frontiers | Frontiers in Energy Research

TYPE Original Research
PUBLISHED 08 January 2024
DOI 10.3389/fenrg.2023.1269988

OPEN ACCESS

EDITED BY
Jun Wu,
Beijing University of Chemical Technology, China

REVIEWED BY
Zhang Xiaohong,
Dalian University, China
Gaoming Yang,
Anhui University of Science and Technology, China

*CORRESPONDENCE
Xiang Yu,
yuxiang@tzc.edu.cn

RECEIVED 31 July 2023
ACCEPTED 12 December 2023
PUBLISHED 08 January 2024

CITATION
Yang X, Yu X and Li X (2024), Computing task allocation for power Internet of Things including renewable energy sources.
Front. Energy Res. 11:1269988.
doi: 10.3389/fenrg.2023.1269988

Computing task allocation for power Internet of Things including renewable energy sources

Xianfei Yang[1], Xiang Yu[1]* and Xiang Li[2]

[1]School of Electronics and Information Engineering, Taizhou University, Taizhou, China, [2]School of Art and Design, Taizhou University, Taizhou, China

Power Internet of Things (PIoT) is the key technology to build a new power system based on new energy. Focusing on the problem that the large amount of data leads to the long computation delay of cloud computing in the operation control process of PIoT including renewable energy sources, this paper establishes a cloud-edge-end collaborative optimization calculation model for PIoT based on edge computation. Combined with data collected by a variety of smart terminal devices (STDs) arranged around the power generation equipment, the edge computation framework of PIoT is analyzed, and a computing task allocation model based on minimizing average system latency is established. The corresponding simulation model is built for simulation verification. Compared with other baseline schemes, it has been demonstrated that the average system latency of all tasks can be significantly decreased by using the proposed optimization scheme.

KEYWORDS

power Internet of Things, renewable energy, edge computation, computing task allocation, cloud edge end

1 Introduction

In order to reach the goal of achieving carbon peak by 2030 and carbon neutrality by 2060, the construction of new power systems with new energy sources as the main body continues to advance in china (Xu et al., 2021). Due to the limited distribution of traditional fossil fuels in China, renewable energy sources such as wind power and photovoltaic power generation will gradually become the main form of energy in the future power system (Xu et al., 2022). By 2020, the installed capacity of wind power and photovoltaic power has exceeded 24.3% of the total installed capacity of power generation, and the installed capacity of grid-connected renewable energy power generation has maintained rapid growth in China. The high proportion of distributed renewable energy connected to grid significantly affects the operation mode of the power grid. Because the operation state of the distribution network is complex and changeable and the control objects are diverse, higher requirements are put forward for the measurement and control level of the power system.

A high proportion of renewable energy is a key feature of future energy systems (Xu, 2019). PIoT provides an effective solution for the efficient operation of distribution networks with renewable energy. PIoT fully applies advanced information and communication technologies such as artificial intelligence and 5G to realize real-time connection between devices in the power system, so that the system state of each link of power production, transmission, and consumption can be fully perceived and controlled. Specifically, affected by natural conditions, renewable energy generation has the characteristics of intermittence and volatility. With the continuous operation of

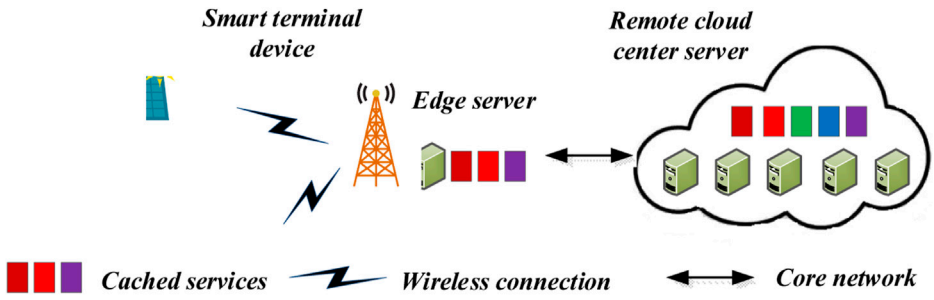

FIGURE 1
Edge computing network architecture for PIoT with renewable energy sources.

TABLE 1 Main notations.

Notation	Description
W_i	Computing tasks of *STD i*
p_i	Number of identical computing tasks of *STD i*
b_i	Amount of input data of computing tasks of *STD i*
v_i	Resource demand of computing tasks of *STD i*
$s_{i,j}$	Computing tasks of *STD i* require *service j*
p_i^{up}	Transmission power of *STD i*
h_i^{up}	Channel gain of *STD i*
B_i	Channel bandwidth of *STD i*
r_i	Transmission rate between *STD i* and the ES
c_i	Computing resources of *STD i*
c_i^e	Computing resources that the ES allocates to *STD i*
y_j	Whether or not cached *service j* at the start of the time slot *t*
y_j^s	Whether or not cached *service j* after the service cache updating
m_j	Data volume of *service j*
m^e	Storage capacity of the ES
c^e	Computing capacity of the ES

renewable energy generation grids, the energy storage capacity of lithium-ion batteries will also continue to decrease. Using a large number of STDs arranged around the power generation equipment to collect data and calculate, the output power of renewable energy generation equipment in the future can be predicted and the health status of lithium-ion batteries can be estimated, which can control the efficient and stable operation of the power grid (Huang and Wei, 2020; Zhang et al., 2020; Qin et al., 2023). PIoT connects power grid enterprises, power generation enterprises, power users, suppliers, and their equipment to achieve data sharing and ensure the intelligent management of the power grid without increasing the complexity of the physical connection structure of the power grid.

With the continuous construction of PIoT, the number of STDs has exploded, resulting in a huge amount of STDs' data that need to be stored, processed, and analyzed. The traditional data processing method is used to transmit the data to the remote cloud center server (RCCS) with strong computing power for calculation. However, due to the heterogeneous and large number of STDs in the PIoT environment, the unified upload of computing tasks to the RCCS will cause network congestion, and the RCCS is arranged at the remote end of STDs. The transmission delay generated by the STD uploading data will greatly increase the completion time of the overall task, thus affecting the real-time performance of the computing task. As an extension and supplement of cloud computing, edge computation solves this problem well. Edge computation technology enables many computing tasks to be completed by local devices without being handed over to the RCCS. The task processing is completed at the edge node close to STDs, and the edge server (ES) can respond to the STD's requests and tasks in a shorter time. However, ESs typically offer fewer computing and storage resources than RCCS. If STDs have many computing tasks, they cannot execute all of them simultaneously. Therefore, the edge computation research has focused on computing task allocation (Bi et al., 2020; Pham et al., 2020; Song et al., 2020; Wei et al., 2020; Zhu and Zhou, 2021).

Generally, a computing task allocation scheme comprises of two parts: offloading decision and resource allocation decision (Chen et al., 2022a; Chen et al., 2022b; Deng et al., 2022; Fang et al., 2022; Wang et al., 2022; Yue et al., 2022; Zhou et al., 2022; Zhou and Zhang, 2022). The former serves to determine which tasks should be performed on ESs or RCCS. The latter specifies the amount of computing and storage resources allocated to each STD by ESs or RCCS. By caching applications and their associated databases on ESs, ESs can perform tasks that require these applications, which is called service cache (Xu et al., 2018). Nowadays, in most computation offloading schemes, all services required by STDs' computing tasks have been cached by ESs. The RCCS can cache all types of services because of its massive storage capacity. However, due to its limited storage resources, there are only a few services that an ES can cache. Accordingly, a joint optimization scheme for service cache updating and computing task allocation (JOSSCUCTA) for PIoT including renewable energy sources was proposed. The contribution of this article mainly included the following three parts:

- JOSSCUCTA was proposed and established as a mathematical model of mixed-integer non-linear programming (MINLP).
- JOSSCUCTA was solved using the optimal method of outer approximation (OA).

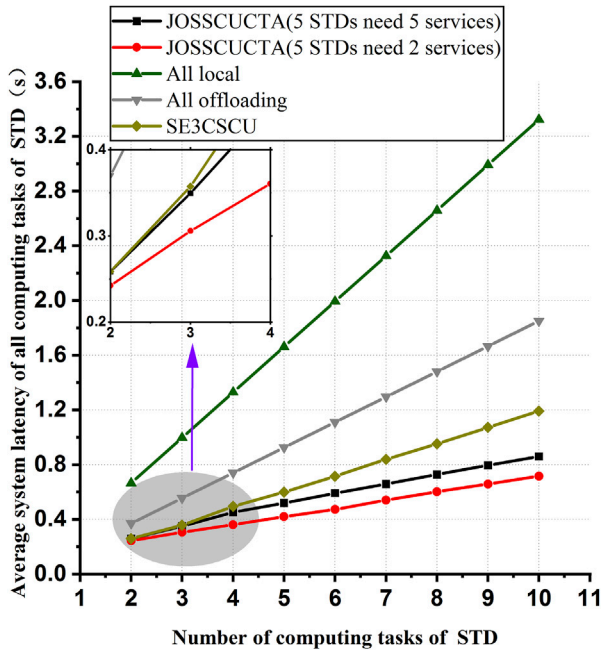

FIGURE 2
Efficiency comparison of all computing offloading schemes using different numbers of computing tasks of an STD.

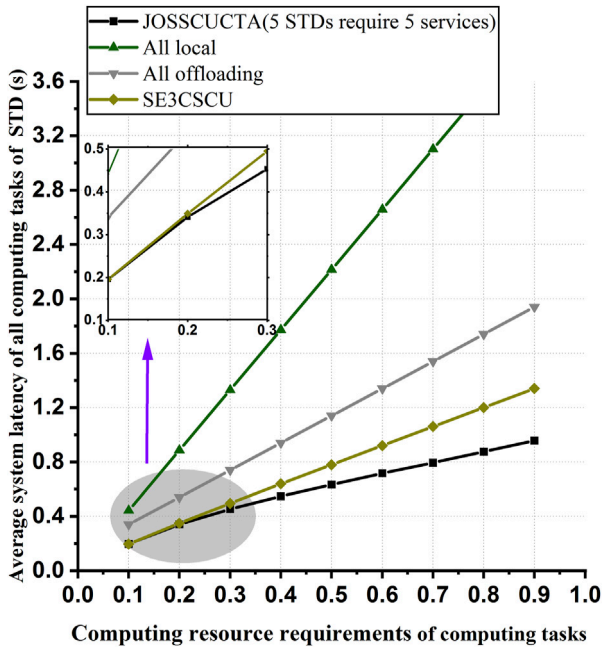

FIGURE 3
Efficiency comparison of all computing offloading schemes using different resource requirements of computing tasks.

- Comparing JOSSCUCTA with other schemes of computing task allocation, simulation results showed that it reduces average system latency of all STDs' computing tasks.

The remainder of this paper is organized as follows: a review on related work on computing task allocation is provided in Section 2. A cloud-edge-end collaboration edge computation network framework is proposed and JOSSCUCTA is proposed and established as an MINLP problem in Section 3. Section 4 provides a solution to the MINLP problem using the OA method. In Section 5, simulation results are analyzed, and conclusion is reported in Section 6 .

2 Related work

Based on optimization objectives, computing task allocation schemes can be categorized into four types: schemes with minimum delay, with minimum consumption of energy, with minimum trade-off between delay and consumption of energy, and with optimization of other performance indicators. Liu et al. (2016) adopted the Markov decision process to establish the computing task allocation scheme optimization model with minimum delay under the power constraint of terminal equipment. Based on experimental results, the scheme greatly reduced computing delay compared to the local computing and cloud computation schemes. Similarly, to obtain a computing task allocation scheme with the shortest latency in an ultra-dense network under the power constraints of terminal equipment, Chen and Hao (2018) demonstrated the computing task allocation problem as an MINLP problem using software-defined networking. According to experimental results, the method can reduce delay by 20% compared with baseline schemes. Xing et al. (2018) introduced a time-division multiple-access communication protocol under the scenario that each STD has several computing tasks. These tasks can be offloaded simultaneously to many ESs and can be completed in parallel. The performance and energy consumption of local computing and computing on ESs were analyzed under the framework of the protocol to develop an optimization model with minimum delay. Zhang et al. (2017) proposed an intelligent vehicle computing task allocation scheme model that utilized a Stackelberg game theory method to determine the task offloading decision with minimum delay, thereby maximizing the efficiency of the vehicle and edge devices. Ning et al. (2018) proposed a partial allocation scheme for computation tasks that minimizes total amount of computing delay and transmission delay when computing tasks may be separated into numerous modules. Considering that STDs communicate with ESs via wireless access networks and communication resources are limited in the edge computation system, STDs' energy consumption can be minimized by computing task allocation (Zhao et al., 2017). In order to determine the optimal c computing task allocation scheme, computer resource allocation and wireless network resource allocation were optimized simultaneously. The simulation results showed that this model had a better energy-saving effect. Because the data source is always far away from the remote cloud, computing tasks with delay constraints can only be offloaded to ESs for execution. Therefore, Guo and Liu (2018) investigated an edge computing network architecture based on a fiber–wireless network and proposed an ES and a remote cloud collaborative computing task

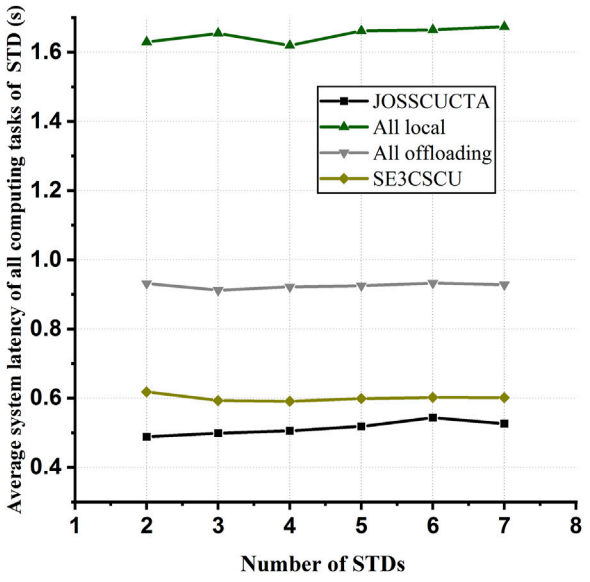

FIGURE 4
Efficiency comparison of all computing offloading schemes using different numbers of STDs.

allocation scheme with the minimum energy consumption of terminal equipment under delay constraints. STDs' energy usage can be reduced by offloading computing tasks to idle wireless devices nearby. Therefore, Cao et al. (2018) proposed an edge computing network architecture composed of STDs, help devices, and an access point device. Lan et al. (2019) developed a computing task allocation scheme that provided the optimal allocation decision while optimizing bandwidth resources, CPU frequency of STDs, and transmission power, which was modeled as an MINLP problem. Meng et al. (2019) proposed an online scheduling algorithm that optimized the allocation decision of computation tasks, the allocation of network bandwidth, and computing resources so that most tasks can be completed before the deadline. Lin and Shen (2015) aimed at optimizing the quality of the users' experience while playing interactive games. A cloud–fog system was designed to reduce delay and improve coverage using response time, network congestion, and service coverage as indicators of users' experience.

3 Model of computing task allocation

3.1 System model

As shown in Figure 1, STDs can access the nearest wireless access point (such as a mobile network base station and gateway) equipped with an ES. By the core network, the ES can offload some computing tasks to the RCCS for execution if there are many computing tasks arriving simultaneously. Meanwhile, the ES can download required services from the RCCS to update the service cache of the ES. The key notations used in this paper are summarized in Table 1.

3.2 Execution latency model of computational tasks performed by STDs locally

We consider an edge computation system for PIoT with n STDs and m services in the time slot t. Let *STD i* and *server j* represent ith STD and jth service, respectively, where $i \in N \triangleq \{1, 2, \cdots, n\}$ and $j \in M \triangleq \{1, 2, \cdots, m\}$. The computing tasks of *STD i* can be represented by a tuple $W_i \triangleq \{p_i, b_i, v_i, s_{i,j}\}$, which means *STD i* has p_i identical computing tasks and each computing task has the amount of input data b_i. Its computing resource demand is v_i, and $s_{i,j}$ indicates that computing tasks require *service j*. Suppose that x_i^l, x_i^e, and x_i^c are the amount of computing tasks performed by *STD i* locally, the ES, and the RCCS, respectively, which meet the constraint $x_i^l + x_i^e + x_i^c = p_i$. The execution latency of x_i^l computing tasks on *STD i* locally can be calculated using the following formula:

$$t_i^l = v_i x_i^l / c_i, \tag{1}$$

where c_i is the computing resource of *STD i*.

3.3 Execution latency of computational tasks performed by the ES

When the ES performs part of computing tasks of *STD i*, the execution latency includes the following three parts: task offloading latency from *STD i* to the ES, computing latency, and result feedback latency. Since the result feedback latency is relatively small, it is ignored in this article. Assuming the channel bandwidth allocated by the wireless network to *STD i* is B_i and the transmission power is p_i^{up}, the Shannon formula can be used to calculate the data transmission rate between *STD i* and the ES as follows:

$$r_i = B_i log_2\left(1 + \frac{p_i^{up} h_i^{up}}{B_i \sigma^2}\right), \tag{2}$$

where h_i^{up} and σ^2 are the channel gain and the noise power spectral density. Therefore, the transmission latency of x_i^e computing tasks is

$$t_i^{e\_tr} = b_i x_i^e / r_i. \tag{3}$$

If the ES has cached the service required by *STD i*, the computing latency depends on the computing resource demand of tasks. Therefore, the computing latency is

$$t_i^{e\_com} = v_i x_i^e / c_i^e, \tag{4}$$

where c_i^e is the computing resource that the ES allocates to *STD i*. Combining Eqs 3, 4, the latency of the ES performing x_i^e computing tasks of *STD i* can be obtained as follows:

$$t_i^e = (b_i / r_i + v_i / c_i^e) x_i^e. \tag{5}$$

3.4 Execution latency of computational tasks performed by the RCCS

If the RCCS performs part of computing tasks of *STD i*, execution latency includes the following two parts: the first part

is the transmission latency of computing tasks, including latency from *STD i* to the ES and from the ES to the RCCS. The second part is computing latency on the RCCS. The calculation formulas of transmission latency and calculation latency are, respectively, given as

$$t_i^{c\_tr} = \left(b_i/r_i + b_i/R\right)x_i^c, \tag{6}$$

$$t_i^{c\_com} = v_i x_i^c/c_i^c, \tag{7}$$

where R is the transmission rate of computing tasks from the ES to the RCCS. c_i^c is the computing resource allocated to *STD i* by the RCCS. Because it has massive computing resources, its computing resources are not necessary to be allocated to each STD optimally. The formula for calculating the latency of x_i^c computing tasks of *STD i* offloaded to the RCCS for execution is

$$t_i^c = \left(b_i/r_i + b_i/R + v_i/c_i^c\right)x_i^c. \tag{8}$$

According to Eqs 1, 5, 8, without considering service cache updating, the formula for calculating the execution latency expectation of computing tasks of *STD i* is

$$\begin{aligned} E_i &= \frac{x_i^l}{p_i}t_i^l + \frac{x_i^e}{p_i}t_i^e + \frac{x_i^c}{p_i}t_i^c \\ &= \frac{\left(x_i^l\right)^2 v_i}{p_i c_i} + \frac{\left(x_i^e\right)^2}{p_i}\left(\frac{b_i}{r_i} + \frac{v_i}{c_i^e}\right) + \frac{\left(x_i^c\right)^2}{p_i}\left(\frac{b_i}{r_i} + \frac{b_i}{R} + \frac{v_i}{c_i^c}\right). \end{aligned} \tag{9}$$

3.5 Latency of service cache updating

If STDs offload computing tasks to the ES for execution and the ES has not cached services required by these computing tasks, service need to be downloaded from the RCCS to the ES. Let the binary variable $y_j^s = 1$ denote that the ES has cached *service j* at the start of the time slot t, otherwise $y_j^s = 0$. Assume variable $y_j = 1$ indicates that the ES has cached *service j* after the service cache updating, otherwise $y_j = 0$. Then, the latency of service cache updating of the ES is

$$t^u = \sum_{j=1}^{m} y_j\left(1 - y_j^s\right)\frac{m_j}{R_0}, \tag{10}$$

where m_j is the data volume of *service j* and R_0 is the transmission rate of downloading services from the RCCS.

3.6 Mathematical description and optimization modeling of problems

Let $X \triangleq < x_1^l, \cdots, x_n^l, x_1^e \cdots, x_n^e, x_1^c \cdots, x_n^c >$ denote an offloading decision scheme, $Y \triangleq < y_1, \cdots, y_m >$ represent a service cache updating scheme, and $C^e \triangleq < c_1^e, \cdots, c_n^e >$ indicate a computing resource allocation scheme by the ES. In JOSSCUCTA, the optimization variable is $\Gamma \triangleq < X, Y, C^e >$. Because the system latency of all STDs' computing tasks includes execution latency and service cache updating latency, aiming to minimize the average system latency of all STDs' computing tasks, the optimization model is given as

$$\begin{aligned} P0\text{: } & min f\left(\Gamma\right) = \frac{1}{n}\left(\sum_{i=1}^{n} E_i + t^u\right), \\ s.t.\quad & C1\text{: } x_i^l, x_i^e, x_i^c \epsilon \{0, 1, \cdots, p_i\},\ i \in N, \\ & C2\text{: } c_i^e \geq 0,\quad i \in N, \\ & C3\text{: } y_j \in \{0, 1\},\quad j \in M, \\ & C4\text{: } y_j \geq s_{i,j} x_i^e / p_i,\ i \in N, j \in M, \\ & C5\text{: } x_i^l + x_i^e + x_i^c = p_i,\ i \in N, \\ & C6\text{: } \sum_{i=1}^{n} c_i^e \leq c^e, \\ & C7\text{: } \sum_{j=1}^{m} y_j m_j \leq m^e. \end{aligned} \tag{11}$$

Every optimization variable has a value range restricted by constraints $C1 \sim C4$. Suppose that c^e is the computing capacity of the ES. Constraint $C6$ indicates that the ES's computing capacity is greater than the sum of computing resources it allocates to all STDs. Let m^e be the storage capacity of the ES. Constraint $C7$ indicates that the total storage requirements for all services cached on the ES cannot exceed the storage capacity of the ES.

4 Solution for the optimization mode

Taking Eqs 9, 10 into the objective function of $P0$, we obtain

$$\begin{aligned} f\left(\Gamma\right) = & \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\left(x_i^l\right)^2 v_i}{p_i c_i} + \frac{\left(x_i^e\right)^2}{p_i}\left(\frac{b_i}{r_i} + \frac{v_i}{c_i^e}\right) + \frac{\left(x_i^c\right)^2}{p_i}\left(\frac{b_i}{r_i} + \frac{b_i}{R} + \frac{v_i}{c_i^c}\right)\right) \\ & + \frac{1}{n}\sum_{j=1}^{m} y_j\left(1 - y_j^s\right)\frac{m_j}{R_0}. \end{aligned} \tag{12}$$

It can be proved that if the integer variables X and Y are relaxed to real variables, Eq. 12 is a convex function. Therefore, the optimization model $P0$ is an MINLP problem which can be resolved using the OA method. This method can reduce the complexity of optimization problems by transforming MINLP into a non-linear programming (NLP) problem and a mixed-integer linear programming (MILP) problem. Because the objective function of the optimization model $P0$ is non-linear and suppose a real variable τ satisfies the constraint $f\left(\Gamma\right) \leq \tau$, the optimization model $P0$ can be transformed into an equivalent optimization model $P1$ as follows:

$$\begin{aligned} & P1\text{: } \min \tau, \\ & s.t.\ \ f\left(\Gamma\right) \leq \tau, \\ & C1 \sim C7. \end{aligned} \tag{13}$$

Let $\bar{X}^{(h)}$ and $\bar{Y}^{(h)}$ be the certain values of the integer variables X and Y, respectively, satisfying the constraint $C1 \sim C7$. By substituting them into the optimization model $P1$, an NLP problem $P2$ can be obtained:

$$\begin{aligned} & P2\text{: } \min \tau, \\ s.t.\ & \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\left(\bar{x}_i^{l(h)}\right)^2 v_i}{p_i c_i} + \frac{\left(\bar{x}_i^{e(h)}\right)^2}{p_i}\left(\frac{b_i}{r_i} + \frac{v_i}{c_i^e}\right) + \frac{\left(\bar{x}_i^{c(h)}\right)^2}{p_i}\left(\frac{b_i}{r_i} + \frac{b_i}{R} + \frac{v_i}{c_i^c}\right)\right) \\ & + \frac{1}{n}\sum_{j=1}^{m} \bar{y}_j^{(h)}\left(1 - y_j^s\right)\frac{m_j}{R_0} \leq \tau C1 \sim C7, \end{aligned} \tag{14}$$

The interior point method can be used to solve the optimization model $P2$. Suppose the values of the optimal variables are

$\Gamma^{(h)} \triangleq < \bar{X}^{(h)}, \bar{Y}^{(h)}, \bar{C}^{e(h)} >$, the upper bound of the optimization objective function value of $P0$ can be updated by calculating $f(\Gamma^{(h)})$. Using the tangent function of the non-linear convex function to replace the non-linear function itself, the OA method relaxes the non-linear constraint into a linear constraint, thereby transforming the MINLP problem into an MILP problem. Let

$$p^{(h)}(\Gamma) = f\left(\Gamma^{(h)}\right) + \nabla_{X,Y,C^e} f(\Gamma)\big|_{\Gamma=\Gamma^{(h)}}\left(\Gamma - \Gamma^{(h)}\right). \tag{15}$$

Add $p^{(h)}(\Gamma) \le \tau$ to the optimization model $P1$, and replace the non-linear constraint $f(\Gamma) \le \tau$. The optimization model $P1$ is transformed into an MILP problem.

$$\begin{aligned} &P3:\ \min \tau, \\ &s.t.\ \ p^{(h)}(\Gamma) \le \tau, \\ &C1 \sim C7. \end{aligned} \tag{16}$$

By solving the optimization model $P3$, the optimal objective function value τ and the integer optimization variable values $\bar{X}^{(h+1)}$ and $\bar{Y}^{(h+1)}$ can be obtained, and the lower bound of the optimization objective value can be updated with the function value τ. The integer optimization variable values $\bar{X}^{(h+1)}$ and $\bar{Y}^{(h+1)}$ are substituted into the optimization model $P2$ to realize the iterative calculation of the lower and upper bounds of the optimization target value of $P0$. The algorithm steps of solving the optimization model $P1$ using the OA method are shown in Algorithm 1

input: w_i, y_j^s, ε;
output: X, Y, C^e;
Initialization: $UB = \infty$, $LB = -\infty$;
$(\bar{X}^{(0)}, \bar{Y}^{(0)}) = rand\_genernation\_with\_constraints()$;
$\bar{c}^{e(0)} = optimization\_model\_P2(\bar{X}^{(0)}, \bar{Y}^{(0)})$;
$UB = f(\bar{X}^{(0)}, \bar{Y}^{(0)}, \bar{c}^{e(0)})$;
$(\bar{X}^{(1)}, \bar{Y}^{(1)}, \tau) = optimization\_model\_P3(\bar{X}^{(0)}, \bar{Y}^{(h0)}, \bar{c}^{e(0)})$.
$LB = \tau$;
$h = 1$;
While $UB - LB > \varepsilon$ **do**
$\bar{c}^{e(h)} = optimization\_model\_P2(\bar{X}^{(h)}, \bar{Y}^{(h)})$;
$UB = min(UB, f(\bar{X}^{(h)}, \bar{Y}^{(h)}, \bar{c}^{e(h)})$;
$(\bar{X}^{(h+1)}, \bar{Y}^{(h+1)}, \tau) = optimization\_model\_P3(\bar{X}^{(h)}, \bar{Y}^{(h)}, \bar{c}^{e(h)})$.
$LB = \tau$;
$h = h + 1$;
End While

Algorithm 1. The JOSSCUCTA scheme based on the OA method.

5 Performance evaluation

An evaluation of the efficiency of JOSSCUCTA is conducted in this section. The specific simulation environment is as follows: the simulation in this paper was conducted using MATLAB. The edge computation system includes one RCCS, one WiFi wireless access point, one ES, and multiple STDs. STDs are randomly distributed in a square area, and the center of the area is deployed with one wireless access point and one ES. Channel gain is $h_i^{up} = d_i^{-4}$, and d_i is the distance between wireless access point and *STD i*. c_i satisfies the uniform distribution $U \sim (0.8, 1)Gsycles/s$. $p_i^{up} = 100mv$, $B_i = 1Mbps$, and $\sigma^2 = -174dbm/Hz$. b_i satisfies the distribution $U \sim (0.5, 1)Mbps$. The ES and the RCCS computing resource are $15Gsycles/s$ and $100Gsycles/s$, respectively. The ES's storage resource is $m^e = 35Mbps$. There are five services, and the storage space required by each service obeys the distribution $U \sim (10, 20)Mbps$. The efficiency of JOSSCUCTA was verified by comparing with the following computing task allocation schemes:

- All local: STDs perform all computing tasks locally.
- All offloading: The BS or the RCCS execute all computing tasks.
- STD–edge-cloud collaborative computing without service cache updating (SE3CSCU): Use the optimization scheme proposed in this article to allocate computing tasks to devices, but set the download times of services to infinity, making cached service updating impossible.

To verify whether JOSSCUCTA can reduce the average system latency of all STDs' computing tasks when the ES has not cached services required by STDs, experiments 1 and 2 assume that services required by all STDs are not cached by the ES. The simulation scenario of experiment 1 is as follows: there are five STDs that require five servers, or that require two services. A comparison of the average system latency of different computing offloading schemes as the number of computing tasks of each STD increases is illustrated in Figure 2. As each computing task requires more and more computing resources, the average system latency of different task allocation schemes is compared, as shown in Figure 3.

Figures 2, 3 show that as the number of computing tasks on each STD increases or the computing resources required to perform each computing task increase, the average system latency of all computation offloading schemes increases. Because each STD's total computing tasks require more computing resources and each STD or the ES has constant computing resources, each computing task is allocated fewer resources. As a result, the average system latency increases.

The average system latency of JOSSCUCTA increases as the number of servers required by STDs increases because the more STDs need the same service, the earlier the service is downloaded from the RCCS to the ES. The average system latency of all local is the highest among all computation offloading schemes, which indicates that computing intensive tasks are hardly to perform locally. SE3CSCU has higher average system latency than JOSSCUCTA because without service cache updating, computing tasks of STDs can only be offloaded to the RCCS.

The simulation scenario of experiment 2 is that the server required by each STD in the edge computation system is different. As shown in Figure 4, the average system latency of JOSSCUCTA gradually increases as the number of STDs increases when there are fewer STDs in the edge computation system. This is because the ES allocates fewer computing resources to each task when STDs offload more computing tasks to it. However, with an increasing number of STDs, more computing tasks are executed on the RCCS to obtain a smaller execution delay. Thus, average system latency does not continue to increase. Meanwhile, in computing task allocation schemes of all local and all offloading, the average execution latency does not increase or decrease consistently as it depends on computing resource requirements, computing resources supplied by all STDs, and transmission latency.

To verify whether JOSSCUCTA can reduce the average system latency of all STDs' computing tasks when the ES has not cached part of the services required by STDs, the simulation scenario of

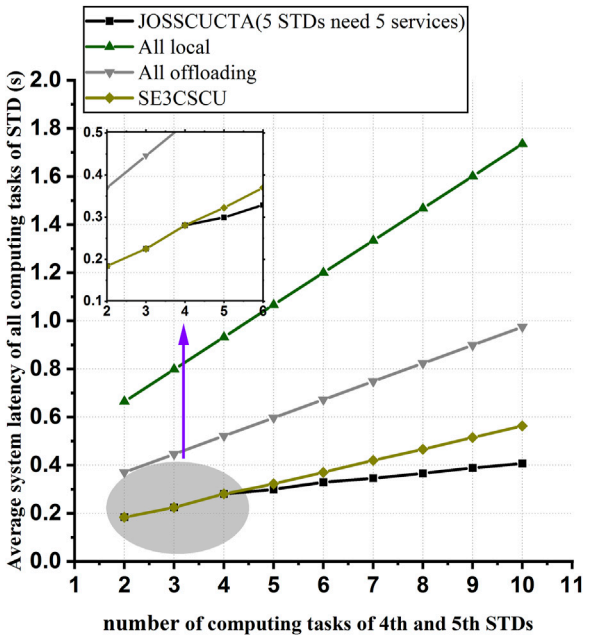

FIGURE 5
Efficiency comparison of all computing offloading schemes using different numbers of computing tasks of fourth and fifth STDs.

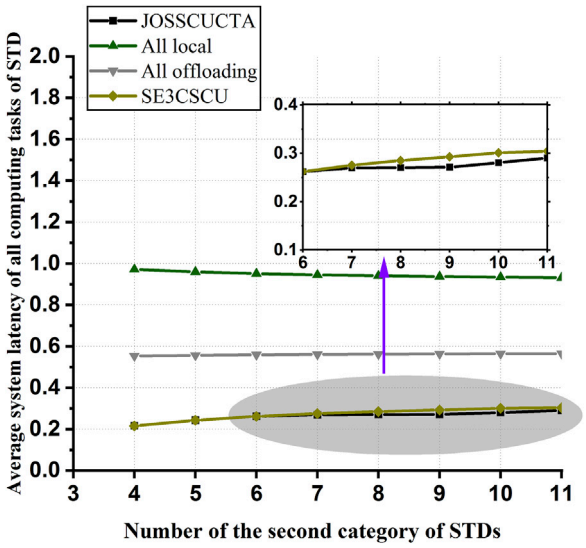

FIGURE 6
Efficiency comparison of all computing offloading schemes using different numbers of the second category of STDs.

experiment 3 is that the edge computation system contains five STDs, and each of them requires different services. The ES has cached three services which are required by first to third STDs, and there is no enough storage capacity in the ES to cache more servers. Figure 5 illustrates the comparison of all computing offloading schemes when the number of computing tasks of the fourth and fifth STDs increases.

In Figure 5, we can see even if the services cached in the ES are all required by STDs and the ES has no more storage resources to cache other services, JOSSCUCTA can reduce the average system latency. Because an STD has many computing tasks, it will increase transmission latency to offload them to the RCCS. JOSSCUCTA can download the service required by the STD to the ES, reducing the transmission latency of these computing tasks. To cache the new service, the ES needs to delete services required by other STDs with a small number of computing tasks and perform their computing tasks on the RCCS. Even though the total execution latency of these STDs has increased, it remains that the pros outweigh the cons.

The simulation scenario of experiment 4 is that there are two categories of STDs in the edge computation system. The services required by one category of STDs have been cached on the ES, and there are no more storage resources to cache other services on the ES. The services required by another category of STDs are not cached by the ES, but they are the same services. Figure 6 shows the comparison of all computing offloading schemes as the number of the second category of STDs increases.

We can see that when there are fewer STDs of the second category, the average system latency of JOSSCUCTA is the same as that of SE3CSCU, as shown in Figure 6 because server cache updating does not occur. However, as more and more STDs need the same service, JOSSCUCTA improves performance further. This is because the required service can be downloaded to the ES through service cache updating. Therefore, JOSSCUCTA can effectively adapt quickly to the changing service requirements of many STDs.

6 Conclusion

Massive heterogeneous data in PIoT with renewable energy need to be processed, and the demand for computing power and communication resources has increased dramatically. Edge computation can significantly improve the computational performance of the smart power grid. However, as the categories of computing tasks increase in PIoT and ESs are equipped with limited storage resources, ESs cannot cache all types of services required by STDs' computing tasks. Therefore, this article proposed JOSSCUCTA for PIoT with renewable energy sources and established it as an MINLP problem. Simulation experiments showed that the JOSSCUCTA scheme can effectively reduce the average system latency of all STDs' computing tasks when the ES has not cached or cached part of the services required by computing tasks.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary material; further inquiries can be directed to the corresponding author.

Author contributions

XYa: Writing–review and editing. XYu: Writing–original draft, Writing–review and editing. XL: Writing–review and editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. The paper was funded by the Science and Technology Project of Taizhou (2003gy15 and 20ny13).

Acknowledgments

The authors gratefully acknowledge the support provided by the Science and Technology Project of Taizhou (2003gy15 and 20ny13).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Bi, R., Liu, Q., Ren, J., and Tan, G. J. T. S., (2020). Utility aware offloading for mobile-edge computing. *Util. aware offloading mobile-edge Comput.* 26 (2), 239–250. doi:10.26599/TST.2019.9010062

Cao, X., Wang, F., Xu, J., Zhang, R., and Cui, S. (2018). Joint computation and communication cooperation for energy-efficient mobile edge computing. *IEEE Internet Things J.* 6 (3), 4188–4200. doi:10.1109/JIOT.2018.2875246

Chen, M., and Hao, Y. (2018). Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE J. Sel. Areas Commun.* 36 (3), 587–597. doi:10.1109/JSAC.2018.2815360

Chen, X., Zhang, J., Lin, B., Chen, Z., Wolter, K., and Min, G. (2022a). Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments. *Ieee Trans. Parallel Distributed Syst.* 33 (3), 683–697. doi:10.1109/tpds.2021.3100298

Chen, Y., Zhang, S., Jin, Y., Qian, Z., Xiao, M., Ge, J., et al. (2022b). LOCUS: user-perceived delay-aware service placement and user allocation in MEC environment. *Ieee Trans. Parallel Distributed Syst.* 33 (7), 1581–1592. doi:10.1109/tpds.2021.3119948

Deng, X., Li, J., Guan, P., and Zhang, L. (2022). Energy-efficient UAV-aided target tracking systems based on edge computing. *Ieee Internet Things J.* 9 (3), 2207–2214. doi:10.1109/jiot.2021.3091216

Fang, T., Yuan, F., Ao, L., and Chen, J. (2022). Joint task offloading, D2D pairing, and resource allocation in device-enhanced mec: a potential game approach. *Ieee Internet Things J.* 9 (5), 3226–3237. doi:10.1109/jiot.2021.3097754

Guo, H., and Liu, J. (2018). Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks. *IEEE Trans. Veh. Technol.* 67 (5), 4514–4526. doi:10.1109/TVT.2018.2790421

Huang, Q., and Wei, S. (2020). Improved quantile convolutional neural network with two-stage training for daily-ahead probabilistic forecasting of photovoltaic power. *Energy Convers. Manag.* 220, 113085. doi:10.1016/j.enconman.2020.113085

Lan, X., Cai, L., and Chen, Q. (2019). "Execution latency and energy consumption tradeoff in mobile-edge computing systems," in 2019 IEEE/CIC International Conference on Communications in China: IEEE), Changchun, China, 11-13 August 2019, 123–128.

Lin, Y., and Shen, H. (2015). "Cloud fog: towards high quality of experience in cloud gaming," in 2015 44th International Conference on Parallel Processing: IEEE, Beijing, China, September 1-4, 2015, 500–509.

Liu, J., Mao, Y., Zhang, J., and Letaief, K. B. (2016). *Delay-optimal computation task scheduling for mobile-edge computing systems*. IEEE. 1451-1455 %@ 1509018069 Available at: https://arxiv.org/abs/1604.07525.

Meng, J., Tan, H., Li, X.-Y., Han, Z., and Li, B. (2019). Online deadline-aware task dispatching and scheduling in edge computing. *IEEE Trans. Parallel Distributed Syst.* 31 (6), 1270–1286. doi:10.1109/TPDS.2019.2961905

Ning, Z., Dong, P., Kong, X., and Xia, F. (2018). A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things. *IEEE Internet Things J.* 6 (3), 4804–4814. doi:10.1109/JIOT.2018.2868616

Pham, X.-Q., Nguyen, T.-D., Nguyen, V., and Huh, E.-N. J. I. C. L. (2020). Joint service caching and task offloading in multi-access edge computing: a qoe-based utility optimization approach. *IEEE Commun. Lett.* 25 (3), 965–969. doi:10.1109/LCOMM.2020.3034668

Qin, Y., Yuen, C., Yin, X., and Huang, B. (2023). A transferable multistage model with cycling discrepancy learning for lithium-ion battery state of health estimation. *IEEE Trans. Industrial Inf.* 19 (2), 1933–1946. doi:10.1109/TII.2022.3205942

Song, F., Xing, H., Luo, S., Zhan, D., Dai, P., and Qu, R. (2020). A multiobjective computation offloading algorithm for mobile-edge computing. *IEEE Internet Things J.* 7 (9), 8780–8799. doi:10.1109/JIOT.2020.2996762

Wang, X., Ning, Z., Guo, L., Guo, S., Gao, X., and Wang, G. (2022). Online learning for distributed computation offloading in wireless powered mobile edge computing networks. *Ieee Trans. Parallel Distributed Syst.* 33 (8), 1841–1855. doi:10.1109/tpds.2021.3129618

Wei, Z., Pan, J., Lyu, Z., Xu, J., Shi, L., and Xu, J. J. C. C. (2020). An offloading strategy with soft time windows in mobile edge computing. *Comput. Commun.* 164, 42–49. doi:10.1016/j.comcom.2020.09.011

Xing, H., Liu, L., Xu, J., and Nallanathan, A. (2018). "Joint task assignment and wireless resource allocation for cooperative mobile-edge computing," in 2018 IEEE International Conference on Communications: IEEE), Kansas City, MO, USA, 20-24 May 2018.

Xu, J., Chen, L., and Zhou, P. (2018). "Joint service caching and task offloading for mobile edge computing in dense networks," in IEEE INFOCOM 2018-IEEE Conference on Computer Communications: IEEE), Honolulu, Hawaii, USA, 16-19 April 2018, 207–215.

Xu, X., Lin, Z., Li, X., Shang, C., and Shen, Q. J. I. J. o.P. R. (2022). Multi-objective robust optimisation model for MDVRPLS in refined oil distribution. *Int. J. Prod. Res.* 60, 6772–6792. doi:10.1080/00207543.2021.1887534

Xu, X., Wang, C., and Zhou, P. (2021). GVRP considered oil-gas recovery in refined oil distribution: from an environmental perspective. *Int. J. Prod. Econ.* 235, 108078. doi:10.1016/j.ijpe.2021.108078

Xu, X. J. R. P., Wei, Z., Ji, Q., Wang, C., and Gao, G. (2019). Global renewable energy development: influencing factors, trend predictions and countermeasures. *Resour. Policy* 63, 101470. doi:10.1016/j.resourpol.2019.101470

Yue, S., Ren, J., Qiao, N., Zhang, Y., Jiang, H., Zhang, Y., et al. (2022). TODG: distributed task offloading with delay guarantees for edge computing. *Ieee Trans. Parallel Distributed Syst.* 33 (7), 1650–1665. doi:10.1109/tpds.2021.3123535

Zhang, K., Mao, Y., Leng, S., Maharjan, S., and Zhang, Y. (2017). "Optimal delay constrained offloading for vehicular edge computing networks," in 2017 IEEE International Conference on Communications: IEEE), Paris, France, 21-25 May 2017.

Zhang, Y., Li, Y., and Zhang, G. (2020). Short-term wind power forecasting approach based on Seq2Seq model using NWP data. *Energy* 213, 118371. doi:10.1016/j.energy.2020.118371

Zhao, P., Tian, H., Qin, C., and Nie, G. (2017). Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. *IEEE Access* 5, 11255–11268. doi:10.1109/ACCESS.2017.2710056

Zhou, H., Jiang, K., Liu, X., Li, X., and Leung, V. C. M. (2022). Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. *Ieee Internet Things J.* 9 (2), 1517–1530. doi:10.1109/jiot.2021.3091142

Zhou, J., and Zhang, X. (2022). Fairness-Aware task offloading and resource allocation in cooperative mobile-edge computing. *Ieee Internet Things J.* 9 (5), 3812–3824. doi:10.1109/jiot.2021.3100253

Zhu, X., and Zhou, M. J. I. I. o.T. J. (2021). Multiobjective optimized cloudlet deployment and task offloading for mobile-edge computing. *IEEE Internet Things J.* 8 (20), 15582–15595. doi:10.1109/JIOT.2021.3073113