Mu and Zhong *J Wireless Com Network* (2020) 2020:247
https://doi.org/10.1186/s13638-020-01865-4

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Computation offloading to edge cloud and dynamically resource-sharing collaborators in Internet of Things

Siqi Mu[1,2*] and Zhangdui Zhong[1]

*Correspondence:
musiqi@bjtu.edu.cn
[1] State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Haidian District, 100044 Beijing, People's Republic of China
Full list of author information is available at the end of the article

Abstract

With the diversity of the communication technology and the heterogeneity of the computation resources at network edge, both the edge cloud and peer devices (collaborators) can be scavenged to provide computation resources for the resource-limited Internet-of-Things (IoT) devices. In this paper, a novel cooperative computing paradigm is proposed, in which the computation resources of IoT device, opportunistically idle collaborators and dedicated edge cloud are fully exploited. Computation/offloading assistance is provided by collaborators at idle/busy states, respectively. Considering the channel randomness and opportunistic computation resource share of collaborators, we study the stochastic offloading control for an IoT device, regarding how much computation load is processed locally, offloaded to the edge cloud and a collaborator. The problem is formulated into a finite horizon Markov decision problem with the objective of minimizing the expected total energy consumption of the IoT device and the collaborator, subject to satisfying the hard computation deadline constraint. Optimal offloading policy is derived based on the stochastic optimization theory, which demonstrates that the energy consumption can be reduced by a proportional factor through the cooperative computing. More energy saving is achieved with better wireless channel condition or higher computation energy efficiency of collaborators. Simulation results validate the optimality of the proposed policy and the efficiency of the cooperative computing between end devices and edge cloud, compared to several other offloading schemes.

Keywords: Mobile edge computing, Cooperative computing, Internet of Things, Stochastic offloading control

1 Introduction

Nowadays, Internet of things (IoT), which connects billions or even trillions of smart devices and objects embedded with sensors, is evolving rapidly. IoT devices, including wearable devices, laptops, sensors, smart cars and industrial components, etc., have a demand to support more and more intelligent services and applications. However, most of intelligent services, such as e-health, automatic driving and industrial automation, are computation-intensive, fast developing and outgrowing the computing and storage capabilities of IoT devices [1]. To address the issue, mobile edge computing (MEC)

has risen as a promising technology that provides cloud computing capabilities in close proximity to the data sources. Through one-hop transmissions, resource-constrained IoT devices can offload their application-related data to edge clouds deployed at network edges, such as base stations or access points. Compared with mobile cloud computing that requires IoT devices to access remote cloud servers through the Internet, mobile edge computation offloading (MECO) can potentially reduce the transmission latency between IoT devices and the servers, which is especially important for supporting time-constraint applications. Considering massive device access in current wireless networks, plenty of literatures have studied the offloading decision and resource allocation problem [2–6]. However, most of them take no consideration on the time-varying characteristics of channel conditions in the process of computation offloading, which limits the scope of application for their conclusions.

On the other hand, peer-to-peer computing is also regarded as an efficient approach for computation offloading [7]. Through some proximate communication technologies, such as device to device (D2D) communication and bluetooth, the idle resources on nearby IoT devices, i.e., *collaborators*, can be exploited to enhance the offloading efficiency. In this way, the cost on additional deployment of the edge servers at network edges can be cut down. Meanwhile, the traffic burden and computation load on the edge base station and servers are effectively alleviated. Most of the existing works [8–11] on peer-to-peer computing mainly consider that collaborators offer constant idle computation resources until completing the offloaded computation loads. However, due to its own tasks with high priority, the central processing unit (CPU) state of collaborators is opportunistically idle and available to the offloaded tasks [12, 13], which is different from the dedicated resource provisioning of edge servers for IoT subscribers. Moreover, with some incentive mechanism [14], collaborators not only share idle computation resources, and their network function can also be utilized to assist computation offloading.

Considering the stochastic channel condition and the dynamic computation resources share, in this paper, we propose the cooperative computing which integrates both the dedicated edge cloud and opportunistic collaborators to provide computation resources for IoT devices. In particular, collaborators provide two functions: *computation assistance* at idle states and *offloading assistance* at busy states. With given computation load and completion time constraint, an IoT device should adapt the local computation load and offloading computation load to the dynamic process of the channel states and the collaborator states. Our contributions are summarized as follows:

1 A novel cooperative computing paradigm is proposed, which supports the parallel computing on local devices, edge cloud and collaborators with dynamic computation resources. A collaborator assists in computing or further offloading the computation to the edge cloud based on its stochastic CPU availability.
2 A dynamic offloading decision problem on how much computation load is executed locally, offloaded to edge cloud and a collaborator, is modeled as a finite horizon Markov decision problem. We aim to minimize the expected sum energy consumption of both the IoT device and the collaborator, subject to the computation completion time constraint.

3 Based on the stochastic optimization theory, the optimal offloading policy is derived by Karush–Kuhn–Tucker (KKT) conditions and backward induction, which facilitates the design of a low-complexity dynamic programming algorithm and alleviates the well-known curse of dimensionality.
4 Optimal offloading policy shows that the energy consumption can be reduced by a proportional factor through cooperative computing. More energy saving is achieved with better wireless channel condition or higher computation energy efficiency of collaborators. Simulation results validate the optimality of the proposed policy and the efficiency of the cooperative computing between end devices and edge cloud.

The rest of the paper is organized as follows. Section 2 summarizes some related work. Section 3 describes the system model and problem formulation. Optimal offloading policy and the corresponding algorithm are presented in Sect. 4. In Sect. 5, the simulation experiments and performance evaluation of the proposed algorithm are provided. Finally, conclusions and future work are given in Sect. 6.

2 Related work

In this section, some related work on MECO and peer-to-peer computing is summarized. Considering computation offloading to dedicated edge servers, [2] investigated the partial offloading for a mobile device with dynamic voltage scaling technology. The offloading ratio, computational speed and transmit power of the device are optimized for minimizing its energy consumption or application execution latency. For multiple users that require offloading services, time-division multiple access (TDMA) and frequency-division multiple access (FDMA) are adopted in [3]. A convex optimization problem was formulated and solved optimally for joint offloading ratio and time allocations, such that the total energy consumption of multiple devices is minimized. A heuristic algorithm was then proposed for offloading ratio and frequency channel allocations. Decentralized algorithms for resource allocation and offloading decision were studied in [4, 5] by using game theory and decomposition techniques, respectively. These works focused on the quasi-static scenario where channel or link quality remains unchanged in the offloading process. Targeting at time-varying channel conditions when computation offloading, [15] investigated binary decision, i.e., offloading or local computing, in Gilbert-Elliott channel model. Then the authors considered linear task topology and random channel in [16], and the problem of optimal offloading control was formulated as a shortest path problem. "One-climb" policy structure was proved that the tasks should only migrate at most once between an end device and the cloud. Considering the intermittent connectivity between an end device and edge cloud due to network congestions, an analytical framework was built in [17] and a closed-form expression of the task processing time under different network conditions was derived.

For peer-to-peer computing, D2D-enabled collaborative computation was studied in [8], where a local user offloads multiple independent tasks to its nearby devices through TDMA. Computation latency was minimized by optimizing task assignment jointly with the time for task offloading and results downloading, subject to the individual energy constraints at the local user and the collaborators. In [9], we investigated the application

partitioning and collaborator selection problem for multiple users when multiple idle collaborative devices are available. Both the centralized and decentralized algorithms were proposed to minimize the energy consumption of all the users. Considering a collaborator may opportunistically provide idle computation resources due to its local computation; [12] studied a user exploits non-causal information on the CPU state of collaborator to control offloaded data sizes. The CPU information was assumed to be predicted accurately in offline. Instead, the offloading policy was then studied in [13] based on the statistic distribution on the dynamic collaborator's CPU state and the random channel state.

Integrating nearby collaborators with dedicated servers, recent studies show the efficiency of this cooperative computing. In [10], the peer-to-peer offloading was integrated with edge severs to maximize the number of devices that the system can support, where parts of the task that cannot be completed in time by each local device are offloaded to an edge node and a nearby device. The optimal transmit power allocation and the task offloading strategy are obtained. The selection on application execution approaches, including local computing, offloading to a collaborator or an edge cloud for multiple users, was considered in [11]. The problem was formulated as a sequential game to minimize the weighted sum of the task executive delay and energy consumption. Apart from utilizing the computing resources of the collaborators, network function was also considered in [18]. Through time allocation, portions of data are transmitted from a mobile device to a collaborator for computation at the first time block; then, the collaborator assists another portions of data transmission from the mobile device to an AP as a relay at the subsequent two time blocks. These existing works on cooperative computing considered static scenarios with no channel varying and assumed collaborator offers constantly dedicated idle computation resources similar as edge/cloud servers. However, due to signal interference and user mobility, as well as stochastic tasks arrival, the channel dynamics and the randomness of collaborator states should be considered in offloading policy design, which motivates our study in this work. We model the dynamic offloading control as a finite horizon MDP problem. Different with some existing work [19–21] that solved MDP problems numerically and sub-optimally using deep reinforcement learning, we aim to derive the optimal solution that yields useful insight into the policy structure.

3 System model

We consider an edge computing system with an IoT device, a collaborator and an edge cloud server that is co-located at an access point (AP). The IoT device has computationally intensive applications to be completed within the hard deadline constraint, and the application-related computing data are stored in its buffer. In this paper, we focus on data-partitioned applications, such as virus scan, file/figure compression and text conversion [2, 22, 23], for which the application data can be partitioned continuously and processed in parallel. The edge cloud provides the dedicated computation services for the resources-limited IoT device, and the resources are sufficient for completing the workload from the device. The collaborator can also share its computation capability to assist the computation of the IoT device. However, due to its own randomly arriving computation tasks, the available idle computation resources of the collaborator for assistance are dynamic. To model

the dynamic property of wireless channel and collaborator state, the system time is divided to slots with equal duration τ. The channel and collaborator conditions remain constant within a time slot but may vary between different time slots. Consider an application with data size W should be completed before deadline D, i.e., $T = D/\tau$ time slots. In each time slot t, a_t^{L} bits of data are processed locally by the IoT device. In parallel, through some existing cellular connectivity and proximate communication technology, such as LTE and D2D communication, the IoT device offloads a_t^{E} and a_t^{C} bits to the edge server and the collaborator, respectively. At the next time slot $t+1$, the remaining data size in the buffer of the IoT device can be expressed as $B_{t+1} = B_t - a_t^{\mathsf{L}} - a_t^{\mathsf{E}} - a_t^{\mathsf{C}}$.

3.1 Local computing

According to [24], utilizing dynamic voltage and frequency scaling (DVFS) techniques, the local CPU frequency can be dynamically adjusted in each time slot in order to minimize the computation energy consumption. Let γ denote the number of CPU cycles for computing per data bit, then the local CPU-cycle frequency of the IoT device at time slot t is $c_t^{\mathsf{L}} = \frac{\gamma a_t^{\mathsf{L}}}{\tau}$. The local computation power consumption P_t^{L} can be modeled as $P_t^{\mathsf{L}} = \kappa^{\mathsf{L}}(c_t^{\mathsf{L}})^3$ [25], where κ^{L} is the energy conversion coefficient of the device depending on the chip architecture. The larger κ^{L} is, the lower its computation energy efficiency is. Then the local computation energy consumed by the IoT device at time slot t is expressed as

$$\mathcal{E}_t^{\mathsf{L}}(a_t^{\mathsf{L}}) = P_t^{\mathsf{L}}\tau = \beta^{\mathsf{L}}(a_t^{\mathsf{L}})^3 \tag{1}$$

where $\beta^{\mathsf{L}} = \frac{\kappa^{\mathsf{L}}\gamma^3}{\tau^2}$.

3.2 Direct offloading to edge cloud

Due to the effect of shadowing, multipath interference and mobility of the IoT device, the correlated fading channels are considered in this paper. Such correlated fading wireless communication channels can be modeled by the finite state Markov chain (FSMC) model with K states, indexed as $k = \{0, 1, 2, \ldots, K\}$. The channel state between the IoT device and the edge cloud H_t^{le} keeps constant during one time slot t, but may change in different time slots. The transition probability from two neighboring states $k-1$ and k is denoted as $P_{k-1,k}$, $1 \le k \le K$. These probabilities can be calculated based on practical fading models [26]. The transition probability between two states that are not neighbors is 0. According to the empirical model in the literature, the transmission power of the IoT device to the edge cloud at t can be modeled as the following monomial function:

$$P_t^{\mathsf{T}} = \lambda \frac{s_t^m}{H_t^{le}} \tag{2}$$

where λ is the energy coefficient incorporating the effects of bandwidth and noise power, $s_t = a_t^{\mathsf{E}}/\tau$ is the transmission rate at time slot t, and m is the monomial order determined by the modulation-and-coding scheme. It is shown by [27, 28] that the power-rate relation can be well approximated by the monomial function. Following [13], the monomial order of ($m = 3$) is adopted to approximate transmission power, such that the coding scheme for the targeted error probability is less than 10^{-6}. Then the offloading energy consumption from the IoT device to the edge cloud at time slot t is written as

$$\mathcal{E}_t^{\mathsf{E}}(a_t^{\mathsf{E}}) = P_t^{\mathsf{T}}\tau = \eta\frac{(a_t^{\mathsf{E}})^3}{H_t^{le}} \tag{3}$$

where $\eta = \frac{\lambda}{\tau^2}$. We omit the computation energy consumption of the edge cloud since it is often powered by stable on-grid power.

3.3 Collaborator-assisted computing or offloading

Since the computation resource availability of collaborator in the future is often variable and unknown, statistic information based on historical data can be utilized. As shown in [29], the idle/busy intervals in the quantized CPU utilization traces roughly follow the geometric distribution, while the idle/busy intervals in Markov chains are exactly geometrically distributed. Hence, the CPU availability processes can be modeled as two-state Markov chains [13, 29]. Based on the idle/busy state distribution, the state transition probability can be also obtained by theoretic derivation [17] or real-world experiments [29]. In each time slot t, the CPU state of the collaborator is $C_t \in \{0, 1\}$, where $C_t = 0$ and $C_t = 1$ denote the idle and busy states, respectively. The current CPU state C_t only depends on the previous random state C_{t-1} and is independent of the past states $\{C_1, \ldots, C_{t-2}\}$. Let P_{bb} and P_{ii} denote the busy-to-busy and idle-to-idle transition probabilities. Then the busy-to-idle and idle-to-busy transition probabilities are $P_{bi} = 1 - P_{bb}$ and $P_{ib} = 1 - P_{ii}$, respectively. When $C_t = 0$, the CPU is idle and the offloaded computation from the IoT device can be processed by the collaborator; otherwise, the collaborator will further forward the computation to the edge cloud. The system model is shown as Fig. 1.

We now consider the assisted computing when $C_t = 0$. Recall that a_t^{C} data bits are offloaded to the collaborator in time slot t. Similar with the transmission energy consumption model described in Sect. 3.2 and computation energy consumption model described in Sect. 3.1, the total energy consumption of the IoT device and the collaborator at time slot t is expressed as

$$\mathcal{E}_t^{\mathsf{CC}}(a_t^{\mathsf{C}}) = \eta\frac{(a_t^{\mathsf{C}})^3}{H_t^{lc}} + \beta^{\mathsf{C}}(a_t^{\mathsf{C}})^3, \quad C_t = 0 \tag{4}$$

where $\beta^{\mathsf{C}} = \frac{\kappa^{\mathsf{C}}\gamma^3}{\tau^2}$ and κ^{C} is the energy conversion coefficient of the collaborator. H_t^{lc} is the channel state between the IoT device and the collaborator. The first term in R.H.S of

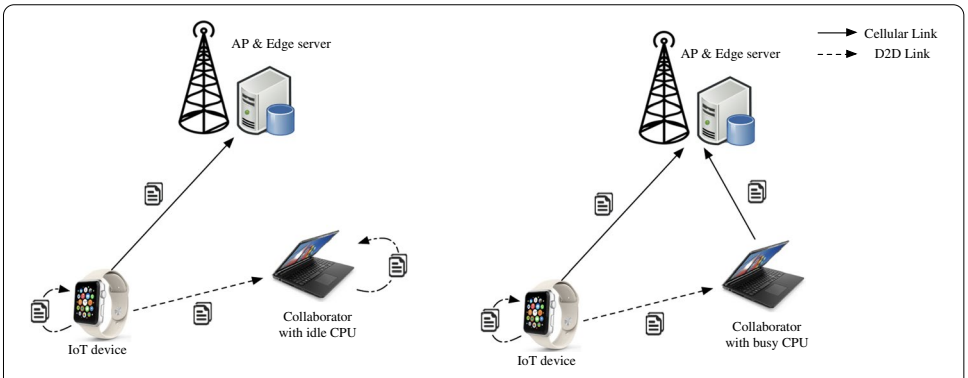

Fig. 1 System model

(4) denotes offloading energy consumption of the IoT device, and the second one represents the computation energy consumption of the collaborator.

When $C_t = 1$, the offloaded computation in this time slot cannot be processed by the collaborator. In this case, the collaborator further forwards the offloaded computation a_t^{C} to the edge cloud for processing. Therefore, the total transmission energy consumption of the IoT device and the collaborator at time slot t is expressed as

$$\mathcal{E}_t^{\mathsf{CO}}(a_t^{\mathsf{C}}) = \eta \frac{(a_t^{\mathsf{C}})^3}{H_t^{lc}} + \eta \frac{(a_t^{\mathsf{C}})^3}{H_t^{ce}}, \quad C_t = 1 \tag{5}$$

where H_t^{lc} is the channel state between the collaborator and the edge cloud server.

3.4 Problem formulation

Markov decision process (MDP) is a mathematical framework for modeling the decision-making problems in a stochastic system with multiple states and statistic system information, which suits our problem well. Furthermore, considering the hard completion time constraint of the application, we then formulate this opportunistic cooperative computation offloading as a finite-horizon MDP problem. The expected energy consumption of both the IoT device and the collaborator is minimized, subject to the hard deadline constraint of the application.

(1) State and action sets

The system state space of our MDP formulation is represented as $\mathcal{S} = \Phi_1 \times \Phi_2 \times \Phi_3 \times \mathcal{B} \times \mathcal{C}$, where channel states $H_t^{le} \in \Phi_1$, $H_t^{lc} \in \Phi_2$, $H_t^{ce} \in \Phi_3$ and collaborator CPU state $C_t \in \mathcal{C}$ evolve as Markov chains, and buffer state $B_t \in \mathcal{B}$. The action state space is denoted as $\mathcal{A}$, where $\boldsymbol{a}_t = (a_t^{\mathsf{L}}, a_t^{\mathsf{E}}, a_t^{\mathsf{C}}) \in \mathcal{A}$ denotes the action in time slot t. Due to the completion time constraint, the *feasible* actions depend on the current state. Given a specific system state $s_t \in \mathcal{S}$, the feasible actions in time slot t should satisfy the following conditions:

$$\begin{cases} a_t^{\mathsf{L}} \geq 0, a_t^{\mathsf{E}} \geq 0, a_t^{\mathsf{C}} \geq 0, a_t^{\mathsf{L}} + a_t^{\mathsf{E}} + a_t^{\mathsf{C}} \leq b_t, t < T, \\ a_t^{\mathsf{L}} \geq 0, a_t^{\mathsf{E}} \geq 0, a_t^{\mathsf{C}} \geq 0, a_t^{\mathsf{L}} + a_t^{\mathsf{E}} + a_t^{\mathsf{C}} = b_t, t = T. \end{cases} \tag{6}$$

2 State transition probability

Given the current state $s_t = \{h_t^{le}, h_t^{lc}, h_t^{ce}, b_t, c_t\}$, the state transition probability is the probability that the system will be in the next state $s_{t+1} = \{h_{t+1}^{le}, h_{t+1}^{lc}, h_{t+1}^{ce}, b_{t+1}, c_{t+1}\}$ after an action $\boldsymbol{a}_t$ is taken, denoted as

$$\begin{aligned} &\mathbb{P}(s_{t+1}|s_t, \boldsymbol{a}_t) \\ &\quad = \mathbb{P}(h_{t+1}^{le}|h_t^{le}) \times \mathbb{P}(h_{t+1}^{lc}|h_t^{lc}) \times \mathbb{P}(h_{t+1}^{ce}|h_t^{ce}) \\ &\qquad \times \mathbb{P}(c_{t+1}|c_t) \times \mathbb{I}(b_{t+1} = b_t - a_t^{\mathsf{L}} - a_t^{\mathsf{E}} - a_t^{\mathsf{C}}) \end{aligned} \tag{7}$$

where $\mathbb{I}(l)$ is an indicator function, which is equal to 1 if the condition (l) holds and 0 otherwise.

The total energy consumption of the IoT device and the collaborator in each time slot t, including the transmission energy consumption and the computation energy consumption, can be given by

$$\begin{aligned} J(s_t, \boldsymbol{a}_t) = &\mathcal{E}_t^{\mathsf{L}}(a_t^{\mathsf{L}}) + \mathcal{E}_t^{\mathsf{E}}(a_t^{\mathsf{E}}) \\ &+ (1 - c_t)\mathcal{E}_t^{\mathsf{CC}}(a_t^{\mathsf{C}}) + c_t\mathcal{E}_t^{\mathsf{CO}}(a_t^{\mathsf{C}}) \end{aligned} \tag{8}$$

An offloading policy consists of T decision rules for the T decision epochs: $t = \{1, 2, \ldots, T\}$, is defined below. A decision rule at time slot t maps states to actions and is denoted by $\boldsymbol{a}_t : \mathcal{S} \to \mathcal{A}$.

Definition 1 An *admissible* offloading policy is a function mapping the buffer state, the CPU state of the collaborator, the channel states and the time slot information into an action in each decision period:

$$\pi = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_T) : \{1, \ldots, T\} \times \mathcal{S} \to \mathcal{A} \tag{9}$$

The space of all admissible policies is denoted by Π.

Our objective is to minimize the expected total energy consumption of the IoT device and the collaborator given an initial state $s_1 \in \mathcal{S}$:

$$\min_{\pi \in \Pi} \mathbb{E}\left\{ \sum_{t=1}^{T} J(S_t, \boldsymbol{a}_t) | s_1 \right\} \tag{10}$$

where the expectation is taken with regard to the stochastic system state S_t for all t. In this paper, we consider deterministic Markov policies which is shown to be optimal under the expected total reward criteria [30]. The solving of the optimal policy is non-trivial since the decision on each time slot cannot be taken independently. The action in each time slot affects the system transition probability, the future states and energy consumption in subsequent time slots. The resultant computation offloading policy can be computed numerically by brute-force method or some reinforcement learning techniques, but on the one hand, brute-force method leads to an exponential increase on the complexity of policy solving with the number of state variables; on the other hand, reinforcement learning requires amount of time to training and learning a better strategy from trial and error. In the following section, we exploit the known model information to derive the closed-form policy, which provides some insight for the strategy design and efficiently alleviate the curse of dimensionality in stochastic optimization.

4 Optimal computation offloading policy

In this section, we derive the optimal computation offloading policy in each time slot based on the principle of optimality and dynamic programming (DP) [31]. The optimal closed-form solution greatly reduces the complexity of the problem solving.

Define $V_t(s_t)$ as the cost-to-go function that represents the minimum expected sum energy consumption from time slot t to T. Based on the principle of optimality, we have the following Bellman optimal equation:

$$V_t(s_t) = \begin{cases} \min\limits_{\boldsymbol{a}_t \in \pi(s_t)} J(s_t, \boldsymbol{a}_t) + \mathbb{E}[V_{t+1}(S_{t+1})|s_t, \boldsymbol{a}_t], & t < T, \\ \min\limits_{\boldsymbol{a}_t \in \pi(s_t)} J(s_t, \boldsymbol{a}_t), & t = T. \end{cases} \tag{11}$$

It can be seen the optimal cost-to-go function in each time slot depends on the functions in subsequent time slots. We first consider the policy in the last time slot T given the system state s_T. Then the problem for $V_T(s_T)$ can be written as follows.

Problem $\mathbb{P}_1$:

$$V_T(s_T) = \begin{cases} \min\limits_{\boldsymbol{a}_T \in \pi(s_T)} \beta^{\mathsf{L}} \left(a_T^{\mathsf{L}}\right)^3 + \eta \frac{\left(a_T^{\mathsf{E}}\right)^3}{h_T^{le}} \\ \qquad\qquad + \eta \frac{\left(a_T^{\mathsf{C}}\right)^3}{h_T^{lc}} + \beta^{\mathsf{C}} \left(a_T^{\mathsf{C}}\right)^3, c_T = 0, \\ \min\limits_{\boldsymbol{a}_T \in \pi(s_T)} \beta^{\mathsf{L}} \left(a_T^{\mathsf{L}}\right)^3 + \eta \frac{\left(a_T^{\mathsf{E}}\right)^3}{h_T^{le}} \\ \qquad\qquad + \eta \frac{\left(a_T^{\mathsf{C}}\right)^3}{h_T^{lc}} + \eta \frac{\left(a_T^{\mathsf{C}}\right)^3}{h_T^{ce}}, c_T = 1. \end{cases} \tag{12}$$

The action $\boldsymbol{a}_T$ should satisfy the following constraints.

$$\left\{ \boldsymbol{a}_T | a_T^{\mathsf{L}} \geq 0, a_T^{\mathsf{E}} \geq 0, a_T^{\mathsf{C}} \geq 0, a_T^{\mathsf{L}} + a_T^{\mathsf{E}} + a_T^{\mathsf{C}} = b_T \right\} \tag{13}$$

Through solving the constrained optimization problem, the optimal offloading decision can be given as follows.

Lemma 1 *At the last time slot T, the optimal amount of data for local processing, offloading to the edge cloud and offloading to the collaborator is, respectively, derived as*

$$\begin{cases} a_T^{\mathsf{L}\star} = b_T \left(1 + \sqrt{\frac{\beta^{\mathsf{L}} h_T^{le}}{\eta}} + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_T^{lc}} + (1 - c_T)\beta^{\mathsf{C}} + c_T \frac{\eta}{h_T^{ce}}}} \right)^{-1} \\ a_T^{\mathsf{C}\star} = \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_T^{lc}} + (1 - c_T)\beta^{\mathsf{C}} + c_T \frac{\eta}{h_T^{ce}}}} a_T^{\mathsf{L}\star} \\ a_T^{\mathsf{E}\star} = \sqrt{\frac{\beta^{\mathsf{L}} h_T^{le}}{\eta}} a_T^{\mathsf{L}\star} \end{cases} \tag{14}$$

The minimum energy consumption $V_T(s_T)$ is

$$V_T(s_T) = \beta^{\mathsf{L}} (b_T)^3 \left(1 + \sqrt{\frac{\beta^{\mathsf{L}} h_T^{le}}{\eta}} + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_T^{lc}} + (1 - c_T)\beta^{\mathsf{C}} + c_T \frac{\eta}{h_T^{ce}}}} \right)^{-2} \tag{15}$$

Proof It is obvious that Problem $\mathbb{P}_1$ is a convex optimization problem. When $c_T = 0$, define the Lagrangian function

$$\begin{aligned}\mathcal{L}(\boldsymbol{a}_T,\delta) = \beta^{\mathsf{L}}\left(a_T^{\mathsf{L}}\right)^3 + \eta\frac{\left(a_T^{\mathsf{E}}\right)^3}{h_T^{le}} + \frac{\left(a_T^{\mathsf{C}}\right)^3}{h_T^{lc}} \\ + \beta^{\mathsf{C}}\left(a_T^{\mathsf{C}}\right)^3 + \delta\left(a_T^{\mathsf{L}} + a_T^{\mathsf{E}} + a_T^{\mathsf{C}} - b_T\right)\end{aligned} \tag{16}$$

where δ is nonnegative Lagrangian multiplier. Applying the KKT condition, we have

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial a_T^{\mathsf{L}}} = 3\beta^{\mathsf{L}}\left(a_T^{\mathsf{L}}\right)^2 + \delta = 0 \\ \frac{\partial\mathcal{L}}{\partial a_T^{\mathsf{E}}} = 3\eta\frac{\left(a_T^{\mathsf{E}}\right)^2}{h_T^{le}} + \delta = 0 \\ \frac{\partial\mathcal{L}}{\partial a_T^{\mathsf{C}}} = 3\frac{\left(a_T^{\mathsf{C}}\right)^2}{h_T^{lc}} + 3\beta^{\mathsf{C}}\left(a_T^{\mathsf{C}}\right)^2 + \delta = 0 \\ \frac{\partial\mathcal{L}}{\partial\delta} = a_T^{\mathsf{L}} + a_T^{\mathsf{E}} + a_T^{\mathsf{C}} - b_T = 0 \end{cases} \tag{17}$$

The optimal solution can be obtained as

$$\begin{cases} a_T^{\mathsf{L}\star}(c_T=0) = \left(1 + \sqrt{\frac{\beta^{\mathsf{L}}h_T^{le}}{\eta}} + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_T^{lc}}+\beta^{\mathsf{C}}}}\right)^{-1} b_T \\ a_T^{\mathsf{C}\star}(c_T=0) = \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_T^{lc}}+\beta^{\mathsf{C}}}}\,a_T^{\mathsf{L}} \\ a_T^{\mathsf{E}\star}(c_T=0) = \sqrt{\frac{\beta^{\mathsf{L}}h_T^{le}}{\eta}}\,a_T^{\mathsf{L}} \end{cases} \tag{18}$$

Similarly, the optimal offloading solution can be also derived when $c_T = 1$. Combining the results of $c_T = 0$ and $c_T = 1$, the optimal policy at time slot T is thus obtained. □

Some important observations from Lemma 1 can be highlighted. Firstly, the offloading data size to edge cloud is closely related to two parameters, i.e., the local computation energy consumption per bit β^{L}, and the transmission energy consumption per bit η/h_T^{le}. The IoT device offloads more data to the edge cloud when the local computation per bit is energy consuming or the channel between the device and the edge cloud is in a good state. Similarly, apart from β^{L}, the offloading data size to the collaborator is also related to the sum energy consumption of transmission (from the IoT device to the collaborator) and computation (of the collaborator) $\eta/h_T^{lc} + \beta^{\mathsf{C}}$ when $c_T = 0$, the two-hop transmission energy consumption from the IoT device to the edge cloud $\eta/h_T^{lc} + \eta/h_T^{ce}$ when $c_T = 1$. The data portions of offloading and local processing are determined by these parameters. Besides, the energy consumption can be reduced by the proportional factor $\left(1 + \sqrt{\frac{\beta^{\mathsf{L}}h_T^{le}}{\eta}} + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_T^{lc}}+(1-c_T)\beta^{\mathsf{C}}+c_T\frac{\eta}{h_T^{ce}}}}\right)^{-2}$ through parallel computing. More energy saving is achieved with better wireless channel condition or higher collaborator's computation energy conversion efficiency.

Lemma 1 provides the optimal solution for different system states and the minimum network energy consumption at the last time slot. Based on this closed-form policy, the optimal computation offloading policy in each time slot can be derived through backward induction approach, shown as below.

Theorem 1 *At time slot $t = 1, 2, \ldots, T$, the optimal policy determines data $a_t^{\mathsf{L}\star}$ for local processing, $a_t^{\mathsf{E}\star}$ for offloading to the edge cloud and $a_t^{\mathsf{C}\star}$ for offloading to the collaborator, which satisfies:*

$$\begin{cases} a_t^{\mathsf{L}\star} = b_t \left(1 + \frac{1}{\sqrt{\Phi_t(s_t)}} + \sqrt{\frac{\beta^{\mathsf{L}} h_t^{le}}{\eta}} + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_t^{lc}} + (1-c_t)\beta^{\mathsf{C}} + c_t \frac{\eta}{h_t^{ce}}}} \right)^{-1} \\ a_t^{\mathsf{C}\star} = \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_t^{lc}} + (1-c_t)\beta^{\mathsf{C}} + c_t \frac{\eta}{h_t^{ce}}}} a_t^{\mathsf{L}\star} \\ a_t^{\mathsf{E}\star} = \sqrt{\frac{\beta^{\mathsf{L}} h_t^{le}}{\eta}} a_t^{\mathsf{L}\star} \end{cases} \tag{19}$$

where $\Phi_t(s_t)$ is defined as

$$\begin{aligned} &\Phi_t(s_t) \\ &= \begin{cases} \sum\limits_{h_{t+1}^{le}} \sum\limits_{h_{t+1}^{lc}} \sum\limits_{h_{t+1}^{ce}} \sum\limits_{c_{t+1}} \mathbb{P}(h_{t+1}^{le}|h_t^{le}) \mathbb{P}(h_{t+1}^{lc}|h_t^{lc}) \mathbb{P}(h_{t+1}^{ce}|h_t^{ce}) \\ \quad \times \mathbb{P}(c_{t+1}|c_t) \times \left(1 + \frac{1}{\sqrt{\Phi_{t+1}(s_{t+1})}} + \sqrt{\frac{\beta^{\mathsf{L}} h_{t+1}^{le}}{\eta}} \right. \\ \quad \left. + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_{t+1}^{lc}} + (1-c_{t+1})\beta^{\mathsf{C}} + c_{t+1} \frac{\eta}{h_{t+1}^{ce}}}} \right)^{-2}, & t < T \\ \infty, & t = T \end{cases} \end{aligned} \tag{20}$$

Correspondingly, the minimum expected energy consumption of the network $V_1(s_1)$ is

$$\begin{aligned} V_1(s_1) = \beta^{\mathsf{L}} W^3 \Bigg(& 1 + \frac{1}{\sqrt{\Phi_1(s_1)}} + \sqrt{\frac{\beta^{\mathsf{L}} h_1^{le}}{\eta}} \\ & + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_1^{lc}} + (1-c_1)\beta^{\mathsf{C}} + c_1 \frac{\eta}{h_1^{ce}}}} \Bigg)^{-2} \end{aligned} \tag{21}$$

Proof Firstly, we derive the optimal action and expected energy consumption from time slot $T-1$ to T for both $c_{T-1} = 0$ and $c_{T-1} = 1$. When $c_{T-1} = 0$, the optimization problem for $V_{T-1}(s_{T-1})$ can be given as

$$\begin{aligned} &V_{T-1}(s_{T-1}) \\ &\quad = \min_{\boldsymbol{a}_{T-1}} \{ J(s_{T-1}, \boldsymbol{a}_{T-1}) + \mathbb{E}[V_T(S_T)|s_{T-1}, \boldsymbol{a}_{T-1}] \} \\ &\quad = \min_{\boldsymbol{a}_{T-1}} \Bigg\{ J(s_{T-1}, \boldsymbol{a}_{T-1}) + \sum_{h_T^{le}} \sum_{h_T^{lc}} \sum_{h_T^{ce}} \sum_{c_T} \mathbb{P}(h_T^{le}|h_{T-1}^{le}) \\ &\quad \mathbb{P}(h_T^{lc}|h_{T-1}^{lc}) \mathbb{P}(h_T^{ce}|h_{T-1}^{ce}) \mathbb{P}(c_T|c_{T-1}) V_T(s_T) \Bigg\} \end{aligned} \tag{22}$$

Based on $V_T(s_T)$ in Lemma 1 and define

$$\begin{aligned}
&\Phi_{T-1}(s_{T-1}) \\
&= \sum_{h_T^{le}} \sum_{h_T^{lc}} \sum_{h_T^{ce}} \sum_{c_T} \mathbb{P}(h_T^{le}|h_{T-1}^{le}) \mathbb{P}(h_T^{lc}|h_{T-1}^{lc}) \mathbb{P}(h_T^{ce}|h_{T-1}^{ce}) \\
&\quad \times \mathbb{P}(c_T|c_{T-1}) \left(1 + \sqrt{\frac{\beta^{\mathsf{L}} h_T^{le}}{\eta}} + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_T^{lc}} + \beta^{\mathsf{C}}}}\right)^{-2}
\end{aligned} \tag{23}$$

$V_{T-1}(s_{T-1})$ can be further written as

$$\begin{aligned}
V_{T-1}(s_{T-1}) &= \min_{\boldsymbol{a}_{T-1}} \Big\{ \beta^{\mathsf{L}} (a_{T-1}^{\mathsf{L}})^3 \\
&+ \eta \frac{(a_{T-1}^{\mathsf{E}})^3}{h_{T-1}^{le}} + \eta \frac{(a_{T-1}^{\mathsf{C}})^3}{h_{T-1}^{lc}} + \beta^{\mathsf{C}} (a_{T-1}^{\mathsf{C}})^3 \\
&+ \beta^{\mathsf{L}} (b_{T-1} - a_{T-1}^{\mathsf{L}} - a_{T-1}^{\mathsf{C}} - a_{T-1}^{\mathsf{E}})^3 \Phi_{T-1}(s_{T-1}) \Big\}
\end{aligned} \tag{24}$$

where $\boldsymbol{a}_{T-1}$ satisfies the constraint

$$\begin{aligned}
&\Big\{ \boldsymbol{a}_{T-1} | a_{T-1}^{\mathsf{L}} \geq 0, a_{T-1}^{\mathsf{E}} \geq 0, a_{T-1}^{\mathsf{C}} \geq 0, \\
&\quad a_{T-1}^{\mathsf{L}} + a_{T-1}^{\mathsf{E}} + a_{T-1}^{\mathsf{C}} \leq b_{T-1} \Big\}
\end{aligned} \tag{25}$$

Applying the KKT condition, the optimal policy when $c_{T-1} = 0$ can be obtained as

$$\begin{cases}
a_{T-1}^{\mathsf{L}\star}(c_{T-1} = 0) = b_{T-1} \left(1 + \frac{1}{\sqrt{\Phi_{T-1}(s_{T-1})}} + \sqrt{\frac{\beta^{\mathsf{L}} h_{T-1}^{le}}{\eta}} + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_{T-1}^{lc}} + \beta^{\mathsf{C}}}}\right)^{-1} \\
a_{T-1}^{\mathsf{C}\star}(c_{T-1} = 0) = \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_{T-1}^{lc}} + \beta^{\mathsf{C}}}} a_{T-1}^{\mathsf{L}\star} \\
a_{T-1}^{\mathsf{E}\star}(c_{T-1} = 0) = \sqrt{\frac{\beta^{\mathsf{L}} h_{T-1}^{le}}{\eta}} a_{T-1}^{\mathsf{L}\star}
\end{cases} \tag{26}$$

Similarly, the optimal policy when $c_{T-1} = 1$ can be derived. Combining the results of both cases, the optimal offloading policy and minimum energy consumption at time slot $T-1$ are, respectively

$$\begin{cases}
a_{T-1}^{\mathsf{L}\star} = b_{T-1} \left(1 + \frac{1}{\sqrt{\Phi_{T-1}(s_{T-1})}} + \sqrt{\frac{\beta^{\mathsf{L}} h_{T-1}^{le}}{\eta}} + \right. \\
\qquad \left. \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_{T-1}^{lc}} + (1 - c_{T-1})\beta^{\mathsf{C}} + c_{T-1} \frac{\eta}{h_{T-1}^{ce}}}} \right)^{-1} \\
a_{T-1}^{\mathsf{C}\star} = \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_{T-1}^{lc}} + (1 - c_{T-1})\beta^{\mathsf{C}} + c_{T-1} \frac{\eta}{h_{T-1}^{ce}}}} a_{T-1}^{\mathsf{L}\star} \\
a_{T-1}^{\mathsf{E}\star} = \sqrt{\frac{\beta^{\mathsf{L}} h_{T-1}^{le}}{\eta}} a_{T-1}^{\mathsf{L}\star}
\end{cases} \tag{27}$$

$$
\begin{aligned}
&V_{T-1}(s_{T-1}) \\
&= \beta^{\mathsf{L}}(b_{T-1})^3 \left(1 + \frac{1}{\sqrt{\Phi_{T-1}(s_{T-1})}} + \sqrt{\frac{\beta^{\mathsf{L}} h_{T-1}^{le}}{\eta}} \right. \\
&\left. + \sqrt{\frac{\beta^{\mathsf{L}}}{\frac{\eta}{h_{T-1}^{lc}} + (1-c_{T-1})\beta^{\mathsf{C}} + c_{T-1}\frac{\eta}{h_{T-1}^{ce}}}}\right)^{-2}
\end{aligned} \tag{28}
$$

Comparing the optimal solution (14) for $t = T$ with (27) for $t = T - 1$, similar structure can be found. Therefore, utilizing the backward induction and similar derivation process, the optimal computation offloading policy can be given as Theorem 1. □

Theorem 1 demonstrates that in each time slot, the computation offloading policy depends not only on the current system state, but also on the future states through the term $\Phi_t(s_t)$. In detail, a larger $\Phi_t(s_t)$ means more expected energy consumption for completing per bit data in future time slots. Therefore, more data should be processed in the current time slot no matter for local computing or offloading. Moreover, the same conclusion as Lemma 1 can be obtained. That is, the fraction of the optimal data size for locally computing, offloading to the edge cloud and offloading to the collaborator in each time slot is determined by the local computation energy consumption per bit β^{L}, and the offloading energy consumption per bit η/h_T^{le}, $\eta/h_T^{lc} + \beta^{\mathsf{C}}$ or $\eta/h_T^{lc} + \eta/h_T^{ce}$. Given the sum-processed data in one time slot, the preferred destination (local device, edge cloud or collaborator) at which more data are processed relies on the energy consumption for per-bit data.

Algorithm 1 Optimal Computation Offloading Policy

1: Initialize: set $t = T$ and $\Phi_t(s_t) = \infty$ for all $s_t \in \mathcal{S}$.
2: **repeat**
3: $t = t - 1$
4: Compute $\Phi_t(s_t)$ using (20) for each $s_t \in \mathcal{S}$
5: Compute the optimal policy using (19) for each $s_t \in \mathcal{S}$
6: **until** $t = 1$
7: Compute the minimum expected energy consumption $V_1(s_1)$ using (21)

Based on the Theorem 1, Algorithm 1 shows the optimal closed-form computation offloading policy. We then analyze the computational complexity of the algorithm. Based on Theorem 1, the optimal policy is determined by the known original system state s_1 and unknown $\Phi_1(s_1)$, where the complexity for calculating $\Phi_1(s_1)$ depends on the three channel state spaces, the dimensions of collaborator CPU state and the number of time slots. Denote the number of the channel state as N, the computational complexity of the algorithm is thus $\mathcal{O}(2N^3T)$. For the direct DP approach that searches all actions in policy space and in each time slot, the dimensions of state space, action space are $\mathcal{O}(2N^3)$ and $\mathcal{O}(W^3)$ in time slot t, respectively, leading to the total computational complexity $|\mathcal{S}|^2\mathcal{A}T = \mathcal{O}(4N^6W^3T)$, which is impractical for large data size W and time slots T. Compare our closed-form policy to the brute-force policy search, the computational complexity is reduced dramatically.

Note that the entire process including the offloading decision and application execution is controlled by the AP. Once an application is generated, the IoT device reports its data size S, application deadline T and related parameters of local computing via feedback channel. Based on the historical statistical information on the channel state and the CPU state of the collaborator, the AP executes Algorithm 1 to get the amounts of data for local computing (a_t^{L}) and offloading (a_t^{E} and a_t^{C}) in each time slot. The policy is informed of the IoT device by the bidirectional feedback channel and the corresponding computation offloading process starts.

In this work, we consider the hard deadline requirement of an IoT application, which is usually an explicit parameter upon it is generated. This is common in most of the existing literature, see [10–13]. However, if the deadline parameter is hard to obtain in a fine-grained way, the statistical value of the deadline for a kind of IoT application should be got from the historical information. In this case, there are two options:

(1) A guard time can be set to cope with this uncertainty. That is, the deadline is set as the minimum historical value less a guard time. Our scheme can be still adopted in this case.
(2) The application should be completed before a probabilistic deadline. This make the original problem more complicated since the constraint is a probabilistic form. The problem will be considered in our future work.

5 Results and discussion

In this section we simulate the performance of proposed Algorithm 1. Default parameters are set as follows unless stated otherwise. The time slot duration is 20 ms [32]. We consider an IoT device has an application with 400 kilobits under the deadline constraint of 140 ms, i.e., $T = 7$ time slots. The delay budget T is the actual delay budget of the application less the time needed to run Algorithm 1. Energy conversion coefficients $\kappa^{\mathsf{L}} = 10^{-27}$, $\kappa^{\mathsf{C}} = 0.3 * 10^{-27}$ [18] and the required CPU cycles for computing 1-bit data are $\gamma = 1000$ cycle/bit [33]. For large-scale fading, the distances for IoT device-edge cloud, IoT device-collaborator and collaborator-edge cloud are 300 m, 50 m and 260 m. Path-loss exponent is 2.4. Small-scale fading for the channel is modeled as a two-state Markov chain, i.e., if the measured channel gain is below a threshold, the channel is considered as "bad"; otherwise, the "good" condition. The average channel power gains are 1 and 0.01 for the "good" and "bad" states, respectively [15]. The good-to-bad and bad-to-good transition probabilities are, respectively, set as $P_{gb} = 3/7$ and $P_{bg} = 3/10$ [28]. The CPU state of the collaborator follows another Markov chain with $P_{bb} = 0.7$ and $P_{ii} = 0.8$ [13] and the energy coefficient $\lambda = 10^{-21}$. To validate the optimality of the derived policy, optimal numerical results solved by brute-force policy search are presented. Then three algorithms are considered for performance comparison.

(1) Local or edge server execution (LESE) [15]: The computation load is entirely processed locally by IoT device or offloaded to the edge cloud through stochastic channel. The energy consumption of IoT device is minimized.

(2) Local and dynamic collaborator execution (LDCE) [13]: Part of the computation load is processed locally by IoT device, and the other is offloaded to a collaborator with dynamic computation resources through stochastic channel concurrently. The energy consumption of IoT device is minimized.
(3) Equal allocation in each slot (EA): The computation load is processed locally by IoT device, offloaded to the edge cloud and offloaded to a collaborator with dynamic computation resources in parallel. The input data W are allocated into T time slots equally, regardless of channel conditions and the collaborator CPU state. Then the offloading policy in each slot $\boldsymbol{a}_T = (a_T^{\mathsf{L}}, a_T^{\mathsf{E}}, a_T^{\mathsf{C}})$ is obtained with similar derivation procedure as the proposed algorithm. The energy consumption of both the IoT device and the collaborator is minimized.

Figure 2 shows the results of the total expected minimum energy consumption versus data size for the derived closed-form policy and brute-force policy search under different deadline constraints. It is noticed that the results of the two approaches are very close, which demonstrates the optimality of our computation offloading policy. Meanwhile, as the data size of application increases, the expected energy consumption grows at an increasing rate. The reason is that more energy, both computation energy and transmission energy, is consumed for completing more computation within the given deadline constraint. More stringent deadline constraint also leads to more expected energy consumption.

The total expected energy consumption versus the application data size for our approach and the three benchmark schemes are depicted in Fig. 3. With the increase of the application data size, the total expected energy consumption grows and the increasing rate gets larger in all four schemes, which is consistent with Fig. 2. It also can be observed that the total expected energy consumption of the proposed scheme is less than the EA scheme. This is because the proposed scheme considers the

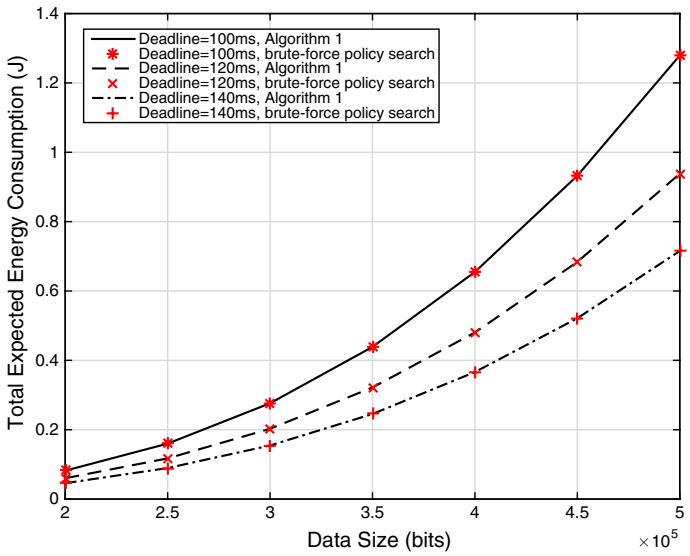

Fig. 2 Total expected energy consumption versus data size

channel and CPU states when offloading. More (less) data are transmitted when channel state is good (bad), which saves the transmission energy consumption. The proposed scheme also outperforms LDCE in [13]. The reason is that the presence of edge server increases the offloading opportunity for the IoT device to reduce its energy consumption. More wireless channels and computation resources promote the IoT device to transmit more application-related data on better channels, which achieves diversity gain. Moreover, the collaborator and edge server can process the data in parallel, which reduces the application completion time substantially. Similarly, the performance of the proposed scheme is better than LESE in [15] due to the diversity gain. Overall, this demonstrates the efficiency of the proposed cooperative computing.

Figure 4 depicts the total expected energy consumption versus the application deadlines. As the deadline extends, the total expected energy consumption reduces at an decreasing rate in the four schemes. On the one hand, local processing rate can be lowered with extended time and thus the local computation energy consumption further reduces. On the other hand, the computation can be opportunistically offloaded at good channel condition or favorable collaborator CPU state in more time slots, leading to the reduction of the transmission energy consumption. This also demonstrates that extending deadline contributes more to the energy saving when deadline constraint is stringent. When the deadline is relaxed, the constraint is inactive and has less impacts on the expected energy consumption.

We then evaluate the effect of the collaborator energy conversion coefficient κ^{C} on the total expected energy consumption in Fig. 5. As κ^{C} increases, the energy consumption correspondingly increases for the three collaborator-related schemes, since the per-bit computation energy consumption for the collaborator grows. Only with available computation resources of the local device and the collaborator, the energy consumption of the LDCE scheme grows rapidly. The increase of the energy

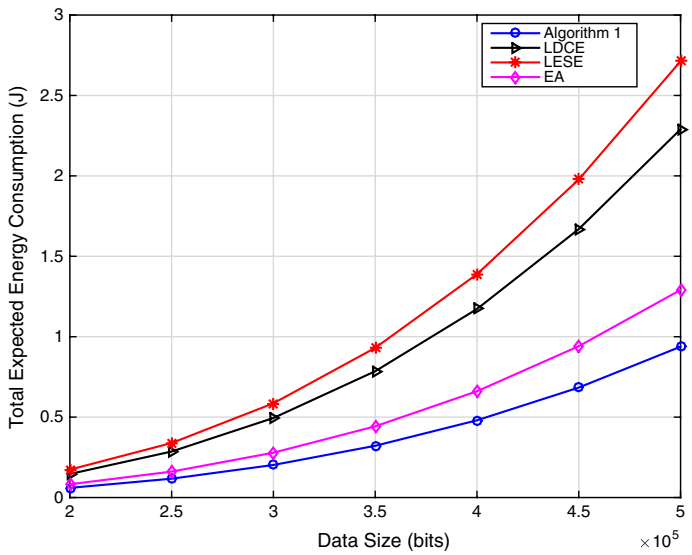

Fig. 3 Total expected energy consumption versus data size

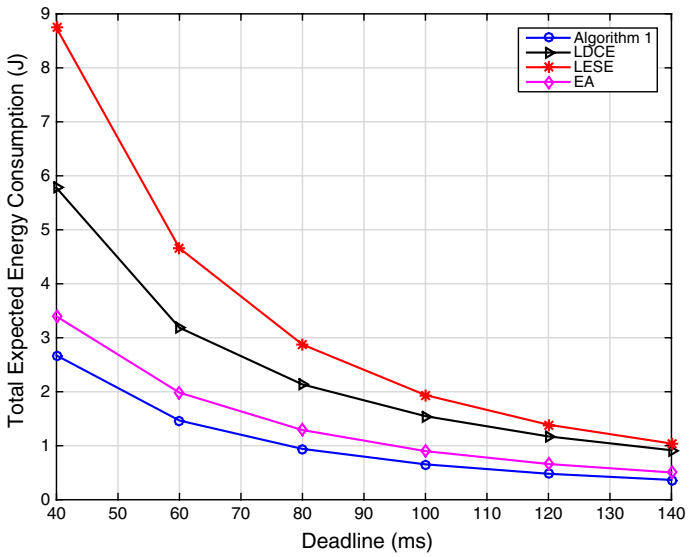

Fig. 4 Total expected energy consumption versus application deadline

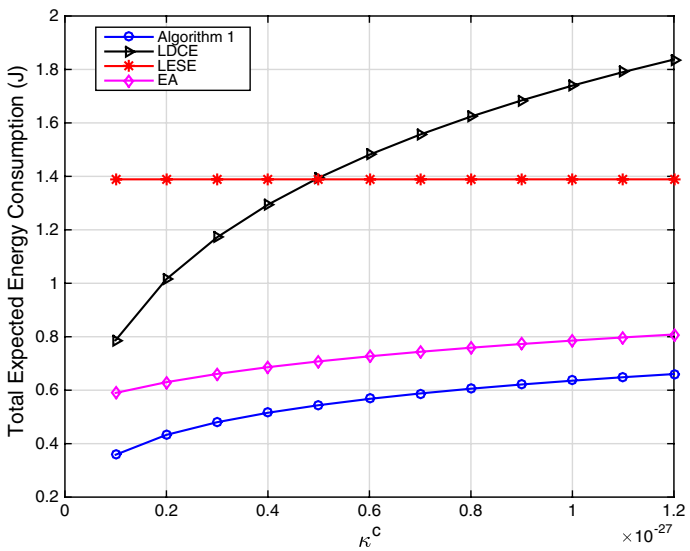

Fig. 5 Total expected energy consumption versus κ^C

consumption of the other two schemes, i.e., the proposed and EA scheme, is alleviated because more computation can be offloaded to the edge cloud.

The total expected energy consumption versus the CPU cycles per bit γ is shown in Fig. 6. The energy consumption grows as γ increases in all the schemes, but at different increasing rates. For the LDCE scheme, the changes of the slope are the largest. This is because the computation is processed only at the local IoT device and the collaborator, and both of the computation energies are related to γ. For the LESE scheme, local computation energy grows with the increases of γ, and more computation is offloaded to the edge cloud. As a result, the expected energy consumption grows with a decreasing rate. The increasing rate for the proposed scheme is similar to the LESE scheme rather

than the LDCE scheme, which shows offloading to the edge cloud is more preferred as γ increases. But the performance of our scheme is still better than the LDCE and LESE schemes since more computing resources can be utilized. Moreover, the advantage of the proposed approach is more obvious with larger γ. Thus, the performance gap between the proposed approach and the equal allocation gets larger.

Figure 7 shows the total expected energy consumption versus the distance between the IoT device and the collaborator. It can be noticed that the energy consumption grows as the collaborator gets more far away from the device, since more transmission energy consumption incurred. Besides, the energy consumption of the proposed scheme is close to the LDCE scheme when the distance is short, but the performance gap gets larger as the distance extends. This also demonstrates the effectiveness of the proximate computing.

Table 1 shows the execution time of Algorithm 1, brute-force policy and the other three comparison algorithms. These algorithms are deployed at the computer with a dual-core Intel CPU at 2.9 GHz frequency. The time is collected by 1000 average with the data sizes uniformly distributed in [200, 500] kilobits. It can be noticed that the execution time of Algorithm 1 is much lower than brute-force policy, thanks to the derived closed-form optimal policy. The execution time of Algorithm 1 is larger than the other three comparison algorithms, due to the trade-off between the performance gain and the computational complexity. It should be highlighted that the absolute time is just a reference because it is highly dependent on the machine running the algorithms.

6 Conclusions

In this work, we study the cooperative computing between IoT devices, collaborators with dynamic idle computation resources and dedicated edge cloud. Specifically, an IoT device can compute locally, offload computation load to a collaborator and edge cloud in parallel. The collaborator assists in computing at idle states and in

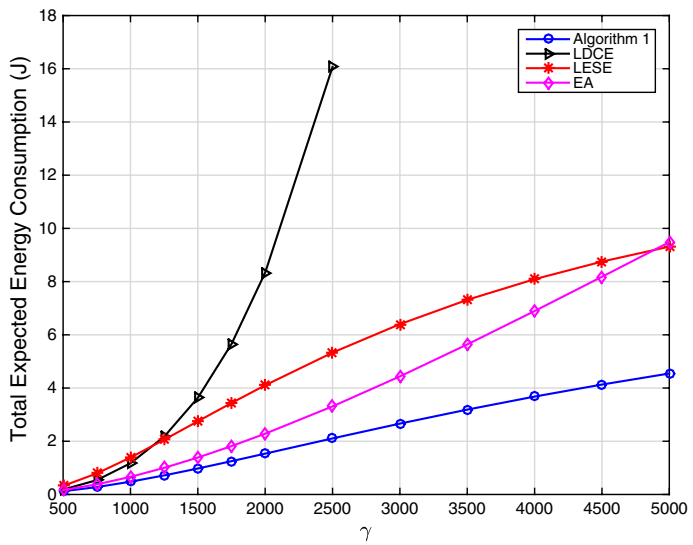

Fig. 6 Total expected energy consumption versus γ

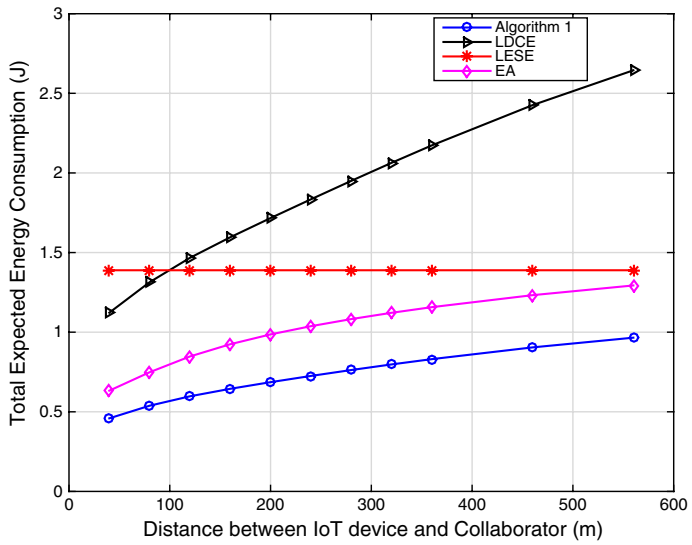

Fig. 7 Total expected energy consumption versus distance between IoT device and collaborator

Table 1 Execution time of different algorithms

Algorithm	Execution time (s)
Brute-force policy search	20.235
Algorithm 1	2.137
LESE	0.114
LDCE	0.406
EA	1.837

further offloading to the edge cloud at busy states. The problem on how much computation load is executed locally, offloaded to edge cloud and a collaborator, is modeled as a finite horizon Markov decision problem with the objective of minimizing the expected total energy consumption of the IoT device and the collaborator, subject to satisfying the hard completion time constraint. Optimal offloading policy is derived based on the stochastic optimization theory, which alleviates the well-known curse of dimensionality and facilitates the design of a low-complexity dynamic programming algorithm. Simulation results validate the optimality of the proposed policy and show that more energy saving is achieved with better wireless channel condition or higher computation energy efficiency of collaborators.

In the future, we focus on extending this work to more general scenarios, where a large number of IoT devices exist. Resources competition (on communication and computation resources) or collaborator selection problem need to be addressed efficiently. Moreover, based on the useful insights in this work, devising online learning algorithm is also an interesting direction, which may require less model information but more time (for training) to get a satisfactory policy. Lastly, although we consider the energy consumption of the peer collaborators in this work, incentive mechanisms

can still be designed for effectively encouraging nearby peer devices to share their idle communication and computation resources and achieve win–win situation.

Abbreviations
IoT: Internet of Things; MEC: Mobile edge computing; MECO: Mobile edge computation offloading; D2D: Device to device; CPU: Central processing unit; KKT: Karush–Kuhn–Tucker; TDMA: Time-division multiple access; FDMA: Frequency-division multiple access; AP: Access point; MDP: Markov decision process; DVFS: Dynamic voltage and frequency scaling; FSMC: Finite state Markov chain; DP: Dynamic programming; LTE: Long-term evolution.

Acknowledgements
The authors would like to thank the editors and anonymous reviewers who have contributed to the enhancement of the paper's completeness with their valuable suggestions. We also thank Prof. Miao Hu for his valuable suggestions.

Authors' contributions
SM contributed to the system modeling, algorithm design, performance analysis and simulations. ZZ reviewed and commented on the manuscript. Both authors read and approved the manuscript.

Funding
Not applicable.

Availability of data and materials
The details of experimental parameters are given in Sect. 5.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Haidian District, 100044 Beijing, People's Republic of China. [2] Department of Electronic Engineering, Beijing Jiaotong University, Haidian District, Beijing, People's Republic of China.

Received: 3 May 2020 Accepted: 16 November 2020
Published online: 02 December 2020

References

1. X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, Y. Zhang, Selective offloading in mobile edge computing for the green internet of things. IEEE Netw. **32**(1), 54–60 (2018)
2. Y. Wang, M. Sheng, X. Wang, L. Wang, J. Li, Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans. Commun. **64**(10), 4268–4282 (2016)
3. C. You, K. Huang, H. Chae, B.-H. Kim, Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **16**(3), 1397–1411 (2017)
4. X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. **24**(5), 2795–2808 (2016)
5. Z. Tan, F.R. Yu, X. Li, H. Ji, V.C. Leung, Virtual resource allocation for heterogeneous services in full duplex-enabled SCNS with mobile edge computing and caching. IEEE Trans. Veh. Technol. **67**(2), 1794–1808 (2018)
6. Mu, S., Zhong, Z., Zhao, D., Ni, M.: Latency constrained partial offloading and subcarrier allocations in small cell networks, in *Proceedings of IEEE ICC*, pp. 1–6 (2019)
7. Y. Geng, G. Cao, Peer-assisted computation offloading in wireless networks. IEEE Trans. Wirel. Commun. **17**(7), 4565–4578 (2018)
8. H. Xing, L. Liu, J. Xu, A. Nallanathan, Joint task assignment and resource allocation for D2D-enabled mobile-edge computing. IEEE Trans. Commun. **67**(6), 4193–4207 (2019)
9. S. Mu, Z. Zhong, D. Zhao, M. Ni, Joint job partitioning and collaborative computation offloading for Internet of Things. IEEE Internet Things J. **6**(1), 1046–1059 (2019)
10. Y. He, J. Ren, G. Yu, Y. Cai, D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. IEEE Trans. Wirel. Commun. **18**(3), 1750–1763 (2019)
11. G. Hu, Y. Jia, Z. Chen, Multi-user computation offloading with D2D for mobile edge computing, in *Proceedings of IEEE GLOBECOM*, pp. 1–6 (2018)
12. C. You, K. Huang, Exploiting non-causal CPU-state information for energy-efficient mobile cooperative computing. IEEE Trans. Wirel. Commun. **17**(6), 4104–4117 (2018)
13. Y. Tao, C. You, P. Zhang, K. Huang, Stochastic control of computation offloading to a helper with a dynamically loaded CPU. IEEE Trans. Wirel. Commun. **18**(2), 1247–1262 (2019)
14. A.S. Prasad, M. Arumaithurai, D. Koll, X. Fu, Raera: a robust auctioning approach for edge resource allocation, in *Proceedings of the Workshop on Mobile Edge Communications*, pp. 49–54 (2017)
15. W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, D.O. Wu, Energy-optimal mobile cloud computing under stochastic wireless channel. IEEE Trans. Wirel. Commun. **12**(9), 4569–4581 (2013)
16. W. Zhang, Y. Wen, D.O. Wu, Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Trans. Wirel. Commun. **14**(1), 81–93 (2014)

17. M. Hu, D. Wu, W. Wu, J. Cheng, M. Chen, Quantifying the influence of intermittent connectivity on mobile edge computing. IEEE Trans. Cloud Comput. (2019). https://doi.org/10.1109/TCC.2019.2926702
18. X. Cao, F. Wang, J. Xu, R. Zhang, S. Cui, Joint computation and communication cooperation for energy-efficient mobile edge computing. IEEE Internet Things J. **6**, 4188–4200 (2018)
19. N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, X. Shen, Space/aerial-assisted computing offloading for IoT applications: a learning-based approach. IEEE J. Sel. Areas Commun. **37**(5), 1117–1129 (2019)
20. X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, M. Bennis, Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet Things J. **6**(3), 4005–4018 (2018)
21. S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, K.K. Leung, Dynamic service migration in mobile edge computing based on Markov decision process. IEEE/ACM Trans. Netw. **27**(3), 1272–1288 (2019)
22. A.P. Miettinen, J.K. Nurminen, Energy efficiency of mobile clients in cloud computing, in *Proceedings USENIX HotCloud*, pp. 4–11 (2010)
23. O. Munoz, A. Pascual-Iserte, J. Vidal, Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. IEEE Trans. Veh. Technol. **64**(10), 4738–4755 (2015)
24. B. Prabhakar, E.U. Biyikoglu, A. El Gamal, Energy-efficient transmission over a wireless link via lazy packet scheduling, in *Proceedings IEEE INFOCOM 2001*, vol. 1, pp. 386–394 (2001)
25. A.P. Chandrakasan, S. Sheng, R.W. Brodersen, Low-power CMOS digital design. IEICE Trans. Electron. **75**(4), 371–382 (1992)
26. H.S. Wang, N. Moayeri, Finite-state Markov channel—a useful model for radio communication channels. IEEE Trans. Veh. Technol. **44**(1), 163–171 (1995)
27. M.J. Neely, E. Modiano, C.E. Rohrs, Dynamic power allocation and routing for time varying wireless networks. IEEE J. Sel. Areas Commun. **23**(1), 89–103 (2005)
28. M. Zafer, E. Modiano, Minimum energy transmission over a wireless channel with deadline and power constraints. IEEE Trans. Autom. Control **54**(12), 2841–2852 (2009)
29. T. He, S. Chen, H. Kim, L. Tong, K.-W. Lee, Scheduling parallel tasks onto opportunistically available cloud resources, in *2012 IEEE Fifth International Conference on Cloud Computing*, pp. 180–187 (2012)
30. M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (Wiley, Hoboken, 2014)
31. D.P. Bertsekas, D.P. Bertsekas, D.P. Bertsekas, D.P. Bertsekas, *Dynamic Programming and Optimal Control* (Athena Scientific, Belmont, 1995)
32. D.V. Djonin, V. Krishnamurthy, Q-learning algorithms for constrained Markov decision processes with randomized monotone policies: application to mimo transmission control. IEEE Trans. Signal Process. **55**(5), 2170–2181 (2007)
33. X. Huang, K. Xu, C. Lai, C. Qianbin, Z. Jie, Energy-efficient offloading decision-making for mobile edge computing in vehicular networks. EURASIP J. Wirel. Commun. Netw. **35**, 1–16 (2020)

Publisher's Note