EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Intelligent resource allocation scheme for cloud-edge-end framework aided multi-source data stream

Yuxin Wu[1], Changjun Cai[2*], Xuanming Bi[3], Junjuan Xia[1*], Chongzhi Gao[1*], Yajuan Tang[1] and Shiwei Lai[1]

*Correspondence:
ChangjunCai11@hotmail.com;
xiajunjuan@gzhu.edu.cn;
czgao@gzhu.edu.cn

[1] School of Computer Science, Guangzhou University, Guangzhou, China
[2] Guangzhou Metro Group Co., Ltd, Guangzhou, China
[3] School of Mathematics and Information, Software, South China Agricultural University, Guangzhou, China

Abstract

To support multi-source data stream generated from Internet of Things devices, edge computing emerges as a promising computing pattern with low latency and high bandwidth compared to cloud computing. To enhance the performance of edge computing within limited communication and computation resources, we study a cloud-edge-end computing architecture, where one cloud server and multiple computational access points can collaboratively process the compute-intensive data streams that come from multiple sources. Moreover, a multi-source environment is considered, in which the wireless channel and the characteristic of the data stream are time-varying. To adapt to the dynamic network environment, we first formulate the optimization problem as a markov decision process and then decompose it into a data stream offloading ratio assignment sub-problem and a resource allocation sub-problem. Meanwhile, in order to reduce the action space, we further design a novel approach that combines the proximal policy optimization (PPO) scheme with convex optimization, where the PPO is used for the data stream offloading assignment, while the convex optimization is employed for the resource allocation. The simulated outcomes in this work can help the development of the application of the multi-source data stream.

Keywords: Multi-source data stream, Edge computing, Collaborative offloading, Proximal policy optimization

1 Introduction

Owing to the rapid advancement and innovation of wireless communication in the 5th generation (5G), an increasing number of smart devices are linked to the Internet through wireless communication, which facilitates the birth and development of the Internet of Things (IoT). In IoT networks, one typical application is how to process multi-source data streams generated from IoT devices [1–3]. In particular, the characteristic of the data stream is high-dimensional, heterogeneous, and compute-intensive, which leads to a considerable cost for processing at the devices [4–6]. To solve this problem, mobile cloud computing (MCC) is devised as a new computing pattern by uploading data streams to a more powerful cloud server for computing. Based

Journal : **BMCOne 13634**	Dispatch : **14-5-2023**	Pages : **20**
Article No : **1018**	☐ LE	☐ TYPESET
MS Code :	☑ CP	☑ DISK

on MCC, energy consumption can be significantly reduced at the devices. However, there exists an unbearable latency in the MCC networks. As the distance is often far from the devices to the cloud server, the transmission latency becomes a bottleneck [7–9].

In order to handle the issue caused by cloud computing, mobile edge computing (MEC) emerges as a promising computing pattern with the advantage of low latency and high bandwidth [10–12]. By setting the edge server closer to the devices, the IoT devices can upload more computational tasks to the edge server in order to obtain ultra-low latency. For the MEC networks, one significant part is to make offloading ratio assignments [13, 14]. Literature [15, 16] studied a MEC network with task dependency and proposed some static numerical solutions to reduce the delay and energy overhead. In reality, the network environment was dynamic, where the wireless channel was time-varying, and the characteristic of the data stream was variable [17, 18]. For this case, some dynamic offloading methods were devised based on deep reinforcement learning (DRL) [19, 20]. The researchers in [21] proposed a Q-learning based binary offloading strategy to reduce the task execution time. Moreover, to make offloading strategy more flexible, a partial task unpacking decision was proposed based on the Deep Q network (DQN) [20]. In further, with the massive increase in the number of devices and the limited computing resources of CAP, joint optimization of resource allocation and offloading strategy were widely studied for the MEC network [22]. Literature [23] applied the Lyapunov optimization for resource allocation and offloading strategy while ensuring the maximization of long-term quality of experience (QoE). The authors in [24] proposed a low-complexity algorithm for the real-time MEC system. Furthermore, physical layer security was taken into consideration to ensure a secure transmission rate, meanwhile decreasing the system delay for the MEC network [20].

In addition, the performance of the MEC network has been widely investigated. Literature [9, 25] studied an intelligent reflect surface (IRS)-aided MEC network and derived the closed-form of outage probability of system latency. The researchers in [26] evaluated and optimized the performance of the cache-aided relaying MEC network. Moreover, a hybrid spectrum access technology was studied to improve the performance of the non-orthogonal multiple access (NOMA)-based network [27]. Furthermore, literature [28] considered a realistic scenario that the perfect estimation was tough to obtain and devised a dynamic resource allocation to maximize the energy efficiency for the NOMA-based MEC network. Although edge computing can effectively relieve the burden on the core network compared to cloud computing, its constrained computing and communication resources became the barrier to development [29]. Thereby, collaborative computing between the cloud server and edge server can further enhance the performance of the MEC network [30].

However, the works listed above mainly focus on the resource allocation as well as offloading strategy to improve the performance of the MEC network, which fails to consider a charge service mechanism. If users are allocated more computational resources, they need to pay more. Meanwhile, each user should have a budget constraint that decides how many resources can be purchased. Therefore, the service mechanism may influence the performance of the MEC system. As far as we know, few works consider the charge service in the collaborative computing network. Motivated by this, a charge

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

service mechanism is incorporated for the cloud-edge-end computing network. The main contributions of this work are listed as follows:

- To improve the performance of MEC-assisted multi-source data stream computing, we study a cloud-edge-end computing architecture in the case of a dynamic environment, where the wireless channel is time-varying, and the data stream characteristic is variable. Moreover, the charge service mechanism is involved in the considered network.
- In order to guarantee the effectiveness of the considered computing framework, we formulate an optimization for minimizing system latency by optimizing the offloading strategy assignment, the computation resource allocation, and the bandwidth resource allocation under the device's budget jointly.
- We design a novel approach that combines DRL with convex optimization, named "ECC-PPO." Specifically, the DRL is used for offloading strategy assignment subproblem, while the convex optimization is applied for the resource distribution subproblem.
- Simulated outcomes reveal the designed scheme "ECC-PPO" works effectively on the dynamic network and can help improve the performance of the application of the multi-source data stream.

2 System model

As depicted in Fig. 1, we present a collaborative cloud-edge-end three-tier architecture with M IoT devices, N computational assess points (CAPs), and one centralized cloud server with powerful computation capacity. We use $\mathcal{M} = \{1, 2, \ldots, M\}$ and $\mathcal{N} = \{1, 2, \ldots, N\}$ to denote the IoT device set and the CAP set, respectively. Specifically, each CAP comprises one base station and one MEC server, which can serve multiple devices. Meanwhile, each device has already connected with one CAP via a wireless channel, and each CAP has associated with the cloud server through different wireless backhaul links in advance [31].

We assume that each IoT device generates a compute-intensive data stream in real time, which can be arbitrarily divided into several parts and executed simultaneously at the device, the CAP, and the cloud server. Generally speaking, the computation capacity of the device terminal is insufficient compared to the CAP and the cloud server. Therefore, the devices need to offload a portion of the data streams to the CAP or the cloud server. In addition, we also consider a practical scenario that the CAP and the cloud should charge the devices based on the size of the data stream and computation resource allocated to the devices, while the devices should purchase computation resources according to individual economic budgets. In the following, we will present the transmission, computation, and pricing models in detail.

2.1 Transmission model

As mentioned, the uplink channels from device m to CAP n and CAP n to the cloud server are wireless for offloading data streams. Besides, we presume those channels are independent, identically distributed (i.i.d), and modeled as Rayleigh channels. Specifically, let p_m^{trans} and p_n^{trans} denote the transmit power of device m and CAP n, respectively.

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | LE | TYPESET
MS Code : | CP | DISK

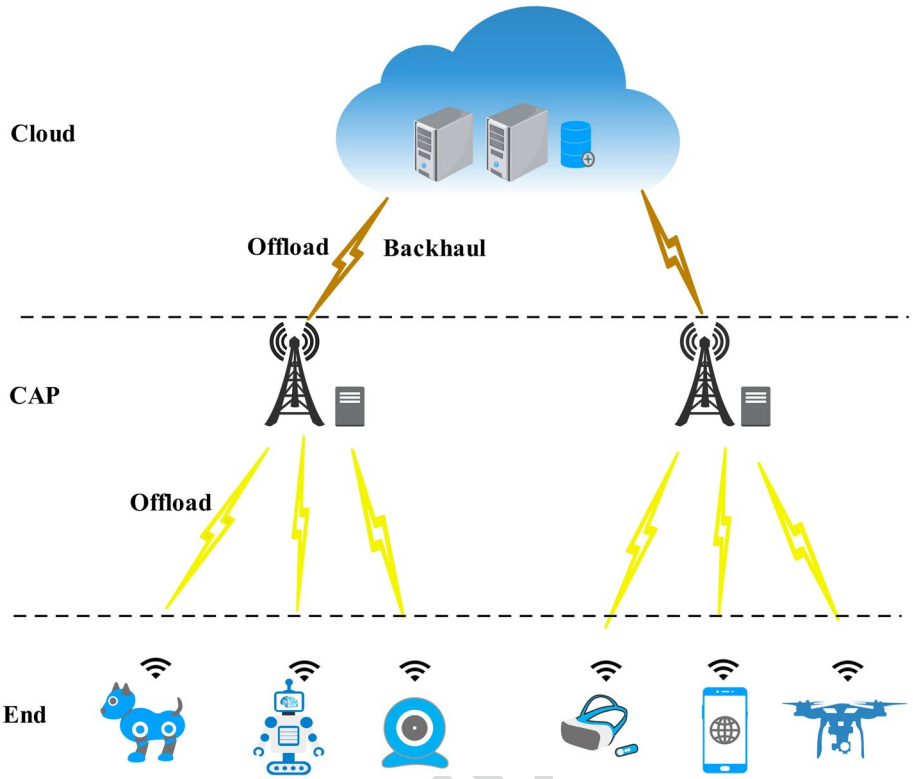

Fig. 1 System model of collaborative cloud-edge-end three-tier architecture

$h_{m,n} \sim \mathcal{CN}(0, \varpi_1)$ and $g_{m,n} \sim \mathcal{CN}(0, \varpi_2)$ are the instantaneous channel parameters between device m to CAP n and CAP n to the cloud, respectively. Then, according to Shannon's theory, the corresponding transmission rate are, respectively, given by,

$$r_{m,n}^{\mathrm{CAP}} = w_{m,n}^{\mathrm{CAP}} \log_2 \left(1 + \frac{p_m^{\mathrm{trans}} |h_{m,n}|^2}{\sigma^2}\right), \tag{1}$$

$$r_{m,n}^{c} = w_{m,n}^{c} \log_2 \left(1 + \frac{p_n^{\mathrm{trans}} |g_{m,n}|^2}{\sigma^2}\right), \tag{2}$$

where σ^2 represents the variance of additive white Gaussian noise (AWGN) [32–34]. $w_{m,n}^{\mathrm{CAP}}$ and $w_{m,n}^{c}$ are the wireless bandwidth allocated by CAP n and the cloud, which satisfies the following constraint

$$\sum_{m=1}^{M} w_{m,n}^{\mathrm{CAP}} \leq W_n^{\mathrm{CAP}}, \tag{3}$$

$$\sum_{m=1}^{M} w_{m,n}^{c} \leq W^{c}, \tag{4}$$

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

where W_n^{CAP} and W^c are the total bandwidth of CAP n and the cloud server.

2.2 Computational model

For this part, we pay attention to the computational model to minimize the latency of transmission and computation. At first, device m determines to offload α_m portion of data stream l_m to the CAP n based on some strategies. When receiving the partial data stream $\alpha_m l_m$ from device m, CAP n will judge the computational burden and its own computation capacity. If the computational burden is heavy, CAP n needs further offload β_m proportion of the received data stream $\alpha_m l_m$ to the cloud server with much more computation capacity. Otherwise, the CAP can process the data streams on its own. The detailed procedure for processing data stream l_m is shown in Fig. 2.

Let $\phi_m = (\alpha_m, \beta_m)$ denote the data stream offloading ratio vector, where $\alpha_m \in \{0, 1\}$ and $\beta_m \in \{0, 1\}$. For each device $m \in \mathcal{M}$, the size of data stream l_m for computing at local, the CAP, and the cloud are given as, respectively,

$$l_m^{\mathrm{local}} = l_m(1 - \alpha_m), \tag{5}$$

$$l_{m,n}^{\mathrm{CAP}} = l_m \alpha_m (1 - \beta_m), \tag{6}$$

$$l_{m,n}^{\mathrm{c}} = l_m \alpha_m \beta_m. \tag{7}$$

Similarly, for data stream l_m, the local device's computational latency, the CAP's computational latency, and the cloud's computational latency are given as, respectively,

$$T_m^{\mathrm{local}} = \frac{l_m^{\mathrm{local}} \omega \kappa}{f_m^l}, \tag{8}$$

$$T_{m,n}^{\mathrm{CAP}} = \frac{l_{m,n}^{\mathrm{CAP}} \omega \kappa}{f_{m,n}^{\mathrm{CAP}}}, \tag{9}$$

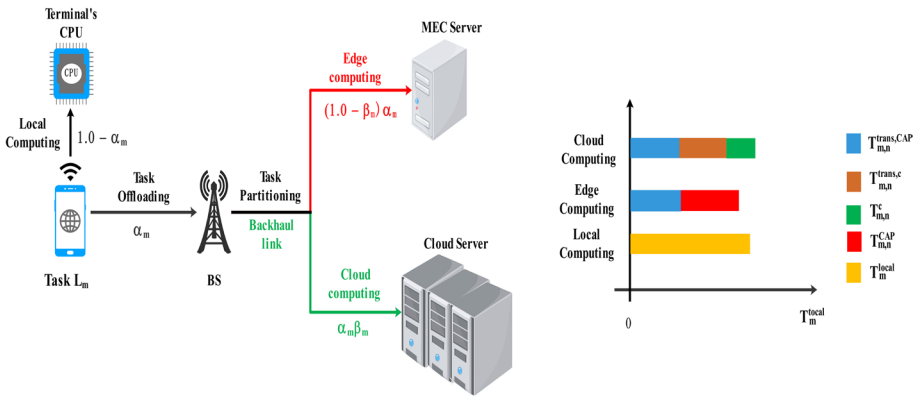

Fig. 2 The detailed procedure for processing data stream l_m

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

$$T_{m,n}^{\mathrm{c}} = \frac{l_{m,n}^{\mathrm{local}} \omega \kappa}{f_m^{\mathrm{c}}}, \tag{10}$$

where ω denotes the CPU cycles computing per bit data stream, κ represents the unit conversion factor from Mbs to bits, $f_{m,n}^{\mathrm{CAP}}$ and f_m^{c} are the CPU frequency allocated by the CAP n and the cloud server, and f_m^l denote the device m's computation capacity. According to (1) and (2), the transmission latency from device m to CAP n and CAP n to the cloud are written as, respectively,

$$T_{m,n}^{\mathrm{trans,\,CAP}} = \frac{l_m \alpha_m \kappa}{r_{m,n}^{\mathrm{CAP}}}, \tag{11}$$

$$T_{m,n}^{\mathrm{trans,\,c}} = \frac{l_m \alpha_m \beta_m \kappa}{r_{m,n}^{c}}. \tag{12}$$

Since the results feedback is small, we ignore the return latency. We assume that each device and each CAP has a transmitting unit and a computing unit, which can work simultaneously. Thus, data stream l_m can be processed and transmitted in parallel. Therefore, the latency for processing data stream l_m is given by

$$T_m^{\mathrm{total}} = \max\{T_m^{\mathrm{local}}, T_{m,n}^{\mathrm{trans,\,CAP}} + T_{m,n}^{\mathrm{CAP}}, T_{m,n}^{\mathrm{trans,\,CAP}} + T_{m,n}^{\mathrm{trans,\,c}} + T_{m,n}^{c}\}. \tag{13}$$

Moreover, the CAP's server or the cloud server should create a virtual machine for each device. Thereby, all data streams of the M devices can be executed simultaneously. The total processing latency of all devices are

$$T^{\mathrm{total}} = \max_{m \in \mathcal{M}} T_m^{\mathrm{total}}. \tag{14}$$

2.3 Pricing model

The pricing model consists of two parts: basic service fee τ_m and calculation service fee π_m. The former is related to the size of the data stream l_m transmitted to the CAP or the cloud server, while the latter is correlated with the computational resource. According to [35], the payment for offloading data stream l_m is

$$\tau_m = \zeta_1(\alpha_m l_m) + \zeta_2(\beta_m \alpha_m l_m), \tag{15}$$

$$\pi_m = \eta_1(f_{m,n}^{\mathrm{CAP}}) + \eta_2(f_m^c), \tag{16}$$

$$U_m = \tau_m + \pi_m, \tag{17}$$

where ζ_1 denotes the unit price for transmitting per Mb of the data stream from device m to the CAP, ζ_2 represents the price coefficient per Mb of the data stream transmitted

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

from the CAP to the cloud, and η_1 and η_2 are the unit price of computing capacity for the CAP and the cloud. In addition, each device m has a finite budget to buy the service, and the limitation for the budget of device m is given by

$$U_m \leq U_m^{max}. \tag{18}$$

3 Problem formulation

In this part, we first denote a sequence time slot $k = \{1, 2, \ldots, K\}$, where the total system delay at time slot k is presented by $T^{\text{total}}(k)$. Then, at each time slot k, our goal is to obtain a minimum total system delay under the constrained devices' budget by jointly optimizing offloading ratio assignment, bandwidth resource management, and computation resource allocation in the considered network, given by

$$\mathbf{P1}: \min_{\{\boldsymbol{\phi}(k), \boldsymbol{w}(k), \boldsymbol{f}(k)\}} T^{\text{total}}(k), \tag{19a}$$

$$\text{s.t.} C_1: \alpha_m(k) \in [0,1], \beta_m(k) \in [0,1], \forall m \in \mathcal{M}, \tag{19b}$$

$$C_2: \sum_{m=1}^{M} w_{m,n}^{\text{CAP}}(k) \leq W_n^{\text{CAP}}, \forall n \in \mathcal{N}, \tag{19c}$$

$$C_3: \sum_{m=1}^{M} w_{m,n}^{c}(k) \leq W^{c}, \tag{19d}$$

$$C_4: \sum_{m=1}^{M} f_{m,n}^{\text{CAP}}(k) \leq F_n^{\text{CAP}}, \forall n \in \mathcal{N}, \tag{19e}$$

$$C_5: \sum_{m=1}^{M} f_{m}^{c}(k) \leq F^{c}, \tag{19f}$$

$$C_6: U_m \leq U_m^{max}, \tag{19g}$$

where C_1 ensures the range of the offloading assignment α_m and β_m. C_2 and C_3 represent the sum of the bandwidth resources assigned to each device, which cannot exceed the entire bandwidth at the CAP or the cloud server. Analogously, C_4 and C_5 denote the limitation of computational resource allocation at the CAP or the cloud server, while C_6 is the maximal budget of each device for purchasing resource services. For convenience, we denote the vectors $\boldsymbol{\phi} = \{\phi_m\}$, $\boldsymbol{w} = \{w_{m,n}^{\text{CAP}}, w_{m,n}^{c}\}$, and $\boldsymbol{f} = \{f_{m,n}^{\text{CAP}}, f_{m}^{c}\}$ as the offloading assignment, bandwidth resources management, and computation resources allocation, respectively.

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | LE | TYPESET
MS Code : | CP | DISK

In this article, what is noteworthy is that a multi-source environment is considered, where the wireless channel is time-varying and the characteristic of the data stream is dynamic at each time slot. To adapt to the multi-source environment and minimize the system latency, we design an algorithm to make decisions at each time slot. In particular, the decision-making at the current time slot k can be used as a reference for that at the next time slot $k+1$. Therefore, we can design the data stream offloading process with resource allocation as a Markov Decision Process (MDP), detailed in the next subsection.

3.1 MDP

As illustrated in the last section, we formulate the data stream offloading problem as an MDP with a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$. We use $a_k \in \mathcal{A}$, $r_k \in \mathcal{R}$, and $s_k \in \mathcal{S}$ to denote the action, reward, as well as state at time frame k, respectively.

3.1.1 State space

For the considered network, we consider a multi-source scenario, where the data stream is heterogeneous and the channel gain is time-varying at each time slot. Therefore, at the beginning time slot k, the system state is depicted as $s_k = \{T^{\text{total}}(k-1), \boldsymbol{\phi}(k-1), \boldsymbol{L}(k), \boldsymbol{H}(k), \boldsymbol{G}(k)\}$, where $\boldsymbol{L}(k) = [l_1(k), l_2(k), \ldots, l_M(k)]$ denotes the data stream vector, $\boldsymbol{H}(k) = [|h_{1,n}|^2(k), |h_{2,n}|^2(k), \ldots, |h_{M,n}|^2(k)]$ and $\boldsymbol{G}(k) = [|g_{1,n}|^2(k), |g_{2,n}|^2(k), \ldots, |g_{M,n}|^2(k)]$ are the instantaneous channel gain vector.

3.1.2 Action space

Recall the problem **P1**, the main factors affecting the system latency are offloading strategy $\boldsymbol{\phi}(k)$, bandwidth resource management $\boldsymbol{w}(k)$, and computation resource allocation $\boldsymbol{f}(k)$. Hence, the action is defined as $a_k = \{\boldsymbol{\phi}(k), \boldsymbol{w}(k), \boldsymbol{f}(k)\}$. Mention that the number of pairs $\{state, action\}$ is infinite, due to the continuous values of a_k and s_k. Therefore, it is inefficient to use a table to store all pairs or apply a value-based method to solve **P1**. To solve this problem, we use a deep neural network (DNN) $\psi(a|s;\theta^a)$ to approximate policy function $\psi(a|s)$, guiding the agent to do action a under state s.

3.1.3 Reward function

The core of the reward function is to evaluate the qualify of action a_t. Specifically, a positive reward will be given, if the agent makes a decision that efficiently minimizes the system latency and vice versa. Thereby, we defined the reward function related to time slot t as

$$r_k = -T^{\text{total}}(k). \tag{20}$$

For this considered network, there exists a central control at the cloud for the considered network, which can obtain all the device information and the whole system network status. Therefore, the central control is regarded as an agent. At the beginning time slot k, the agent first observes the system state s_k and makes a decision a_k based on $\psi(a|s)$. Then, the system will give an immediate reward r_k to the agent, and alter its state from

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

s_k to s_{k+1} with a transit probability $\mathcal{P}$. This process will last for a long time until an end state S_{end} is observed. Meanwhile, the agent's target is to acquire an optimal $\psi^*(a|s)$ to obtain a long-term accumulated reward C_k from the original state s_k, given by

$$C_k = \sum_{t=0}^{\infty} \gamma^t r_{k+t}, \tag{21}$$

where γ is the discount factor.

Although we formulate a specific MDP to deal with **P1**, it is still untoward to solve, due to the max operator and many variables. Since **P1** is a min-max problem, and we can transform **P1** into a mean weighted-sum problem **P2** according to [29, 36, 37], given as

$$\mathbf{P2}: \min_{\{\boldsymbol{\phi}(k),\boldsymbol{w}(k),\boldsymbol{f}(k)\}} \sum_{m=1}^{M} T_m^{\text{total}}(k), \tag{22a}$$

$$(19b) - (19g). \tag{22b}$$

Note that **P2** is also an MDP problem. However, we find that the transition probability of the wireless channels and the data streams' characteristics are unknown. Moreover, the dimension of the action is high and causes a huge action space. In further, the variable of offloading assignment $\boldsymbol{\phi}(k)$ and resource allocation $\{\boldsymbol{w}(k),\boldsymbol{f}(k)\}$ are tightly coupled, leading to difficulty in convergence. To deal with these issues, we decompose problem **P2** into offloading strategy allocation sub-problem and resources allocation sub-problem, where we design a novel DRL-based approach to handle the sub-problems efficiently, specified in the following.

4 Problem decomposition

As illustrated before, the dimension of action is high, which leads to a huge action space. Moreover, the sub-action, which includes offloading ratio assignment $\boldsymbol{\phi}$, bandwidth allocation $\boldsymbol{w}$, and computational allocation $\boldsymbol{f}$, are closely related. Besides, we perceive a vital phenomenon that the data stream offloading $\boldsymbol{\phi}$ assignment affects the transmission delay and the computational simultaneously, but the bandwidth allocation $\boldsymbol{w}$ and the computation allocation $\boldsymbol{f}$ can only influence them, respectively. Therefore, we decompose **P2** into a resources allocation sub-problem and an offloading strategy allocation sub-problem. The former sub-problem is only related to the bandwidth resource allocation and computation resource allocation, and the latter sub-problem is involved with the offloading assignment. We solve the former and the latter sub-problems by convex optimization and DRL methods, respectively.

4.1 Convex optimization based for resource allocation sub-problem

It is obvious that, at any time slot, the resource allocation sub-problem to optimize $(\boldsymbol{w},\boldsymbol{f})$ is a convex optimization problem with linear and convex constraints, given the offloading strategy $\boldsymbol{\phi}$. Therefore, we convert **P2** into

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

$$\mathbf{P3}: \min_{\{\boldsymbol{w},\boldsymbol{f}\}} \sum_{m=1}^{M} (T_{m,n}^{\text{trans, CAP}} + T_{m,n}^{\text{trans, c}} + T_{m,n}^{\text{CAP}} + T_{m,n}^{c}), \tag{23a}$$

$$(19c) - (19g). \tag{23b}$$

The optimal solution can be obtained with the standard convex optimizer, which often needs iteration to solve. To get the optimal solution without iteration, we further slack the budget's constraint (19g), converting **P3** into a Lagrange problem, written as,

$$\begin{aligned}\mathbf{P4}: & \mathcal{L}(\boldsymbol{w},\boldsymbol{f},\lambda,\delta,\mu,\nu) \\ &= \sum_{m=1}^{M} (T_{m,n}^{\text{trans, CAP}} + T_{m,n}^{\text{trans, c}} + T_{m,n}^{\text{CAP}} + T_{m,n}^{c}) \\ &\quad + \lambda\left(\sum_{m=1}^{M} w_{m,n}^{\text{CAP}} - W_n^{\text{CAP}}\right) + \delta\left(\sum_{m=1}^{M} f_{m,n}^{\text{CAP}} - F_n^{\text{CAP}}\right) \\ &\quad + \mu\left(\sum_{m=1}^{M} w_m^{c} - W^{c}\right) + \nu\left(\sum_{m=1}^{M} f_m^{c} - F^{c}\right),\end{aligned} \tag{24}$$

where β, δ, μ and ν are Lagrangian multipliers. Let us take the partial derivatives $\boldsymbol{f}$ and $\boldsymbol{w}$,

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w_{m,n}^{\text{CAP}}} &= -\frac{A_m}{(w_{m,n}^{\text{CAP}})^2} + \lambda, \\ \frac{\partial \mathcal{L}}{\partial f_{m,n}^{\text{CAP}}} &= -\frac{B_m}{(f_{m,n}^{\text{CAP}})^2} + \delta, \\ \frac{\partial \mathcal{L}}{\partial w_{m,n}^{c}} &= -\frac{C_m}{(w_{m,n}^{c})^2} + \mu, \\ \frac{\partial \mathcal{L}}{\partial f_{m}^{c}} &= -\frac{D_m}{(f_{m}^{c})^2} + \nu, \end{cases} \tag{25}$$

where

$$A_m = \frac{l_m \alpha_m \kappa}{\log_2\left(1 + \frac{p_m^{\text{trans}} |h_{m,n}|^2}{\sigma^2}\right)}, \tag{26}$$

$$B_m = \frac{l_m \alpha_m \beta_m \kappa}{\log_2\left(1 + \frac{p_n^{\text{trans}} |g_{m,n}|^2}{\sigma^2}\right)}, \tag{27}$$

$$C_m = l_m \alpha_m (1 - \beta_m) \kappa \omega, \tag{28}$$

$$D_m = l_m \alpha_m \beta_m \kappa \omega. \tag{29}$$

By setting the partial derivative (25) to zero, we can get a optimal solution $\boldsymbol{w}^* = \{w_{m,n}^{\text{CAP}}, w_{m,n}^{c}\}, \forall m \in \mathcal{M}$ and $\boldsymbol{f}^* = \{f_{m,n}^{\text{CAP}}, f_m^{c}\}, \forall m \in \mathcal{M}$,

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | LE | TYPESET
MS Code : | CP | DISK

$$w_{m,n}^{\mathrm{CAP}} = \frac{\sqrt{A_m}}{\sum_{m=1}^{M}\sqrt{A_m}} W_n^{\mathrm{CAP}}, \tag{30}$$

$$f_{m,n}^{\mathrm{CAP}} = \frac{\sqrt{B_m}}{\sum_{m=1}^{M}\sqrt{B_m}} F_n^{\mathrm{CAP}}, \tag{31}$$

$$w_{m,n}^{c} = \frac{\sqrt{C_m}}{\sum_{m=1}^{M}\sqrt{C_m}} W^{c}, \tag{32}$$

$$f_{m}^{c} = \frac{\sqrt{D_m}}{\sum_{m=1}^{M}\sqrt{D_m}} F^{c}. \tag{33}$$

Note that $\boldsymbol{w}^*$ is always effective, since it has nothing to do with the device's budget, while $\boldsymbol{f}^*$ is available only if the constraint (19g) holds. Otherwise, the optimal solution $\boldsymbol{f}^*$ is obtained by the conventional convex tools, e.g., the CVX tools.

4.2 Proximal policy optimization

In this part, we employ the proximal policy optimization (PPO) strategy to solve the offloading ratio assignment sub-problem owing to its advantage of stability and practicability [38, 39]. The PPO strategy originates from the actor-critic scheme, which can effectively deal with continuous action space. Specifically, we use $\psi(a|s;\theta^a)$ and $V(s;\theta^v)$ to denote the actor and critic network, where θ^a and θ^v are the parameter sets of the two DNNs, respectively. The actor DNN $\psi(a|s;\theta^a)$ is responsible for making decisions a given state s, while the critic DNN $V(s;\theta^v)$ is to assess the value of state s.

For the critic network, in order to reduce the error between real value and estimated value generated by $V(s;\theta^v)$, the temporal-difference (TD) scheme is utilized for the loss function, designed by

$$L(\theta^v) = \mathbb{E}_k[r_k + \gamma V(s_{k+1};\theta^v) - V(s_k;\theta^v)]^2, \tag{34}$$

where $\mathbb{E}_k(\cdot)$ is the expectation operator over k samples.

Let $\varrho = (s_1, a_1, r_1, \ldots, s_k, a_k, r_k)$ denote a trajectory in an episode. Generally speaking, the traditional actor network is relied on the policy gradient method, which needs a complete sequence trajectory ϱ and updates itself in one episode, leading to slow convergence and local optima. To deal with this issue, the well-known method PPO-clip [40] is adapted for the actor network, given by

$$L^{\mathrm{clip}}(\theta^a) = \mathbb{E}_k[\min(F(\theta_{new}^a)\hat{A}_k, \mathrm{CLIP}(F(\theta_{new}^a)\hat{A}_k))], \tag{35}$$

where $F(\theta_{new}^a)$ represents the ratio of the difference between the old and new strategies, given by

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | LE | TYPESET
MS Code : | CP | DISK

$$F(\theta^a_{new}) = \frac{\psi(a_k|s+k;\theta^a_{new})}{\psi(a_k|s+k;\theta^a_{old})}. \tag{36}$$

The function CLIP(y) is the clip operator constraining the value of x in $[1-\varepsilon, 1+\varepsilon]$, expressed by

$$C(y) = \begin{cases} 1+\varepsilon, & y > 1+\varepsilon, \\ y, & 1-\varepsilon \le y \le 1+\varepsilon, \\ 1-\varepsilon, & y < 1-\varepsilon, \end{cases} \tag{37}$$

where ε denotes the clip factor. In addition, $\hat{A}_k$ represents the advantage function applied to reduce variance but may increase bias. To make a balance between variance and bias, the generalized advantage estimation (GAE) method [41] is used for the advantage function, written as

$$\delta_k = r_k + \gamma V(s_{k+1};\theta^v) - V(s_k;\theta^v), \tag{38}$$

$$\hat{A}_k^{\mathrm{GAE}(\gamma,\chi)} = \sum_{l=0}^{\infty} (\gamma\chi)^l \delta_{k+l}, \tag{39}$$

where χ is the trade-off coefficient between variance and bias. In the next part, we will present the PPO-based training workflow for data stream offloading sub-problem.

4.3 DRL-based training workflow for data stream offloading assignment sub-problem

For the offloading decision sub-problem, we only focus on using the DRL to get the offloading strategy $\boldsymbol{\phi}(k)$ at each time slot k, given $\boldsymbol{w}(k)$ and $\boldsymbol{f}(k)$. The agent training procedure begins with the initialization of parameter sets θ^v, θ^a_{new}, and θ^a_{old}, respectively, and of the experience pool $\mathcal{B}$. At the beginning time slot k, the agent estimates the channel parameters by some channel estimations and obtains the basic devices' data. Then, the agent makes an offloading decision a_k by running policy $\psi(\cdot|s_k, \theta^a_{old})$ and execute the convex optimization to acquire $\boldsymbol{w}(k)$ and $\boldsymbol{f}(k)$. Meanwhile, the system gives a reward r_k and moves to the next state s_{k+1}. Then, the agent collect the experience (s_k, a_k, r_k, s_{k+1}) into experience pool $\mathcal{B}$ for updating, and interacts with the system K times. As $\mathcal{B}$ is full, the agent updates the parameter sets θ^a_{new} and θ^v by using PPO-clip methods. We regard the training workflow as one episode and train the agent for N episodes.

For the considered system, we only use the DRL to obtain the offloading ratio assignment $\boldsymbol{\phi}$, and acquire the resource allocation $\boldsymbol{w}$ and $\boldsymbol{f}$ by the convex optimization, as this can improve the performance of the agent. Moreover, importance sampling is utilized according to the PPO-clip methods, where the agent samples data based on the old policy $\psi(\cdot|s;\theta^a_{old})$ for updating the parameters θ^a_{new} of the new policy $\psi(\cdot|s;\theta^a_{new})$, which can reuse the data to speed up the convergence. The detailed algorithm is presented in Algorithm 1.

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | LE | TYPESET
MS Code : | CP | DISK

Algorithm 1 Data stream offloading assignment and resource allocation based on the devised ECC-PPO scheme

Input: state $s_k = \{T^{\mathbf{total}}(k-1), \boldsymbol{\phi}(k-1), \boldsymbol{L}(k), \boldsymbol{H}(k), \boldsymbol{G}(k)\}$
Output: action $a_k = \{\boldsymbol{\phi}(k), \boldsymbol{w}(k), \boldsymbol{f}(k)\}$

Initialize the parameter sets θ^{ν}, θ^{a}_{old}, and θ^{a}_{new} of the critic network, the old and new actor network, respectively.
$\theta^{a}_{old} \leftarrow \theta^{a}_{new}$
for episode $= 1, 2, \cdots, N$ **do**
  Initialize state s_0 from the system.
  for time slot $k = 1, 2, \cdots, K$ **do**
    The central control obtains the basic device's data and estimates channel parameters.
    Select the offloading strategy $\boldsymbol{\phi}(k)$ by running policy $\psi(\cdot|s_k; \theta^{a}_{old})$, derive the bandwidth allocation $\boldsymbol{w}(k)$ and computation allocation $\boldsymbol{f}(k)$ by (30)-(33).
    Get reward r_k and next state s_{k+1}
    Calculate GAE (advantage function) $\hat{A}_k$ by (38) and (39).
    Collect (s_k, a_k, r_k, s_{k+1}) into experience pool $\boldsymbol{B}$
    if $\boldsymbol{B}$ is full **or** $k == K$ **then**
      for $m = 1, 2, \cdots, M$ **do**
        Update θ^{ν} via (34)
      end for
      for $m = 1, 2, \cdots, M$ **do**
        Update θ^{a}_{new} via (35)
      end for
      $\theta^{a}_{old} \leftarrow \theta^{a}_{new}$
      Empty experience pool $\boldsymbol{B}$
    end if
  end for
end for

5 Simulation

This section will show some simulation outcomes to evaluate the practicability of the devised optimization algorithm for data stream offloading and resource allocation. In the simulations, the considered network experiences Rayleigh flat fading channels, and a path-loss model is considered with a loss exponential 3 [42, 43]. Besides, the distance from devices to the cloud is normalized as unity, where the distance from devices to CAPs is denoted as $d \in (0, 1)$. Similarly, the distance from CAPs to the cloud is $(1 - d)$. For such, $\varpi_1 = d^{-3}$ and $\varpi_2 = (1 - d)^{-3}$. If not specified, we set $d = 0.2$, $p_n^{\text{trans}} = 2$ W, and $p_m^{\text{trans}} = 1$ W. For the network, there exist 2 heterogeneous CAPs with different computation capacities, which are set to 8×10^8 and 1.2×10^9 cycles per second (cyc/s), while the cloud server has a more powerful computation capacity with 1×10^{10} cyc/s. Logically, each IoT application m has a smaller computation capacity, following a distribution $f_m^l \sim \mathcal{U}(1 \times 10^8, 1.5 \times 10^8)$ cyc/s. In addition, each CAP is connected to 4 different computational sizes of data streams, subjecting to the uniform distribution with $l_1 \sim \mathcal{U}(120, 160)$, $l_2 \sim \mathcal{U}(130, 140)$, $l_3 \sim \mathcal{U}(60, 80)$ and $l_4 \sim \mathcal{U}(40, 60)$ Mb, respectively. For the service charge part, the basic service prices η_1 and η_2 are set to 0.1 and 0.2 per Mb, while the calculation service prices η_1 and η_2 are set to 10 and 2 per computation unit. The detailed network parameters are listed in Table 1.

For the DRL framework, the critic DNN has two fully connected layers with 64 and 128 nodes, and the actor DNN consists of three fully connected layers with 64, 256, and 64 nodes, respectively. To enhance the fitness of DNN, the Rectified Linear Unit (ReLU) is used as the activated function. Moreover, the DRL training process is sped up by adapting the Adam optimizer method. In addition, the DRL training process

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

Table 1 Parameter setting

Parameters	Value
Number of devices M	8
Number of CAPs N	2
Variance of the AWGN σ^2	0.01
Bandwidth of nth CAP and the cloud server	{4 Mhz, 4 Mhz, 12 Mhz}
Computation capacity of nth CAP and the cloud server	$\{8 \times 10^8, 1.2 \times 10^9, 1.0 \times 10^{10}\}$ cyc/s
Distribution of each device's computation capacity f_m^l	$\mathcal{U}(1.0 \times 10^8, 1.5 \times 10^8)$ cyc/s
Distribution of the size of data stream	$l_1 \sim \mathcal{U}(120, 160)$ Mb $l_2 \sim \mathcal{U}(130, 140)$ Mb, $l_3 \sim \mathcal{U}(60, 80)$ Mb, $l_4 \sim \mathcal{U}(40, 60)$ Mb $l_i \sim \mathcal{U}(40, 80)$ Mb, $\forall i > 4$
Transmit power p_m^{trans} of mth IoT device	1 W
Transmit power p_n^{trans} of nth CAP	2 W
Conversion factor κ from Mb to bit	2^{20}
CPU's cycles for calculating one bit data stream ω	40

consists of 200 episodes and each episode has 256 time slots, where the hyper-parameters ε, γ, χ, and $|\mathcal{B}|$ are set to 0.2, 0.92, 0.95, and 128, respectively. To avoid accidents, the training process repeats at least 10 times.

In the simulations, we present six offloading schemes for comparison

- ALL-Local: data streams of M devices are computed locally.
- ALL-CAP: data streams of M devices are fully offloaded to the CAP for computing with average resource allocation.
- ALL-Cloud: data streams of M devices are fully offloaded to the cloud for computing with average resource allocation.
- ECC-PPO: the proposed strategy by using the cloud-edge-end computing framework for resource allocation and data stream offloading assignment.
- MEC-PPO: the computation of data streams is assisted by the CAP using the proposed strategy.
- MCC-PPO: the computation of data streams is assisted by the cloud server using the proposed strategy.

Figure 3 presents the convergence of the devised strategy with the device's budget $U = 160$. As seen from Fig. 3, the system latency of the "ECC-PPO" strategy drops slowly in the first 20 episodes, but then faster in the next 60 episodes, eventually converging at about 100 episodes. Thanks to "ECC-PPO," the action space is largely reduced. Therefore, the DRL agent tends to find a feasible solution after a few training episodes. Moreover, the proposed "ECC-PPO" scheme shows excellent potential in minimizing the system latency, compared to the other five schemes. In particular, in the 140th episode, the system latency of the "ECC-PPO" scheme is close to 15 s approximately, which is the lowest value in the six proposed schemes, about 16%, 31%, 50%, 62%, and 70% lower than "MCC-PPO," "MEC-PPO," "ALL-Cloud,"

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

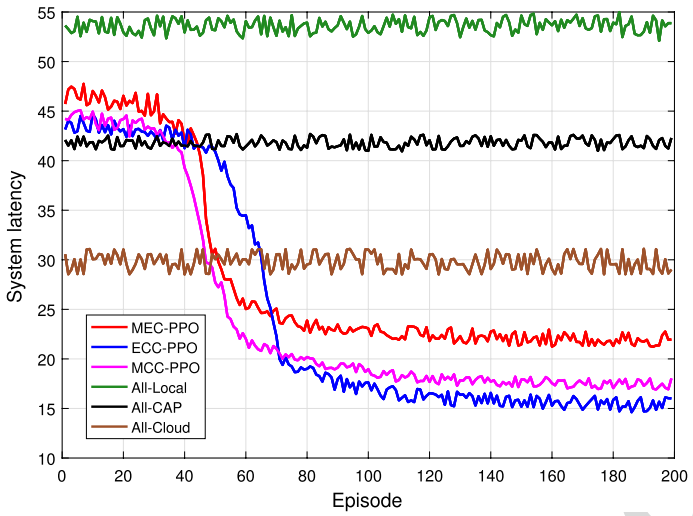

Fig. 3 Convergence of the devised scheme versus episode

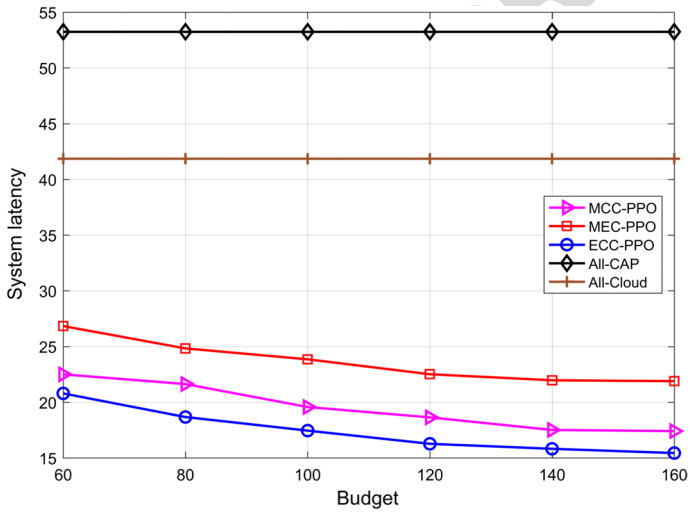

Fig. 4 The impact of the device's budget on the system latency

"All-CAP," and "All-Local" schemes, respectively. The above results in Fig. 3 indicate the effectiveness of the devised scheme for the considered network.

Figure 4 illustrates the impact of the device's budget on the system latency, where the budget ranges in [60, 160]. Obviously, since the device with more budget can buy more computation resources, the performance of the "MCC-PPO," "MEC-PPO," and "ECC-PPO" strategies becomes better as the device's budget increases. Meanwhile, the system latency of "ALL-Cloud" and "ALL-Local" strategies remains stable, as the devices' budget has nothing to do with the fullying offloading strategy. Besides, there exists a little reduction in the system latency when the budget goes from 140 to 160. This is mainly because the transmission latency rather than the computation latency affects the system performance when the device's budget has high budget. Moreover,

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

the system latency of the "ECC-PPO" strategy is the lowest of the five strategies whether the user budget is high or not low, which illustrates the superiority of the proposed "ECC-PPO" strategy in reducing system latency.

Figure 5 reveals the influence of the number of devices on the system performance, in which the number of devices M varies from 6 to 16. Obviously, the system overhead of all schemes rises as the number of devices rises. That is reasonable since the growing number of devices generates more computational data stream and thereby puts a heavy computational burden on the cloud-edge-end system. Although the number of devices impacts the system overhead seriously, we still observe the proposed "ECC-PPO" approach performs better than other ones, which further verifies the superiority of the proposed method for data stream unpacking and resource allocation.

Figure 6 depicts the influence of CAP's bandwidth on the system's overhead among the three schemes, where the bandwidth at each CAP ranges in [2, 10] MHz. From Fig. 6, each scheme shows a small gap between $M = 8$ and $M = 12$. That is reasonable, as the number of device users increases, the task data stream offloaded to the server becomes larger, further increasing the pressure on the network transmission and resulting in significant transmission delays. Therefore, the increasing number of devices M significantly affects the system performance. Moreover, the system performance improves as the bandwidth increases at each CAP. That result is as expected since increasing bandwidth can effectively increase wireless transmission capacity, which can reduce transmission latency and thus enhance the system's performance. In further, the "ECC-PPO" scheme behaves best among the three strategies. For example, when the bandwidth of CAP equals 8 Mhz and $M = 12$, the performance of "ECC-PPO" scheme is about 14% and 16% better than that of "MCC-PPO" and "MEC-PPO" schemes. This result verifies the effectiveness of the designed cloud-edge-end framework.

Figure 7 shows the influence of CAP's computation on the system performance with the device's budget $U = 50$ and $U = 100$. As expected, the system latency decreases swiftly and then remains steady when the CAP's computation capacity changes from

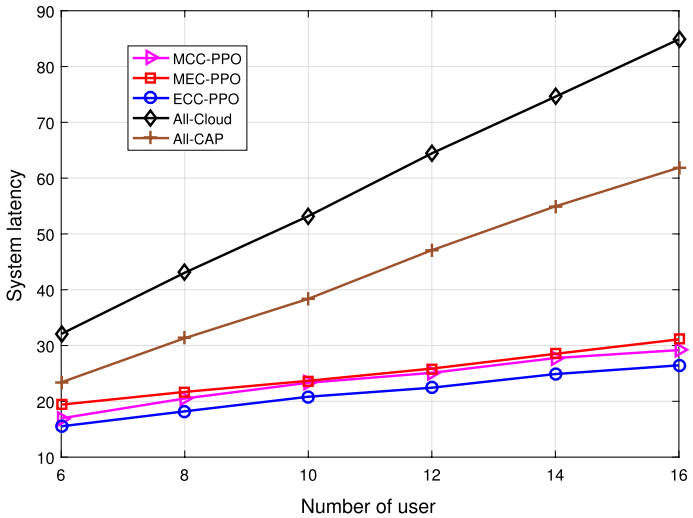

Fig. 5 The influence of the number of devices on the system performance

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

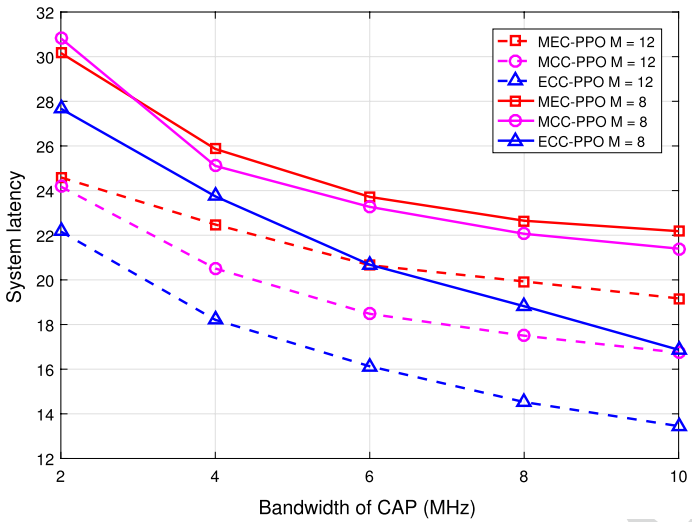

Fig. 6 The influence of CAP's bandwidth on the system's overhead with $M = 8$ and $M = 12$

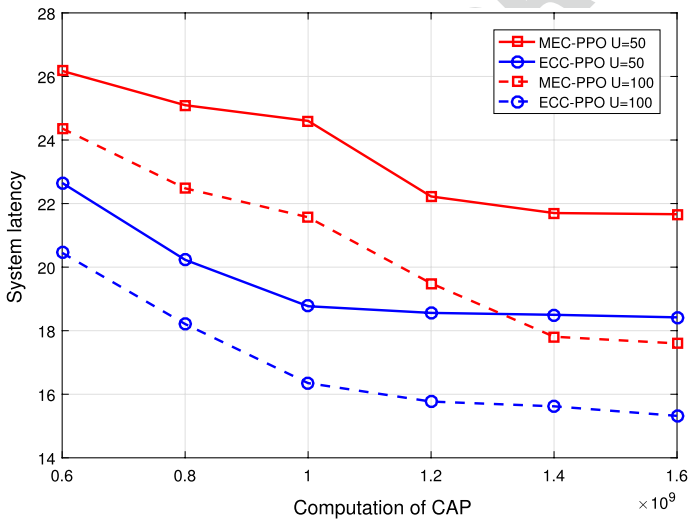

Fig. 7 The influence of CAP's computation capacity on the system performance with the device's budget $U = 50$ and $U = 100$

0.6×10^9 cyc/s to 1.6×10^9 cyc/s. This is owing to the fact, the growth of the number of computing resources leads to a reduction in the completion time of each offloading data stream, while it is also important to ensure the constraint of the computational resource unit that the device can purchase. Moreover, the "ECC-PPO" scheme performs better than the "MEC-PPO" scheme when the computation resource at the CAPs is sufficient, which shows the superiority of the collaborative cloud-edge-end framework for intelligent resource allocation.

Figure 8 demonstrates the impact of the cloud server's computation capacity on the system overhead with the device's budget $U = 50$ and $U = 100$. From Fig. 8, the system overhead gradually decreases as the computation resource of the cloud server

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

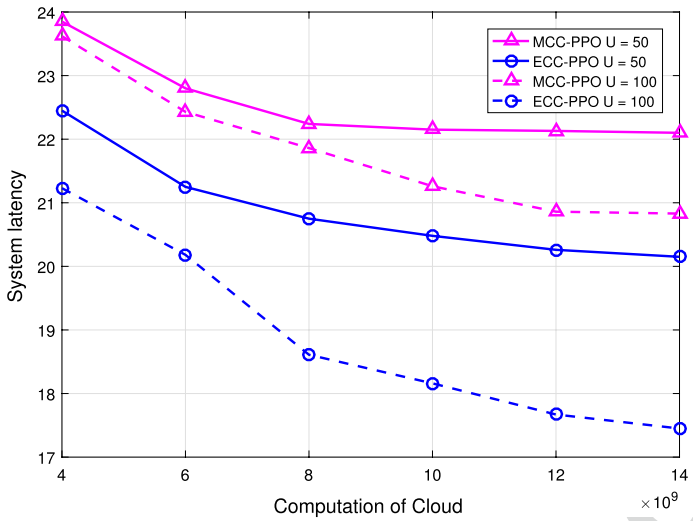

Fig. 8 The impact of cloud server's computation capacity on the system overhead with the device's budget $U = 50$ and $U = 100$

increase. This is because the cloud server with much computation capacity can process data streams more efficiently, and more data streams can be offloaded to the cloud to reduce overall system latency. Besides, the device's budget U is also an important factor influencing the system overhead. When the computation resource of the cloud server is abundant, the system latency of the "ECC-PPO" scheme with $U = 100$ is much lower than that with $U = 50$. Moreover, the "ECC-PPO" scheme performs better than the "MCC-PPO", which shows the advantage of the devised cloud-edge-end architecture in resource allocation and data stream offloading assignment.

6 Conclusion

To enhance the performance of MEC-assisted multi-source data stream computing, we investigated a cloud-edge-end computing network, where the CAPs and the cloud server can collaboratively help process the data streams from IoT devices. For this considered network, the wireless channel was time-varying, and the characteristic of the data streams was variable. To adapt to this dynamic network, we proposed a novel approach that combined the PPO with the lagrangian multiplier method for offloading ratio assignment and resource allocation. Finally, simulation outcomes demonstrated the superiority of the devised scheme and could help develop the multi-source data stream application. In future works, we will discuss more MEC scenarios, e.g., NOMA-based MEC network and IRS-assisted MEC network, and utilize multi-agent reinforcement learning to deal with the offloading strategy.

Abbreviations

IoT	Internet of Things
CAP	Computational access points
PPO	Proximal policy optimization
DRL	Deep reinforcement learning
MEC	Mobile edge computing
MCC	Mobile cloud computing
DQN	Deep Q network

Journal : BMCOne 13634 | Dispatch : 14-5-2023 | Pages : 20
Article No : 1018 | ☐ LE | ☐ TYPESET
MS Code : | ☑ CP | ☑ DISK

IRS	Intelligent reflect surface
DNN	Deep neural network
NOMA	Non-orthogonal multiple access
GAE	Generalized advantage estimation

Author contributions
Y.W. devised the proposed framework and performed the simulations; C.G. and X.B. helped revise the manuscript grammar check; J.X. helped formulate the problem optimization; S.L. helped design the deep neural networks; C.C. helped conduct the simulation, and Y.T. helped explain the simulation outcomes. All authors read and approved the final manuscript.

Funding
The work in this paper was supported by National Key R&D Program of China (No. 2020YFB1808101), and by the Key-Area Research and Development Program of Guangdong Province, China (No. 2019B090904014).

Availability of data and materials
The authors state the data availability in this manuscript.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that there is no competing interests regarding the publication of this paper.

Received: 14 December 2022 Accepted: 27 April 2023

References

1. Z. Na, B. Li, X. Liu, J. Wan, M. Zhang, Y. Liu, B. Mao, UAV-based wide-area internet of things: an integrated deployment architecture. IEEE Netw. **35**(5), 122–128 (2021)
2. X. Liu, C. Sun, M. Zhou, C. Wu, B. Peng, P. Li, Reinforcement learning-based multislot double-threshold spectrum sensing with Bayesian fusion for industrial big spectrum data. IEEE Trans. Ind. Inform. **17**(5), 3391–3400 (2021)
3. S. Tang, Dilated convolution based CSI feedback compression for massive MIMO systems. IEEE Trans. Veh. Technol. **71**(5), 211–216 (2022)
4. W. Wu, F. Zhou, R.Q. Hu, B. Wang, Energy-efficient resource allocation for secure NOMA-enabled mobile edge computing networks. IEEE Trans. Commun. **68**(1), 493–505 (2020)
5. X. Liu, Q. Sun, W. Lu, C. Wu, H. Ding, Big-data-based intelligent spectrum sensing for heterogeneous spectrum communications in 5G. IEEE Wirel. Commun. **27**(5), 67–73 (2020)
6. W. Zhou, X. Lei, Priority-aware resource scheduling for UAV-mounted mobile edge computing networks. IEEE Trans. Veh. Technol. **PP**(99), 1–6 (2023)
7. W. Wu, F. Zhou, B. Wang, Q. Wu, C. Dong, R.Q. Hu, Unmanned aerial vehicle swarm-enabled edge computing: potentials, promising technologies, and challenges. IEEE Wirel. Commun. **29**(4), 78–85 (2022)
8. S. Tang, L. Chen, Computational intelligence and deep learning for next-generation edge-enabled industrial IoT. IEEE Trans. Netw. Sci. Eng. **9**(3), 105–117 (2022)
9. J. Lu, M. Tang, Performance analysis for IRS-assisted MEC networks with unit selection. Phys. Commun. **55**, 101869 (2022)
10. W. Xu, Z. Yang, D.W.K. Ng, M. Levorato, Y.C. Eldar, M. Debbah, Edge learning for B5G networks with distributed signal processing: semantic communication, edge computing, and wireless sensing. IEEE J. Sel. Top. Signal Process. arXiv: 2206.00422 (2023)
11. R. Zhao, C. Fan, J. Ou, D. Fan, J. Ou, M. Tang, Impact of direct links on intelligent reflect surface-aided MEC networks. Phys. Commun. **55**, 101905 (2022)
12. W. Zhou, F. Zhou, Profit maximization for cache-enabled vehicular mobile edge computing networks. IEEE Trans. Veh. Technol. **PP**(99), 1–6 (2023)
13. X. Zheng, C. Gao, Intelligent computing for WPT-MEC aided multi-source data stream. EURASIP J. Adv. Signal Process. **2023**(1) (2023) **(to appear)**
14. L. Chen, Physical-layer security on mobile edge computing for emerging cyber physical systems. Comput. Commun. **194**(1), 180–188 (2022)
15. Z. Gao, W. Hao, S. Yang, Joint offloading and resource allocation for multi-user multi-edge collaborative computing system. IEEE Trans. Veh. Technol. **71**(3), 3383–3388 (2022)
16. J. Ling, C. Gao, DQN based resource allocation for NOMA-MEC aided multi-source data stream. EURASIP J. Adv. Signal Process. **2023**(1) (2023) **(to appear)**
17. J. Ren, X. Lei, Z. Peng, X. Tang, O.A. Dobre, RIS-assisted cooperative NOMA with SWIPT. IEEE Wirel. Commun. Lett. (2023)

	Journal : **BMCOne 13634**	Dispatch : **14-5-2023**	Pages : **20**
	Article No : **1018**	☐ LE	☐ TYPESET
	MS Code :	☑ CP	☑ DISK

18. J. Li, S. Dang, M. Wen, Index modulation multiple access for 6G communications: principles, applications, and challenges. IEEE Net. (2023)
19. X. Liu, C. Sun, W. Yu, M. Zhou, Reinforcement-learning-based dynamic spectrum access for software-defined cognitive industrial internet of things. IEEE Trans. Ind. Inform. **18**(6), 4244–4253 (2022)
20. L. Zhang, C. Gao, Deep reinforcement learning based IRS-assisted mobile edge computing under physical-layer security. Phys. Commun. **55**, 101896 (2022)
21. Y. Li, L. Chen, D. Zeng, L. Gu, A customized reinforcement learning based binary offloading in edge cloud, in *26th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2020, Hong Kong, December 2–4, 2020* (2020), pp. 356–362
22. Y. Wu, C. Gao, Task offloading for vehicular edge computing with imperfect CSI: a deep reinforcement approach. Phys. Commun. **55**, 101867 (2022)
23. H. Jiang, X. Dai, Z. Xiao, A.K. Iyengar, Joint task offloading and resource allocation for energy-constrained mobile edge computing. IEEE Trans. Mob. Comput. (2022). https://doi.org/10.1109/TMC.2022.3150432
24. X. Zhang, X. Zhang, W. Yang, Joint offloading and resource allocation using deep reinforcement learning in mobile edge computing. IEEE Trans. Netw. Sci. Eng. **9**(5), 3454–3466 (2022). https://doi.org/10.1109/TNSE.2022.3184642
25. J. Lu, M. Tang, IRS-UAV aided mobile edge computing networks with constrained latency: analysis and optimization. Phys. Commun. **2023**, 101869 (2023)
26. S. Tang, X. Lei, Collaborative cache-aided relaying networks: performance evaluation and system optimization. IEEE J. Sel. Areas Commun. **41**(3), 706–719 (2023)
27. X. Liu, H. Ding, S. Hu, Uplink resource allocation for NOMA-based hybrid spectrum access in 6g-enabled cognitive internet of things. IEEE Internet Things J. **8**(20), 15049–15058 (2021)
28. F. Fang, K. Wang, Z. Ding, V.C.M. Leung, Energy-efficient resource allocation for NOMA-MEC networks with imperfect CSI. IEEE Trans. Commun. **69**(5), 3436–3449 (2021)
29. J. Ren, G. Yu, Y. He, G.Y. Li, Collaborative cloud and edge computing for latency minimization. IEEE Trans. Veh. Technol. **68**(5), 5031–5044 (2019)
30. S. Wan, R. Wisniewski, G.C. Alexandropoulos, Z. Gu, P. Siano, Special issue on optimization of cross-layer collaborative resource allocation for mobile edge computing, caching and communication. Comput. Commun. **181**, 472–473 (2022)
31. C. Kai, H. Zhou, Y. Yi, W. Huang, Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability. IEEE Trans. Cogn. Commun. Netw. **7**(2), 624–634 (2021)
32. L. Chen, X. Lei, Relay-assisted federated edge learning: performance analysis and system optimization. IEEE Trans. Commun. **PP**(99), 1–12 (2022)
33. Z. Na, Y. Liu, J. Shi, C. Liu, Z. Gao, UAV-supported clustered NOMA for 6g-enabled internet of things: trajectory planning and resource allocation. IEEE Internet Things J. **8**(20), 15041–15048 (2021)
34. J. Li, S. Dang, Y. Huang, Composite multiple-mode orthogonal frequency division multiplexing with index modulation. IEEE Trans. Wirel. Commun. (2023)
35. Q. Wang, S. Guo, J. Liu, C. Pan, L. Yang, Profit maximization incentive mechanism for resource providers in mobile edge computing. IEEE Trans. Serv. Comput. **15**(1), 138–149 (2022). https://doi.org/10.1109/TSC.2019.2924002
36. R.T. Marler, J.S. Arora, Survey of multi-objective optimization methods for engineering. Struct. Multidiscip. Optim. **26**(6), 369–395 (2004)
37. W. Feng, N. Zhang, S. Li, S. Lin, R. Ning, S. Yang, Y. Gao, Latency minimization of reverse offloading in vehicular edge computing. IEEE Trans. Veh. Technol. **71**(5), 5343–5357 (2022)
38. W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, Q. Zhu, Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing. IEEE Internet Things J. **7**(6), 5449–5465 (2020)
39. S. Li, X. Hu, Y. Du, Deep reinforcement learning and game theory for computation offloading in dynamic edge computing markets. IEEE Access **9**, 121456–121466 (2021)
40. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms. CoRR arXiv:1707.06347 (2017)
41. J. Schulman, P. Moritz, S. Levine, M.I. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings*, ed. by Y. Bengio, Y. LeCun (2016). arXiv:1506.02438
42. L. Zhang, S. Tang, Scoring aided federated learning on long-tailed data for wireless IoMT based healthcare system. IEEE J. Biomed. Health Inform. **PP**(99), 1–12 (2023)
43. L. He, X. Tang, Learning-based MIMO detection with dynamic spatial modulation. IEEE Trans. Cogn. Commun. Netw. **PP**(99), 1–12 (2023)

Publisher's Note

	Journal : **BMCOne 13634**	Dispatch : **14-5-2023**	Pages : **20**
	Article No : **1018**	☐ LE	☐ TYPESET
	MS Code :	☑ CP	☑ DISK