}(t)^2}{2}+\left(q_{cons}^i-q_i(t)-A_i(t)-Q_i(t)\right)D_i^{\boldsymbol{w}^*}(t)\right) \\ &\leq B_1+B_2+Ve^*+\frac{1}{2}\sum_{i=1}^{n}(D_i^{max})^2+\sum_{i=1}^{n}q_{cons}^iD_i^{max}. \end{aligned} \tag{26}$$

Because the system dynamics, the queue length fluctuates within a certain range. Let q_i^{max} be the upper bound of $q_i(t)$. In addition, according to Theorems 3 and 2, we know $E\{q_i(t)\}\leq q_{cons}^i$, and $Q_i(t)$ is upper bounded by Q_i^{max}. Taking expectations on B_2, we have

$$E\{B_2\}\leq\frac{1}{2}\sum_{i=1}^{n}(q_i^{max})^2+nA_i^{max}(q_{cons}^i+Q_{cons}^i).$$

Taking expectations on (26), we obtain

$$E\{L(\Theta(t+1))-L(\Theta(t))\}+VE\{e(t)\}\leq C+Ve^*, \tag{27}$$

where

$$C=B_1+\sum_{i=1}^{n}\left(\frac{(q_i^{max})^2}{2}+\frac{(D_i^{max})^2}{2}+q_{cons}^iD_i^{max}\right)+nA_i^{max}(q_{cons}^i+Q_{cons}^i).$$

Summing (27) from slot 0 to $T-1$, it holds

$$E\{L(\Theta(T))-L(\Theta(0))\}+V\sum_{t=0}^{T-1}E\{e(t)\}\leq TC+VTe^*.$$

Since $E\{L(\Theta(T))\}\geq 0$ and $E\{L(\Theta(0))\}=0$, it holds

$$V\sum_{t=0}^{T-1}E\{e(t)\}\leq TC+VTe^*. \tag{28}$$

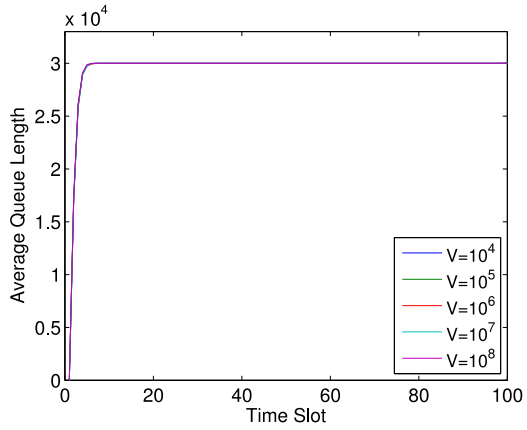
Dividing (28) by VT, and let $T\to\infty$, we obtain

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E\{e(t)\}\leq\frac{C}{V}+e^*.$$

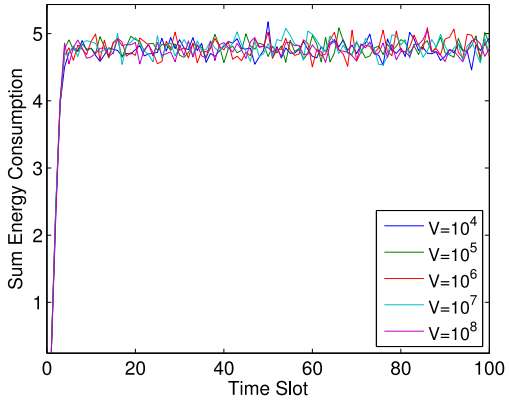
The theorem is proven. □

6. Evaluation

In the experiments, we refer to other articles to set the slot length τ to 1s and consider 100 IoT devices [17,23]. To fully account for the heterogeneity of the device, the frequency of each device i is set as $f_i\sim U[0.5,1]$ GHz, and the queue length thresholds of all devices are set from 20000 bit to 40000 bit in equal step, that is, $q_{cons}^i=20000+20000(i-1)/(n-1)$. ξ is set as 10^{-27} for all devices [16]. The amount of tasks arriving at device i per time slot is set to follow uniform distribution within $[0,40000]$ bits, i.e., $A_i(t)\sim U[0,40000]$ [40]. C_i is set to be distributed uniformly as well, i.e., $C_i\sim U[5000,10000]$. $h_i(t)$ follows an exponential distribution with mean of 1 [17,27]. Notice that an exponential distribution with a mean of 1 can offer a simplified representation of channel gain in MEC scenarios, but does not apply to all MEC scenarios. The number of sub-channels $S(t)$ is selected randomly within $[5,10]$. The transmit power of devices is set as $P_i\sim U[10,200]$ mW. Besides, B is set as 2 MHz and N_0 is set as 10^{-6} W/Hz [17]. For each setting, we run the experiments multiple times and average the results to improve reliability. For each setting, we run the experiments multiple times and average the results to improve reliability. The above experimental parameter settings are only used as a setting to test the performance of our algorithm and cannot cover all MEC scenarios. For different MEC scenarios, the parameter settings can be varied according to the actual situation, but overall it will not affect the execution of the algorithm.

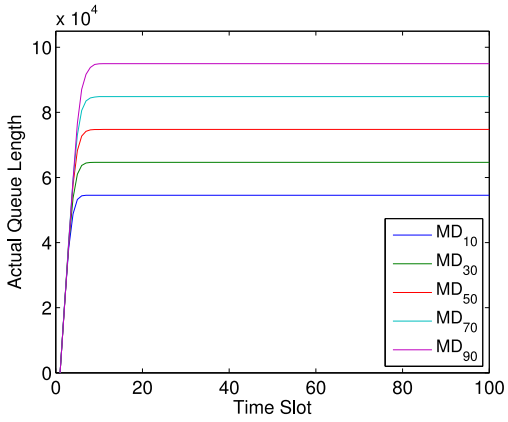


(a) Actual queue length.

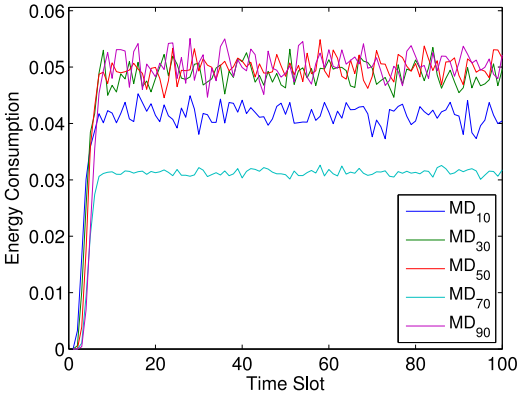


(b) Energy consumption.

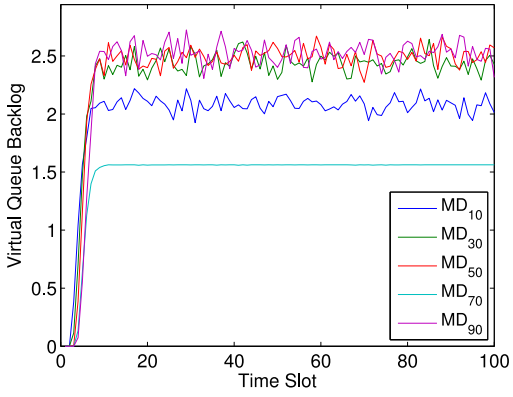
Fig. 3. Queue length and energy consumption under different V.



(a) Actual queue length.



(b) Energy consumption.



(c) Virtual queue backlog.

Fig. 4. Actual queue length, energy consumption and virtual queue backlog of all devices keep stable.

6.1. Selection of trade-off parameter

This group of experiments is to determine a proper parameter V. The relationships between V and the performance indicators are observed. Here, V is set as 10^4, 10^5, 10^6, 10^7, and 10^8, respectively.

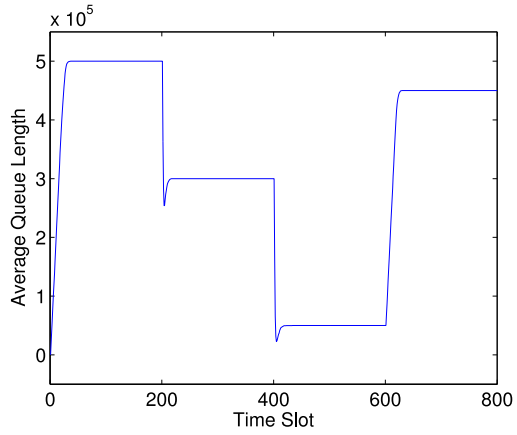
Fig. 3 shows the changing trends of the sum energy consumption and average queue length under different V. The figure shows that both of the two performance indicators can reach stability under different V. Whereas both of them do not show obvious changes as V increases. That is because a queue length constraint is imposed on each device, so the effect of V on the queue length is not obvious. Consequently, the energy consumption is affected slightly by different V as well. In that case, we set V as 10^6 in the following experiments.

6.2. Performance analysis

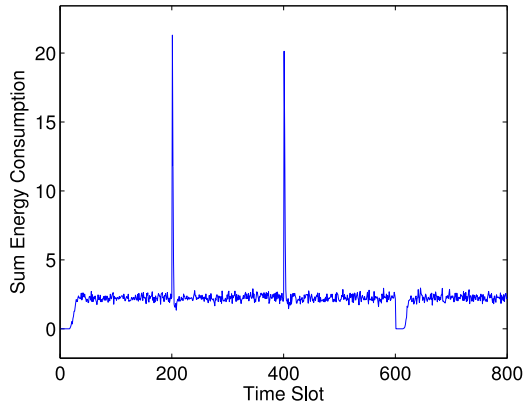
6.2.1. Analysis on the system performance

The performance is analyzed from two aspects: queue stability and constraint satisfaction. In this group of experiments, the number of devices n is set as 100, the bandwidth B is set as 2 MHz, the number of sub-channels in each slot is generated randomly, and $S(t) \sim U[5, 10]$. In our problem, the queue length constraints of all devices are set from 50000 to 100000 in equal steps, that is, $q^i_{cons} = 50000 + 50000(i-1)/(n-1)$. We select five from the 100 devices, which are marked as MD_{10}, MD_{30}, MD_{50}, MD_{70}, MD_{90}, and their corresponding queue length thresholds are approximate to 55000, 65000, 75000, 85000, and 95000, respectively. In Fig. 4, we plot the changing trend of three important performance metrics of the five devices from time slots 0 to 100, where Fig. 4(a) shows the queue length of the five devices over time, Fig. 4(b) shows the changing trend of energy consumption of the selected devices, and Fig. 4(c) shows the virtual length backlog of the five devices.

From Fig. 4(a), we can see that the queue length of each device becomes stable quickly and the queue length constraints of all devices

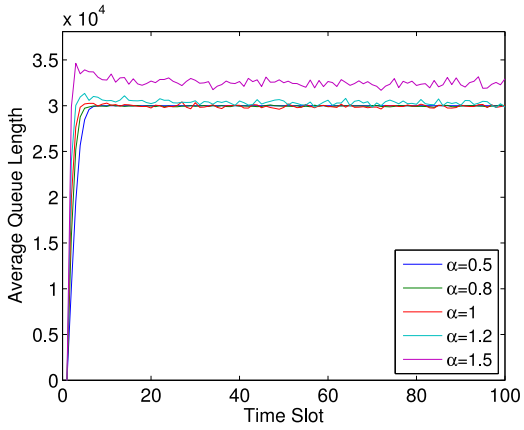


(a) Actual queue length.

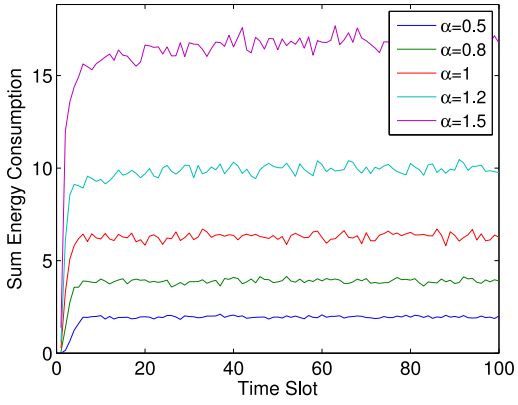


(b) Energy consumption.

Fig. 5. Queue length and energy consumption under different V.



(a) Average queue length.



(b) Energy consumption.

Fig. 6. Queue length and energy consumption with different arrival rates.

are satisfied. For example, the queue length constraint of MD_{30} is 22000, and the actual queue length keeps stable at 22000. The figures show that the energy consumption and the queue length can reach stability quickly, and the virtual queue backlog is upper bound.

6.2.2. Analysis on the dynamic adaptability

As mentioned before, by introducing the queue length threshold, we can artificially adjust and control the system performance with respect to task response time. In the group of experiments, the queue length thresholds of all devices are set from 400000 to 600000 in equal steps at the beginning. To verify the adaptability of the proposed algorithm to the dynamical performance requirement, we adjust the threshold values for all devices at time slots 200 and 400 such that the average threshold of all devices becomes 300000 and 100000, respectively. Lastly, the queue length thresholds of all devices are reset to the initial values at time slot 600.

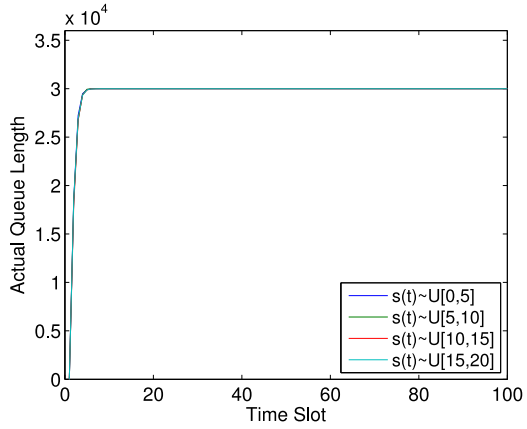
Fig. 5 shows the changing trends of the average task queue length and the sum energy consumption of all devices under the changing threshold settings. From Fig. 5(a), we can see that the task queue length of devices can stabilize at the target thresholds rapidly after each change. Correspondingly, the sum energy consumption shows a sharp increase when the threshold changes to a smaller value and then returns to the original steady state. That is because task processing should speed up to rapidly reach the new steady state when the threshold is set smaller. That is the reason that the sum of energy consumption increases sharply. On the contrary, when the thresholds are set greater at time slot 600, the energy consumption decreases first and then stabilizes at the original level. That is because the amount of tasks processed in unit time is reduced to reach the new stable state.

6.3. Parameter analysis

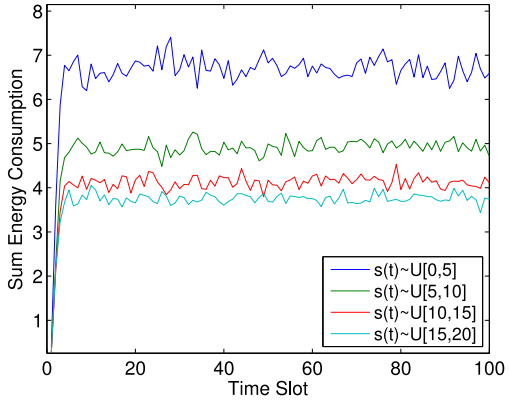
To verify the adaption of the proposed dynamic computation offloading algorithm to the variation of different parameters, the experiments are done in several groups. In each group, we change the value of a parameter and fix the others to observe the effect of the changing parameter on two performance indicators, i.e., the total energy consumption and the average queue length of all devices.

6.3.1. Effect of arrival rate

Fig. 6 shows the changing trend of the two performance indicators with different data arrival rates. The arrival rate of device i is set as $\alpha A_i(t)$ where $A_i(t) \sim U[0, 40000]$ and $\alpha = 0.5$, 0.8, 1, 1.2, and 1.5, respectively. Fig. 6(a) shows that the queue length constraint can be satisfied when the arrival rate is increasing from $0.5A_i(t)$ to $1.2A_i(t)$ (Notice that the queue length thresholds of all devices are set from 20000 to 40000 in the step of $20000/(n-1)$. The mean of all threshold values is 30000, so the result is as expected.). However, when the arrival rate increases to $1.5A_i(t)$ further, the average queue length is stabilized

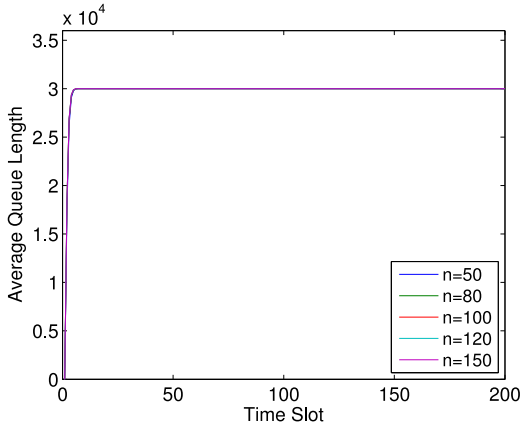


(a) Average queue length.

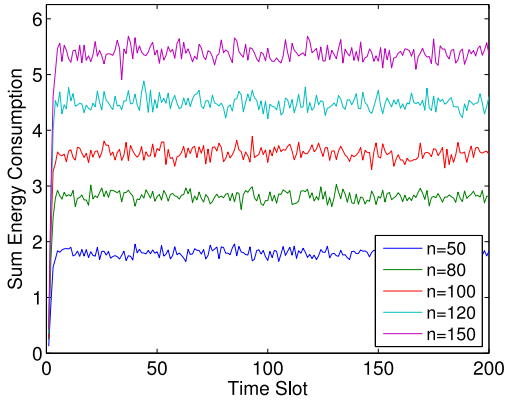


(b) Energy consumption.

Fig. 7. Queue length and energy consumption with different number of sub-channels.



(a) Average queue length.



(b) Energy consumption.

Fig. 8. Queue length and energy consumption with different number of devices.

around 33000, which exceeds the defined threshold. That is because the total data processing capacity is fixed. When the data arrival rate becomes greater, it is harder for the system to maintain a short queue length.

Fig. 6(b) shows that the sum energy consumption increases with the increase of data arrival rate. That is because to maintain the queue length constraints, more tasks should be processed in unit time as the task arrival rate increases, which leads to an increase in energy consumption. Moreover, we can see that the change in arrival rate does not affect the rate of convergence. Both of the two performance indicators reach stable very quickly.

6.3.2. *Effect of number of sub-channels*

In Fig. 7, two performance indicators with different numbers of sub-channels are presented. The number of available sub-channels is set to follow uniform distribution within different ranges, i.e., $S_i(t) \sim U[0,5]$, $U[5,10]$, $U[10,15]$, and $U[15,20]$, respectively.

Fig. 7(a) shows that the queue length constraints can be guaranteed when the number of available sub-channels changes. That is because the system load does not exceed the task processing capacity in all cases. Fig. 7(b) shows that as the number of sub-channels increases, the sum energy consumption decreases as well. That is because more available sub-channels provide higher offloading capability such that more tasks can be offloaded to MEC for remote execution. Under the given parameter settings, offloading tasks to MEC is beneficial to energy saving. Hence, the sum of energy consumption per unit time is reduced as the number of sub-channels increases.

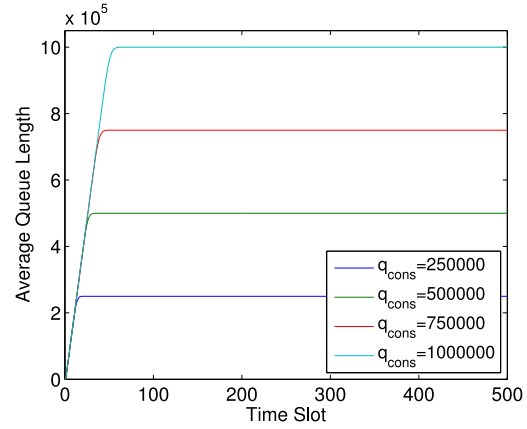
6.3.3. *Effect of number of devices*

Fig. 8 plots the sum energy consumption and average queue length with different numbers of devices. In this group of experiments, the number of devices n is set as 50, 80, 100, 120, and 150, respectively. From Fig. 8(a) we can see that the average queue length of devices remains unchanged. That is because no matter how many devices are, the queue length constraint is set from 20000 to 40000 in equal increments and the average queue length of all devices is 30000. Fig. 8(b) shows that as the number of devices rises, the number of tasks to be processed increases, which leads to an increase in energy consumption.

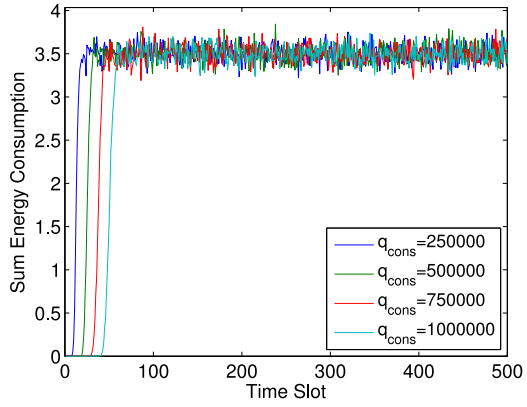
6.3.4. *Effect of queue length threshold*

We define a queue length threshold for each device to make the system performance in task queuing time adjustable and controllable. According to Little's law, the average queuing delay is proportional to queue backlogs. By adjusting the value of the queue length threshold, we can achieve the expected task response time. Consequently, the energy consumption varies with different thresholds. To analyze the effect of the queue length threshold on the total energy consumption, we conduct another group of experiments. For simplicity, all devices are set to the same threshold values in the group of experiments, and the threshold values are set from 250000 to 1000000 in an increment of 250000. The other parameters are set as Fig. 4. The experimental results are shown in Fig. 9.

Fig. 9(a) shows that under different threshold values, the queue length of all devices increases until it reaches and stabilizes at the predefined queue length threshold. Moreover, the higher the threshold is, the slower the queue stabilizes. For example, the queue reaches



(a) Average queue length.



(b) Energy consumption.

Fig. 9. Queue length and energy consumption with different threshold.

stable at time slot 75 when the threshold is 25000, while it reaches stable at time slot 250 when the threshold is 100000. Fig. 9(b) shows the total energy consumption of all devices at each time slot. The figure shows that the devices consume less energy under a greater threshold before time slot 250. That is because it takes longer for the queues to reach a steady state under a greater threshold. Before the queue length stabilizes at the threshold, the tasks processed in each time slot are much less than that processed in a stable time slot. Consequently, the energy consumption in unstable time slots is much less. Moreover, the sum energy consumption becomes stable and identical lastly under all queue length thresholds. That is because once the queues reach steady states, the processed tasks should be consistent with the arrived tasks such that the system can maintain stability.

6.3.5. Convergence analysis

The above experiments only present the results of the cases where the queues can reach stability. If the time-average system workload exceeds the system processing capacity, the system will never reach a steady state. For instance, the time-average arrival rate of devices increases to above 50000, or the transmission capability is reduced further by reducing the bandwidth or the number of sub-channels. In these cases, the task queue will grow indefinitely. The reasons are apparent, so we do not give detailed experimental results here.

6.4. Comparison experiments

To further evaluate the QC-EEDOA's performance, we compare QC-EEDOA with two baseline algorithms, and EEDOA proposed in [17].

Table 2
The comparison of queue length and energy consumption of different algorithms.

Algorithm	QC-EEDOA	DW	EA	EEDOA
Queue Length (bit)	3.000×10^4	17.986	2.242×10^4	∞
Energy Consumption (J)	1.505	1.887	2.182	0.577

- **The EEDOA strategy:** In the strategy, all tasks are offloaded to MEC. By applying Lyapunov optimization, the offloading decision is generated in each time slot to maintain system stability.
- **Equal allocation strategy (EA):**In slot t, the offloading duration is equally allocated among all the devices.
- **Difference-weighted strategy (DW):** In slot t, the offloading duration is allocated among all the devices according to the weighted difference between queue length and threshold.

Notice that the first two strategies give a higher priority to remote offloading, and the rest of the tasks are executed locally as much as possible such that the amount of tasks processed per time slot is maximized.

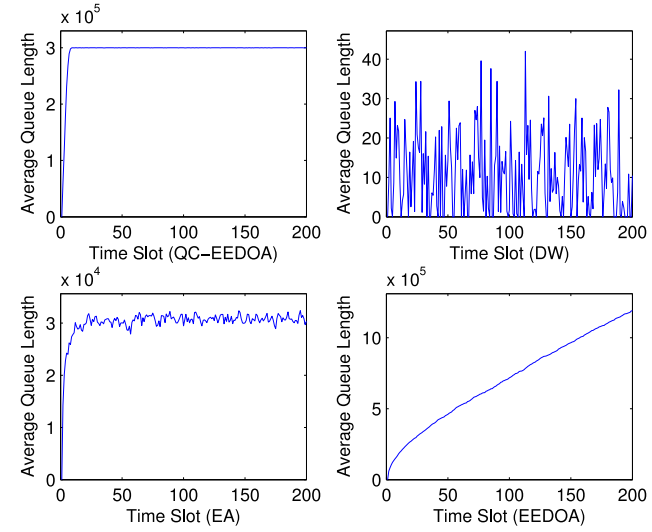
In the following experiments, the task amount generated by each device in each time slot is valued randomly within $[30000, 90000]$ bits. The other parameters are set to the same values with Fig. 4. For QC-EEDOA, we define an individual queue length constraint for each device and $q^i_{cons} = 200000 + 200000(i-1)/(n-1)$.

Table 2 gives the average queue length and energy consumption when the system tends to stabilize. We can see that QC-EEODA keeps the queue length near $\frac{1}{n}\sum_{i=1}^{n} q^i_{cons}$ and minimizes the energy consumption of the system. Compared to DW and EA, QC-EEDOA can reduce the system energy consumption by 20% to 30%. While EEDOA significantly reduces the energy consumption of the system, the queue length cannot be guaranteed.

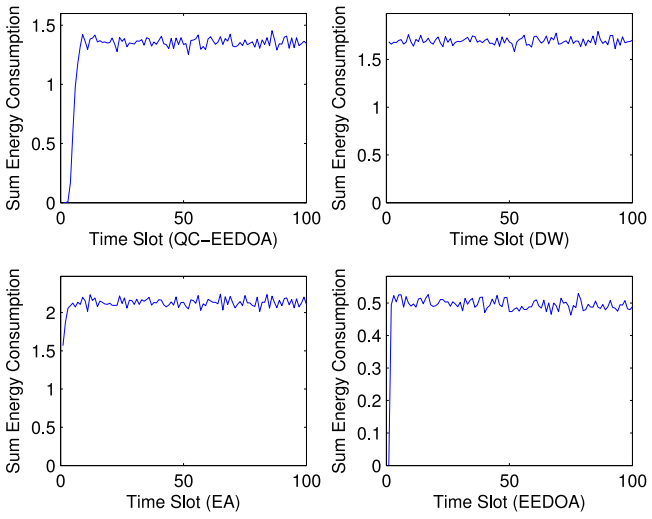
Fig. 10 presents the changing trends of the average queue length and sum energy consumption of the four algorithms, respectively. From Fig. 10(a), we can see that QC-EEDOA can stabilize the queue length around the threshold effectively. EA also can maintain the system stability, but the queue length at the steady state is uncertain. Under the DW strategy, the tasks generated in each time slot can almost be completed at the current time, so the queue length is minimal and negligible. Since the EEDOA strategy only offloads tasks to MEC for remote execution, its processing capacity is seriously inadequate under a heavy task load. The unprocessed tasks are stored in the task buffer queue, so the queue length grows over time. Fig. 10(b) shows that the energy consumption of the four strategies can reach stable after several time slots. Among all strategies, EEDOA consumes the least energy. That is because the tasks are only offloaded to MEC and generate transmission energy consumption. Apart from EEDOA, our strategy can reduce energy consumption effectively compared with the EA and DW strategies.

7. Conclusion

In this paper, we study the computation offloading problem for a multi-device single-MEC system. Due to the dynamic system load and the time-varying wireless channel quality, an online energy-efficient computation offloading algorithm is proposed to approximate the minimal energy consumption while the system keeps in such a stable state where the individual queue length constraints of all devices are satisfied. This algorithm requires no priori statistic information related to task arrival or channel condition but can adjust the ratio of tasks executed locally and offloaded to MEC in real time based on the system load and channel quality. The proposed algorithm is valid only when the average system load in the long term is within the system computing capacity. Experiments results show the proposed algorithm can reduce the energy consumption effectively and the queue length constraints can be guaranteed. Compared to DW and EA, QC-EEDOA can reduce the system energy consumption by 20% to 30%.



(a) Average queue length.



(b) Energy consumption.

Fig. 10. The comparison of queue length and energy consumption of different algorithms.

Declaration of competing interest

We wish to draw the attention of the Editor to the following facts which may be considered as potential conflicts of interest and to significant financial contributions to this work. [OR]

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing, we confirm that we have followed the regulations of our institutions concerning intellectual property.

We further confirm that any aspect of the work covered in this manuscript that has involved either experimental animals or human patients has been conducted with the ethical approval of all relevant bodies and that such approvals are acknowledged within the manuscript.

We understand that the Corresponding Author is the sole contact for the Editorial process (including Editorial Manager and direct communications with the office). He/she is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs. We confirm that we have provided a current, correct email address which is accessible by the Corresponding Author.

Data availability

The authors are unable or have chosen not to specify which data has been used.

Acknowledgments

The authors thank the editors and reviewers for their insightful comment and valuable suggestions. This work was supported by the Program of National Natural Science Foundation of China (Grant No. 62072174, 62372172), Distinguished Youth Science Foundation of Hunan Province, China (Grant No. 2023JJ10030), National Natural Science Foundation of Hunan Province, China (Grant No. 2022JJ40278, 2022JJ30398, 2021JJ30455), Scientific Research Foundation of Hunan Provincial Education Department, China (Grant No. 22A0026), Key Scientific Research Project of Hunan Provincial Education Department, China(Grant No. 22A0056).

References

[1] M.R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, L. Ladid, Internet of Things in the 5G era: Enablers, architecture and business models, IEEE J. Sel. Areas Commun. 34 (3) (2016) 510–527.

[2] D. Chen, N. Zhang, Z. Qin, X. Mao, Z. Qin, X. Shen, X.-y. Li, S2M: A lightweight acoustic fingerprints-based wireless device authentication protocol, IEEE Internet Things J. 4 (1) (2017) 88–100.

[3] H. Duan, Y. Zheng, C. Wang, X. Yuan, Treasure collection on foggy islands: Building secure network archives for Internet of Things, IEEE Internet Things J. 6 (2) (2019) 2637–2650.

[4] F. Wang, J. Xu, S. Cui, Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems, IEEE Trans. Wireless Commun. 19 (4) (2020) 2443–2459.

[5] M. Chiang, T. Zhang, Fog and IoT: An overview of research opportunities, IEEE Internet Things J. 3 (6) (2016) 854–864.

[6] S. Bi, Y.J. Zhang, Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading, IEEE Trans. Wireless Commun. 17 (6) (2018) 4177–4190.

[7] Z. Sheng, C. Mahapatra, V. Leung, C. Min, P.K. Sahu, Energy efficient cooperative computing in mobile wireless sensor networks, IEEE Trans. Cloud Comput. 6 (99) (2018) 114–126.

[8] H. Yu, Q. Wang, S. Guo, Energy-efficient task offloading and resource scheduling for mobile edge computing, in: 2018 IEEE International Conference on Networking, Architecture and Storage, NAS, 2018, pp. 1–4.

[9] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1628–1656.

[10] J. Zhang, X. Zhao, An overview of user-oriented computation offloading in mobile edge computing, in: 2020 IEEE World Congress on Services, SERVICES, 2020, pp. 75–76.

[11] T.Q. Dinh, J. Tang, Q.D. La, T. Quek, Offloading in mobile edge computing: Task allocation and computational frequency scaling, IEEE Trans. Commun. 65 (8) (2017) 3571–3584.

[12] X. Tao, K. Ota, M. Dong, H. Qi, K. Li, Performance guaranteed computation offloading for mobile-edge cloud computing, IEEE Wirel. Commun. Lett. 6 (6) (2017) 774–777.

[13] C. Liu, K. Li, J. Liang, K. Li, COOPER-MATCH: Job offloading with a cooperative game for guaranteeing strict deadlines in MEC, IEEE Trans. Mob. Comput. PP (99) (2019) 1.

[14] K. Li, Heuristic computation offloading algorithms for mobile users in fog computing, ACM Trans. Embed. Comput. Syst. 20 (2) (2021) 1–28.

[15] Z. Tong, X. Deng, J. Mei, L. Dai, K. Li, K. Li, Stackelberg game-based task offloading and pricing with computing capacity constraint in mobile edge computing, J. Syst. Archit. 137 (2023) 102847, http://dx.doi.org/10.1016/j.sysarc.2023.102847.

[16] S. Bi, L. Huang, H. Wang, Y.-J.A. Zhang, Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks, IEEE Trans. Wireless Commun. 20 (11) (2021) 7519–7537.

[17] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, X. Shen, Energy efficient dynamic offloading in mobile edge computing for Internet of Things, IEEE Trans. Cloud Comput. 9 (3) (2021) 1050–1060.

[18] X. Lin, J. Wu, J. Li, W. Yang, M. Guizani, Stochastic digital-twin service demand with edge response: An incentive-based congestion control approach, IEEE Trans. Mob. Comput. 22 (4) (2023) 2402–2416.

[19] J. Ren, K.M. Mahfujul, F. Lyu, S. Yue, Y. Zhang, Joint channel allocation and resource management for stochastic computation offloading in MEC, IEEE Trans. Veh. Technol. 69 (8) (2020) 8900–8913.

[20] Y. Sun, S. Zhou, J. Xu, EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2637–2646.

[21] L. Chen, S. Zhou, J. Xu, Computation peer offloading for energy-constrained mobile edge computing in small-cell networks, IEEE/ACM Trans. Netw. PP (4) (2018) 1–14.

[22] W. Liang, L. Cui, F.P. Tso, Low-latency service function chain migration in edge-core networks based on open Jackson networks, J. Syst. Archit. 124 (2022) 102405.

[23] H. Wu, J. Chen, T.N. Nguyen, H. Tang, Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems, IEEE Trans. Ind. Inform. 19 (2) (2023) 2117–2128.

[24] K. Guo, R. Gao, W. Xia, T.Q.S. Quek, Online learning based computation offloading in MEC systems with communication and computation dynamics, IEEE Trans. Commun. 69 (2) (2021) 1147–1162.

[25] Z. Chang, L. Liu, X. Guo, Q. Sheng, Dynamic resource allocation and computation offloading for IoT fog computing system, IEEE Trans. Ind. Inform. 17 (5) (2021) 3348–3357.

[26] A. Islam, A. Debnath, M. Ghose, S. Chakraborty, A survey on task offloading in multi-access edge computing, J. Syst. Archit. 118 (2021) 102225.

[27] F. Zhao, Y. Chen, Y. Zhang, Z. Liu, X. Chen, Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 2154–2165.

[28] W. Zhou, L. Xing, J. Xia, L. Fan, A. Nallanathan, Dynamic computation offloading for MIMO mobile edge computing systems with energy harvesting, IEEE Trans. Veh. Technol. 70 (5) (2021) 5172–5177.

[29] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, W. Zhuang, Learning-based computation offloading for IoT devices with energy harvesting, IEEE Trans. Veh. Technol. 68 (2) (2019) 1930–1941.

[30] Z. Tong, X. Deng, J. Mei, L. Dai, K. Li, K. Li, Stackelberg game-based task offloading and pricing with computing capacity constraint in mobile edge computing, J. Syst. Archit. (2023).

[31] K. Li, Optimal task execution speed setting and lower bound for delay and energy minimization, J. Parallel Distrib. Comput. 123 (2019) 13–25.

[32] Z. Tong, J. Cai, J. Mei, K. Li, K. Li, Dynamic energy-saving offloading strategy guided by Lyapunov optimization for IoT devices, IEEE Internet Things J. (2022) 1.

[33] L. Qiang, H. Tao, N. Ansari, W. Gang, On designing energy-efficient heterogeneous cloud radio access networks, IEEE Trans. Green Commun. Netw. 2 (3) (2018) 721–734.

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

[35] C. Singhal, S. De, Resource Allocation in Next-Generation Broadband Wireless Access Networks, IGI Global, Hershey, United States, 2017.

[36] X. Lyu, W. Ni, H. Tian, R.P. Liu, X. Wang, G.B. Giannakis, A. Paulraj, Optimal schedule of mobile edge computing for Internet of Things using partial information, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2606–2615.

[37] J. Zhou, J. Fan, J. Wang, Task scheduling for mobile edge computing enabled crowd sensing applications, Int. J. Sens. Netw. 35 (2) (March 2021) 88–98.

[38] M. Guo, W. Wang, X. Huang, Y. Chen, L. Zhang, L. Chen, Lyapunov-based partial computation offloading for multiple mobile devices enabled by harvested energy in MEC, IEEE Internet Things J. 9 (11) (2022) 9025–9035.

[39] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, X. Wang, Learning-aided computation offloading for trusted collaborative mobile edge computing, IEEE Trans. Mob. Comput. 19 (12) (2020) 2833–2849.

[40] J. Mei, L. Dai, Z. Tong, X. Deng, K. Li, Throughput-aware dynamic task offloading under resource constant for MEC with energy harvesting devices, IEEE Trans. Netw. Serv. Manag. (2023) http://dx.doi.org/10.1109/TNSM.2023.3243629.

[41] M. Neely, Stochastic Network Optimization With Application to Communication and Queueing Systems, Morgan & Claypool, 2010.

Jing Mei received the Ph.D. degree in computer science from Hunan University, China, in 2015. She is currently an associate professor in the College of Information Science and Engineering in Hunan Normal University. Her research interests include cloud computing, fog computing and mobile edge computing, high performance computing, task scheduling and resource management, etc. She has published more than 30 research articles in international conference and journals, such as *IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed System, IEEE Transactions on Service Computing, Cluster Computing, Journal of Grid Computing, Journal of Supercomputing*.

Longbao Dai received the B.S. degree in computer science and technology from Hunan University of Science and Engineering, Yongzhou, China, in 2021. He is currently working toward the M.S. degree at the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research interests focus on distributed parallel computing, modeling and resource pricing and allocation in mobile edge computing systems, and combinatorial optimization.

Zhao Tong received the Ph.D. degree in computer science from Hunan University, Changsha, China in 2014. He was a visiting scholar at the Georgia State University from 2017 to 2018. He is currently an associate professor at the College of Information Science and Engineering of Hunan Normal University, the young backbone teacher of Hunan Province, China. His research interests include parallel and distributed computing systems, resource management, big data and machine learning algorithm. He has published more than 25 research papers in international conferences and journals, such as IEEE-TPDS, Information Sciences, FGCS, NCA, and JPDC, PDCAT, etc. He is a senior member of the China Computer Federation (CCF) and a Member of the IEEE.

Lianming Zhang received his Ph.D. degree from the School of Information Science and Engineering of Central South University, China, and his M.Sc. and B.Sc. degrees from the Department of Physics of Hunan Normal University. He is currently a professor in the College of Information Science and Engineering of Hunan Normal University, China. His main research interests include network intelligence, software-defined networking, and edge computing.

Keqin Li is a SUNY Distinguished Professor of Computer Science with the State University of New York. He is also a National Distinguished Professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has authored or co-authored over 850 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top 5 most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an associate editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Services Computing*, and the *IEEE Transactions on Sustainable Computing*. He is an IEEE Fellow and an AAIA Fellow. He is also a Member of Academia Europaea (Academician of the Academy of Europe).