EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Intelligent computing for WPT–MEC-aided multi-source data stream

Xiangdong Zheng[1], Fusheng Zhu[2*], Junjuan Xia[1*], Chongzhi Gao[1*], Tao Cui[1] and Shiwei Lai[1]

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

[1] School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China
[2] Guangdong New Generation Communication and Network Innovative Institute (GDCNi), Guangzhou, China

Abstract

Due to its low latency and energy consumption, edge computing technology is essential in processing multi-source data streams from intelligent devices. This article investigates a mobile edge computing network aided by wireless power transfer (WPT) for multi-source data streams, where the wireless channel parameters and the characteristic of the data stream are varied. Moreover, we consider a practical communication scenario, where the devices with limited battery capacity cannot support the executing and transmitting of computational data streams under a given latency. Thus, WPT technology is adopted for this considered network to enable the devices to harvest energy from the power beacon. In further, by considering the device's energy consumption and latency constraints, we propose an optimization problem under energy constraints. To solve this problem, we design a customized particle swarm optimization-based algorithm, which aims at minimizing the latency of the device processing computational data stream by jointly optimizing the charging and offloading strategies. Furthermore, simulation results illustrate that the proposed method outperforms other benchmark schemes in minimizing latency, which shows the proposed method's superiority in processing the multi-source data stream.

Keywords: Mobile edge computing, Wireless power transfer, Particle swarm optimization, Multi-source data stream

1 Introduction

In the era of the Internet of Things (IoT) [1–4], there will be numerous intelligent devices connected to various communication systems, such as intelligent vehicles and unmanned aerial vehicle (UAV) [5–7]. However, intelligent devices are constrained by energy and latency during executing and transmitting multi-source data streams, frequently due to the material and size limitations of the intelligent devices themselves, as well as the requirements for device response efficiency [8–11]. Currently, this new pattern's critical research focuses on minimizing intelligent devices' energy consumption to prolong their lifetime, while ensuring the devices can handle as many computational data streams as possible to maintain the material and size constraints of the original devices [12–14].

Recently, cloud computing has been proposed to reduce the latency of devices to a certain extent, but the large amount of data stored in the cloud has increased the damage

of information leakage [15, 16]. Mobile edge computing (MEC) network, also known as a multi-access edge computing network, is proposed to assist computation by setting computing access points (CAPs) in the edge network, which significantly reduces the latency of the devices [17–19], and obviously increases the intelligent device's lifetime. A pivotal aspect in MEC networks is the offloading strategy, which decides how much of the computational data stream amount the devices shall delegate to the CAP via the wireless channel. In this direction, the authors in [20] improve the total computational efficiency of the devices by jointly optimizing local and data computation, the authors in [21] design a mobile offloading scheme based on reinforcement learning to optimize the computational power of the devices, and the authors of [22] studied a static system environment and obtained a static offloading strategy. However, for the dynamic environment, the channel parameters or the number of computational data streams of the devices may change. Then the problem of obtaining an optimal offloading ratio in a dynamic environment is raised. The authors in [23, 24] employed a deep Q-network (DQN) method to search for the optimized offloading strategy of the MEC network. Although DQN is highly adaptable to handle a variety of problems, its convergence is often not guaranteed, due to its significant training time and accompanied overestimation problems. To solve this problem, the authors of [25] adopted a convergence speed algorithm named particle swarm optimization (PSO) algorithm. Among the most robust algorithms in intelligent algorithms, the PSO algorithm does not require convexity and derivatives of the problem and has a fast convergence rate. The PSO algorithm can optimize the model quickly and is motivated by the birds' ingestion behaviour and imitates this behaviour. The spatial optimization issue is similar to a bird's communication space, where every bird is regarded as a particle and represents the viable solution to this case.

A critical issue for MEC networks is energy consumption, which determines how long devices can work. In some previous work studying MEC networks, for example, the authors in [26] adopted the energy consumption threshold as a constraint. However, the threshold was usually fixed as a constant, which needed to be improved. At this time, wireless power transfer (WPT) is proposed. The crucial thought behind WPT is to enable energy devices, such as power beacons (PBs), which charge intelligent devices through radio frequency (RF) signals [27]. However, WPT or MEC could not simultaneously overcome energy and latency constraints, which motivates us to integrate these two proposed technologies.

To this date, there are numerous types of research on the integration of WPT and MEC. For instance, in [28], the authors presented to consider energy harvesting (EH) in the MEC network to improve the battery energy queue. In contrast, a dynamic computational offloading algorithm optimizes the system's performance. In [29], the authors studied a nonorthogonal multiple access (NOMA)-assisted WPT–MEC network with a nonlinear EH model that maximizes the computational energy efficiency (CEE) of the system by optimizing a sequence of critical factors in the network. To satisfy the energy consumption requirements of the devices, the authors in [30] considered the PB's process of WPT to the devices as an excitation and provided an offloading strategy. The WPT can effectively satisfy the requirements of intelligent devices while extending the battery life of the devices. This article outlines the current status of methods for

offloading to the CAP in MEC and WPT. In [31], the authors proposed a wireless-powered MEC network architecture that adopted device-to-device (D2D) communication across heterogeneous networks (HetNets), allowing to offload part of the computational data streams to CAPs, and proposed an offloading decision.

Thus, the combination of WPT and MEC can provide a better simulation of latency, energy consumption or energy efficiency (EE) in a realistic environment. However, the above literature only partially considers the latency minimization problem in a latency-sensitive dynamic WPT–MEC network. Therefore, it is essential in the present time and motivates our investigation.

1.1 Contribution

This paper studies a WPT-assisted MEC network that contains multi-source data streams. In each time slot, the wireless channel parameters and the characteristics of the data stream are varied. We consider a practical communication scenario where devices with limited battery capacity cannot support the executing and transmitting of computational data streams within a specific latency. Therefore, this MEC network uses WPT to allow devices to obtain energy from the PB. Considering the devices' energy consumption and latency constraints, we propose a constrained optimization problem. To solve this problem, we design a PSO-based algorithm to minimize the device's latency in processing computational data streams by jointly optimizing the charging and offloading strategies. Simulation results indicate how the proposed method is superior to other benchmark methods in minimizing the system latency, which illustrates the method's superiority in handling multi-source data streams.

1.2 Structure

The remaining of this paper are depicted in the following sections. After the Introduction of Sect. 1, Sect. 2 presents the system model of the WPT-assisted MEC network. Then, Sect. 3 proposes a constrained optimization problem about the system's latency by jointly optimizing the charging and offloading strategies. Section 4 introduces a PSO algorithm-based scheme to optimize the system's latency. Experimental simulation results are provided in Sect. 5 to demonstrate the effectiveness of the proposed method. In Sect. 6, conclusions are delivered.

2 System model

Figure 1 depicts a system model of a WPT-assisted MEC network consisting of a CAP, a PB, and K devices denoted as $\{U_k \mid 1 \leq k \leq K\}$. Specifically, the CAP comprises a signal-receiving base station and an edge computing server, which can serve multiple devices simultaneously. In addition, each device has a rechargeable battery, and each device consists of three units: an EH unit, a computation unit, and a communication unit. We assume these devices are connected to the PB and the CAP via different wireless channels. These devices initially need to obtain energy from the RF signals transmitted by the PB and then use the received energy to execute and transmit the computational data streams from intelligent devices. In particular, we assume that the computational data streams of the devices and the parameters of the wireless channel are dynamic at each time slot. For each device, the amount of energy stored in the battery and the duration of that are constrained. In some

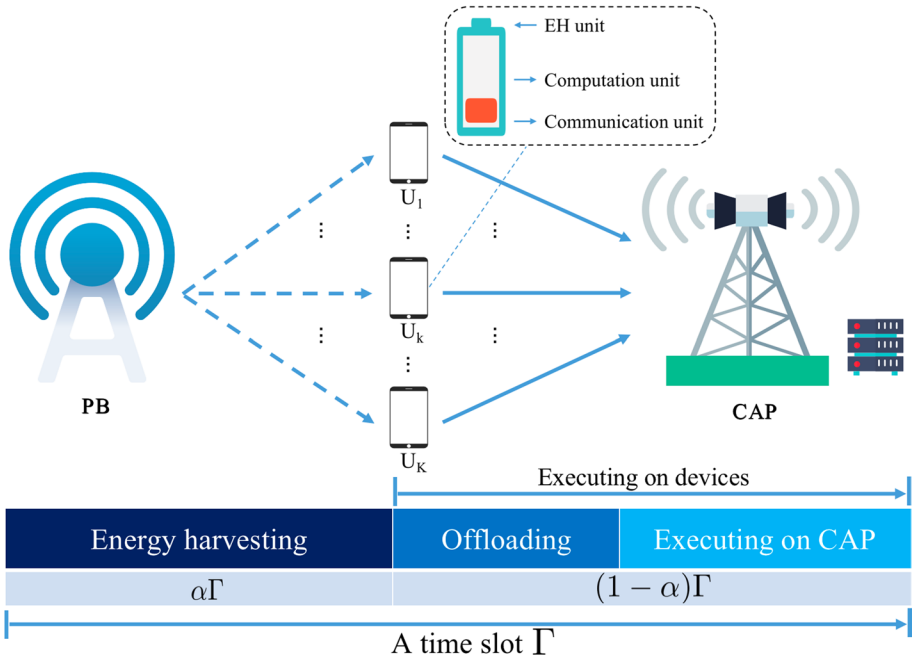

Fig. 1 System model of the WPT–MEC-aided multi-source data stream

situations, devices execute the computational data streams entirely locally is not feasible. Therefore, it is necessary to voluntarily offload part of the computational data stream to the CAP, which can overcome the device's energy consumption and latency constraints.

This paper divides a time slot duration Γ into two phases by the parameter α. In the first phase $\alpha\Gamma$, the PB emits an RF signal through the wireless channels. Then, the devices begin to harvest energy and store it in the batteries. In the second phase $(1-\alpha)\Gamma$, the devices actively offload a proportion of the computational data stream to the CAP with has more computational capacity f_{CAP}, in order to process these computational data streams faster. Specifically, within a time slot, as the parameter α of the devices increases, the devices will allocate significantly more time to the EH phase to harvest more energy. Conversely, when the parameter α decreases, the device will pay more attention to the transmission and execution phases, even though less energy is stored in the former phase. focus on transmitting and executing computational data streams, although less Therefore, the parameter α is a significant factor impacting the system's performance.

2.1 EH phase

In this phase, the PB sends an RF signal to all the devices through the wireless channel with transmission power p_t, and the continuous working time to charge of device U_k is given by

$$T_k^{\mathrm{EH}} = \alpha_k \Gamma, \tag{1}$$

where α_k denotes the duration ratio of the WPT at device U_k. We assume that $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_k, \cdots, \alpha_K]$ denotes the charging strategy. Then, we denote harvested energy of device U_k as[1]

[1] In practical scenarios, the behaviour of devices in EH mode is frequently nonlinear [32]. Therefore, the discrepancy between the linear model and nonlinear operating units causes a partial performance loss in practice, which prompts us to contemplate the nonlinear system model in future work.

$$\mathrm{EH}_k = \beta p_t \left|g_k\right|^2 T_k^{\mathrm{EH}}, \tag{2}$$

where $\beta \in (0,1)$ is the energy harvesting efficiency coefficient and $g_k \sim \mathcal{CN}(0, \lambda_k)$ is the instantaneous channel parameter from the PB to device U_k.

2.2 Transmission phase

As we presented before, part of the computational data streams at devices are offloaded to the CAP. Then, we denote the transmission rate from device U_k to the CAP as

$$r_k = b_k \log_2 \left(1 + \frac{p_k \left|h_k\right|^2}{\sigma^2}\right), \tag{3}$$

where p_k is the transmit power at device U_k, $h_k \sim \mathcal{CN}(0, \delta_k)$ denotes the instantaneous channel parameter from device U_k to the CAP, σ^2 is the variance of the additive white Gaussian noise (AWGN) at the CAP [33–35], and b_k is the wireless bandwidth at device U_k. In this environment, b_k should satisfy the following constraint,

$$\sum_{k=1}^{K} b_k = B, \tag{4}$$

where the B is the total wireless bandwidth of K devices. Therefore, from (3), the transmission latency at device U_k is given by

$$T_k^{\mathrm{trans}} = \frac{\rho_k d_k}{r_k}, \tag{5}$$

where ρ_k is the offloading ratio at device U_k, and d_k is the size of the data stream at device U_k. We assume that $\boldsymbol{\rho} = [\rho_1, \rho_2, \cdots, \rho_k, \cdots, \rho_K]$ denotes the offloading strategy. Moreover, for each time slot, we describe the characteristics of the variation with a prototypical form of the uniform distribution $\mathcal{U}(\cdot)$. Specifically, let $d_k \sim \mathcal{U}(d_{\min}, d_{\max})$, in which $d_{\min}$ and $d_{\max}$ represent the smallest and largest value of the computational data stream of the devices, respectively. In further, we can write the transmission energy consumption as

$$E_k^{\mathrm{trans}} = \frac{p_k \rho_k d_k}{r_k}. \tag{6}$$

2.3 Execution phase

We continue computing the latency and energy consumption of data stream execution at devices and the CAP. Note that the devices start to computational data streams offloading and local computational data streams calculating simultaneously after the WPT is completed. The latency and energy consumption locally calculated at device U_k are

$$T_k^{\mathrm{local}} = \frac{(1-\rho_k) d_k}{f_k}, \tag{7}$$

$$E_k^{\mathrm{local}} = \xi_k f_k^2 (1-\rho_k) d_k, \tag{8}$$

where f_k and ξ_k denote the computational capability and energy consumption coefficients of the processor chip locally at device U_k, respectively.

The execution latency of the computational data stream at the CAP from device U_k is given by

$$T_k^{CAP} = \frac{\rho_k d_k}{f_{CAP}}. \tag{9}$$

From (3)–(9), we can write the latency after WPT at device U_k as

$$T_k = \max\left\{T_k^{\text{local}}, T_k^{\text{trans}} + T_k^{CAP}\right\}. \tag{10}$$

Excessive latency in a specific environment with high latency sensitivity leads to severe problems. For example, someone requires his device to react rapidly in some situations. For this considered network, excessive latency causes the device to be unable to complete its work, resulting in accidents. Therefore, latency reduction is the target problem of the optimization issue in this case. Analogously, we can obtain the total energy consumption at the device U_k

$$E_k = E_k^{\text{local}} + E_k^{\text{trans}}. \tag{11}$$

The system also considers a computational data stream feedback phase in the practice scenario. Nevertheless, this phase's computational data stream size is smaller than the transmission phase, and the CAP has a more powerful signal-transmitting capability. Therefore, the latency of this phase is generally minimal, and the CAP has continuous energy replenishment. Consequently, we ignore the feedback latency and energy consumption in this phase.

3 Problem formulation

In this work, we focus on minimizing the system's latency in this WPT–MEC network of K devices, which is given by

$$T_{\text{total}} = \max\{T_1, T_2, \cdots, T_K\}. \tag{12}$$

The system's latency is maximal over K devices since the devices process computational data streams in parallel.

Based on this, we focus on minimizing the system's latency by jointly optimizing the charging and offloading strategies of the devices in the WPT–MEC system within the linear EH model. Hence, we formulate the latency minimization problem of this network as

$$\begin{aligned} &\min_{\{\boldsymbol{\rho},\boldsymbol{\alpha}\}} \quad T_{\text{total}} \\ s.t. \quad &C1: E_k \le \text{EH}_k, \quad \forall k \in K, \\ &C2: T_k < \Gamma - T_k^{\text{EH}}, \quad \forall k \in K, \\ &C3: 0 \le \rho_k \le 1, 0 \le \alpha_k \le 1, \quad \forall k \in K. \end{aligned} \tag{13}$$

In this problem, C1 means that the harvested energy of device U_k EH_k during the EH phase must exceed its energy consumption during the transmission phase and the execution phase. Note that each device does not necessarily use all its harvested energy at

the end of each time slot, then the remaining energy will be gathered in the battery. Constraint C2 guarantees that the total time spent by each device in the transmission phase and the execution phase do not exceed the time remaining of the system after the EH phase. Constraint C3 ensures that the WPT duration of device U_k should not exceed the total time of a time slot, and the size of the offloaded computational data streams of device U_k are not larger than the total number of computational data streams in that time slot, respectively.

Due to the maximum operation and derivative operation of the offloading ratio $\boldsymbol{\rho}$ and the duration ratio of the WPT $\boldsymbol{\alpha}$ can make the traditional methods challenging to implement. In this article, we use a PSO-based algorithm to improve the offloading and charging strategies in the considered system. Among the most robust algorithms in the intelligent algorithms, the PSO algorithm does not require convexity and derivatives of the problem and has a fast convergence rate. In the next, we will introduce the implementation of this algorithm.

4 System latency minimization-based PSO

4.1 PSO algorithm

Within this model, we optimize the offloading and charging strategies of the system latency considered by using the PSO-based algorithm. The PSO algorithm is a swarm intelligence algorithm, meaning that although the individuals in the swarm are not intelligent, the whole swarm behaves as intelligent behaviour. The process of solving the constrained optimization problem by the PSO algorithm is similar to the behaviour of a swarm of birds foraging in an environment full of traps through collaboration and information sharing among the individuals in the swarm.

In the PSO algorithm, we assume that there are J particles in the particle swarm. Generally, the choice of the number of particle swarms depends on the complexity of the optimization problem. The larger the number of swarms, the larger the search space and the faster the convergence, but it also increases the computational complexity. Therefore, it is reasonable to make a balance when considering the number of swarms for an optimization problem. Since the number of swarms used should be proportional to the complexity of the optimization problem, while more swarms are needed to solve more complex problems. Additionally, the number of swarms should be considered to the available computing resources, as more swarms require more computing power. The number of particle swarms can generally be adjusted from 10 to 200 to obtain better optimization results. When the swarm size increases to a certain level, a further increase will no longer have a significant effect [36].

Each particle includes two important attributes: position $\boldsymbol{P}$ and velocity $\boldsymbol{V}$. At each time slot, we assume that $\boldsymbol{P}_j^i$ and $\boldsymbol{V}_j^i$ denote the position and velocity of the jth particle at the ith iteration, respectively, where $\boldsymbol{P}_j^i = \{\rho_1, \cdots, \rho_K, \alpha_1, \cdots, \alpha_K\}$ denotes as a group of solutions of the constrained optimization problem (13), $\boldsymbol{V}_j^i = \{\nabla\rho_1, \cdots, \nabla\rho_K, \nabla\alpha_1, \cdots, \nabla\alpha_K\}$ denotes the position increment of the jth particle when it moves from the $i-1$th iteration to the ith iteration. Note that the velocity $\boldsymbol{V}$ does not represent the distance the particle moves per unit of time. Instead, it means the distance and direction the particle will move to the next iteration, which is a position vector. As shown in Fig. 2. Therefore, the $\boldsymbol{P}_j^i$ is updated by

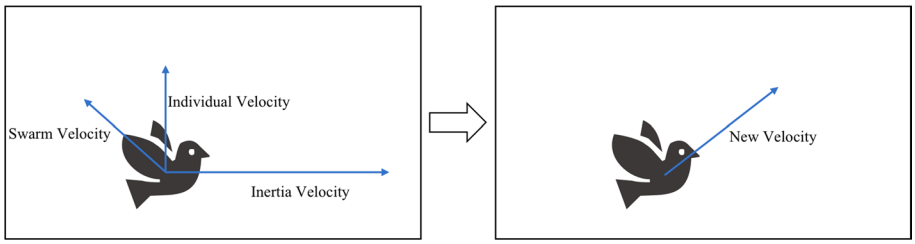

Fig. 2 Particle velocity update

$$\boldsymbol{P_j^i} = \boldsymbol{P_j^{i-1}} + \boldsymbol{V_j^i}, \tag{14}$$

where $\boldsymbol{V_j^i}$ is given by

$$\begin{aligned} \boldsymbol{V_j^i} =& \omega_i \boldsymbol{V_j^{i-1}} + c_1\xi_1\left(\boldsymbol{pbest_j} - \boldsymbol{P_j^{i-1}}\right) \\ &+ c_2\xi_2\left(\boldsymbol{gbest} - \boldsymbol{P_j^{i-1}}\right). \end{aligned} \tag{15}$$

In (15), the first term $\omega_i \boldsymbol{V_j^{i-1}}$ denotes the inertia of particle position update, and ω_i is the inertia factor of the ith iteration. In this paper, we design a dynamic inertia factor ω update method in purpose to make the PSO algorithm with higher efficiency and better result, which is given by

$$\omega_i = \omega_{\max} - (\omega_{\max} - \omega_{\min})\frac{i}{i_{\max}}, \tag{16}$$

where $\omega_{\max}$ and $\omega_{\min}$ denote the maximum and minimum values of inertia factor ω, respectively, i and $i_{\max}$ denote the current and maximum number of iterations, respectively. As the number of iterations i increases, the inertia factor will gradually decrease from $\omega_{\max}$ to $\omega_{\min}$, and the particle swarm will first exhibit a broad global search and then gradually become a detailed local search. When ω=0, the particle would lose the ability to consider its last experience.

The second and third parts $c_1\xi_1\left(\boldsymbol{pbest_j} - \boldsymbol{P_j^{i-1}}\right) + c_2\xi_2\left(\boldsymbol{gbest} - \boldsymbol{P_j^{i-1}}\right)$ represent the update of particle positions in relationship to their owns experience and cooperation with other particles in the swarm, respectively, where c_1 and c_2 are two acceleration factors, which indicate the extent of position of the next step originates from the local and global proportions, respectively. When $c_1 = 0$ and $c_2 \neq 0$, the current particle position update depends only on social experience, not on its exploration. Conversely, when $c_1 \neq 0$ and $c_2 = 0$, the current particle position update depends on its exploration, not a social experience. In addition, when $c_1 = 0$ and $c_2 = 0$, the particle only relies on the momentum of the initial velocity to update its position, which is unable to complete the exploration correctly. Conversely, when $c_1 \neq 0$ and $c_2 \neq 0$, the current particle cannot only update its position with its exploration results but also cooperate and share information with other particles in the swarm. Meanwhile, it should be noted that when the values of c_1 and c_2 are oversized, the speed of the particles will be fast. There is a distinct

possibility that the particles will cross the optimal global solution. In contrast, when the values of c_1 and c_2 are undersized, the speed of the particles will be slow, and the algorithm will be more likely to drop into the local best solution.

Moreover, ξ_1 and ξ_2 are random numbers between 0 and 1, increasing the algorithm's randomness and preventing the algorithm from falling into the local optimum as possible. $\boldsymbol{pbest_j}$ and $\boldsymbol{gbest}$ indicate the optimal solution position of the *j*th particle and the optimal solution position of the population until the current iteration, respectively. Note that the fitness function of PSO will be typified by T_{total}. After $i_{\max}$ rounds, the global optimum at *gbest* shall constitute the eventual result for this algorithm.

Algorithm 1 PSO-based Offloading Strategy and Charging Strategy

Input:
Number of particles, J;
Maximum number of iterations, i_{max};
Output:
Offloading strategy, $\boldsymbol{\rho}$;
Charging strategy, $\boldsymbol{\alpha}$;
Fitness function value, T_{total};

for each time slot **do**
  Obtain the channel condition at each device;
  Obtain the size of computational data streams at each device;
  Initialize swarm location under constraints;
  Initialize swarm velocity;
  Record $\boldsymbol{pbest_j}$;
  Record fitness function T_{total} of $\boldsymbol{pbest_j}$;
  Record $\boldsymbol{gbest}$;
  Record fitness function T_{total} of $\boldsymbol{gbest}$;
  for each iterations $i = 1 : i_{max}$ **do**
    update ω_i by (16);
    update $\boldsymbol{P}$, $\boldsymbol{V}$ and fitness function T_{total} by (14), (15), and (12);
    Abandonment of particles that do not conform to the constraint at this iteration;
    for each particles j=1 : J **do**
      if for fitness function value: $\boldsymbol{P_j^i} < \boldsymbol{pbest_j}$ **then**
        $\boldsymbol{pbest_j} = \boldsymbol{P_j^i}$;
      end if
    end for
    for each particles j=1 : J **do**
      if for fitness function value: $\boldsymbol{pbest_j} < \boldsymbol{gbest}$ **then**
        $\boldsymbol{gbest_j} = \boldsymbol{pbest_j}$;
      end if
    end for
  end for
end for

To summarize the above, we minimize the latency T_{total} by jointly optimizing the offloading strategy $\boldsymbol{\rho}$ and the charging strategy $\boldsymbol{\alpha}$ by applying the PSO algorithm. Because of the randomness, adaptability, intelligence and memory of the PSO algorithm, our proposed method can also be applied to other noisy environments. For example, Poisson noise. In Algorithm 1, we summarize this process.

4.2 Complexity analysis

We summarize the computational complexity of the proposed method. Computational complexity is one of the critical indicators of a good algorithm. The algorithm's performance deteriorates when the computational complexity is more significant, while the algorithm's performance is improved when the computational complexity is smaller. The PSO algorithm has *J* particles, and the maximum number of iterations is $i_{\max}$. We assume

that $I = i_{\max}$ and the algorithm goes through a total of T time slots. Thus, the relevant calculation complex is approximate $\mathcal{O}(I \times T \times J)$, in which the PSO algorithm performs somewhat better as the number of particles and the number of iterations increases.

We summarize notations mentioned above in Table 1.

5 Simulation results

This section describes the simulation results to demonstrate the effectiveness of the proposed PSO algorithm-based offloading and charging strategies.

5.1 Parameter setting

If not specified, we assume the wireless channels experience the Rayleigh fading model [37–40], the average channel gains λ and δ are set up as 1 for all devices. Moreover, the number of devices is fixed to 5, and the size of the computational data stream at device U_k is set as $d_k \sim \mathcal{U}(d_{\min}, d_{\max})$, where $d_{\min} = 10$Mb and $d_{\max} = 20$Mb. The total time slot is set to 100, where the duration of a time slot Γ is set to 20 s. In further, we set the total wireless bandwidth of devices to 20MHz and use an equal bandwidth-sharing model. The transmit power at each device and the variance of the AWGN are set to 1W and 0.01, respectively. Furthermore, we set the computational capacity at the CAP f_{CAP} to 10MHz. In the EH phase, the energy harvesting efficiency coefficient β is 0.88, and the total RF transmit power at the PB is 27dBw. In addition, for the PSO algorithm, the number of swarm particles is 20, the two acceleration factors c_1 and c_2 are both set to 0.4, and the maximum and minimum values of the inertia factor are set to 0.9 and 0.4, respectively. We summarize the above parameter settings in Table 2.

5.2 Simulation results

We will evaluate the proposed method against the following two benchmark schemes: (1) local computing scheme: all devices perform local computational data streams without offloading; (2) fully offloading scheme: all devices offload their computational data streams to the CAP. Note that both directions also need to be performed under the constrained conditions C1–C3.

Figure 3 illustrates the convergence of the proposed method for the different number of devices with increasing iterations, where the number of devices K ranges in $[4, 6]$ and the iterations changes in $[1, 100]$. From Fig. 3, we can observe that for different number of devices, the system latency initially decreases as the number of iterations increases, and then converges to a certain value. Specifically, the system latency with 4 devices decreases from approximately 2.24–2.19, correspondingly, the system latency with 5 devices decreases from about 2.38–2.29, and the system latency with 6 devices decreases from approximately 2.47–2.37. Moreover, we can find that the system latency increases as the value of K increases. This is because more devices increase the communication burden of the system. In further, we notice that the system latency can converge in at least ten iterations, demonstrating the superiority of the proposed method and adapting to the dynamic multi-source data stream network.

Figure 4 shows the variation of latency in the system for three schemes, in which the number of devices increases from 4 to 8. As shown in Fig. 4, we observe the

Table 1 NOTATION LIST

Notation	Definition
K	Number of devices
U_k	kth device
Γ	Duration of a time slot
α_k	Duration ratio of the WPT at device U_k
β	Energy harvesting efficiency coefficient
p_t	Transmission power at the PB
g_k	Wireless Channel parameter from the PB to device U_k
ρ_k	Offloading ratio at device U_k
r_k	Transmission rate from device U_k to the CAP
b_k	Wireless bandwidth allocate to device U_k
B	Total wireless bandwidth at devices
p_k	Transmit power at device U_k
h_k	Wireless channel parameter from device U_k to the CAP
d_k	Amount of the computational data stream at device U_k
f_k	Computation capability at device U_k
ξ_k	Energy consumption coefficient of the processor chip locally at device U_k
f_{CAP}	Computation capability at the CAP
$\mathcal{U}(\cdot)$	Uniform distribution
$d_{\min}$	The smallest value of the computational data stream
$d_{\max}$	The largest value of the computational data stream
T_k	Latency after WPT at device U_k
E_k	Energy consumption at device U_k
T_k^{trans}	Transmission latency at device U_k
E_k^{trans}	Transmission energy consumption at device U_k
E_k^{local}	Device U_k's calculated energy consumption at local
T_k^{local}	Device U_k's calculated latency at local
T_k^{CAP}	Calculated latency at the CAP
T_k^{EH}	Charge latency at device U_k
EH_k	Harvested energy at device U_k
T_{total}	System latency
J	Number of particles
$\boldsymbol{P_j^i}$	The position of the jth particle at the ith iteration
$\boldsymbol{V_j^i}$	The velocity of the jth particle at the ith iteration
ω	Inertia factor
$\boldsymbol{pbest_j}$	Optimal position at particle j
$\boldsymbol{gbest}$	Optimal position at swarm

latency of the proposed system grows with the increasing number of devices, because more devices put more burden on the system in communication. Specifically, in formulation (12), since we assume that the system latency is the maximum latency value between K devices, smaller latencies will be ignored. Moreover, the proposed method outperforms the other schemes. For example, the system latency of the local computing scheme ranges from 4.7 to 5.4, while the system latency of the fullying offloading scheme ranges from 2.4 to 3.5. By comparison, the system latency of the proposed method varies from 2.1 to 2.6. The proposed method is numerically better than both other schemes with the lowest growth rate.

Table 2 Parameter setting

Parameter	Value
Average channel gain from the PB to devices g_k	1
Average channel gain from the devices to the CAP h_k	1
Transmit power at devices p_k	1 W
Duration of a time slot Γ	20 s
Energy harvesting efficiency coefficient β	0.88
Transmit power at the PB p_t	27 dBw
Total wireless bandwidth B	20 MHz
Computation capability at the CAP f_{CAP}	10 MHz
The smallest value of the data stream $d_{\min}$	10 Mb
The biggest value of the data stream $d_{\max}$	20 Mb
Inertia factor $\omega_{\max}$	0.9
Inertia factor $\omega_{\min}$	0.4
Total time slot T	100
Variance of the AWGN at the CAP σ^2	0.01
Number of swarm particles J	20
Acceleration factors c_1 and c_2	0.4, 0.4

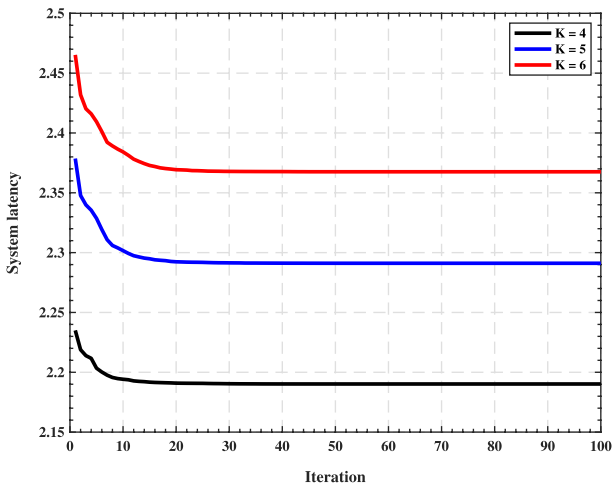

Fig. 3 Convergence of the proposed method

Figure 5 depicts the system latency versus the total communication bandwidth, in which the total system communication bandwidth increases from 20 MHz to 40 MHz. As illustrated in Fig. 5, we can see that when the total system bandwidth increases, the system latency of the fullying offloading scheme and of the proposed method show a decreasing trend. This is because the increase in the total system bandwidth results in more bandwidth allocated to each device, decreasing the computational data stream transmission latency, which facilitates reducing the computational data stream's transmission energy overhead. Specifically, the system latency of the proposed method varies from about 2.3 to 2.1, and the system latency of the fullying offloading scheme varies from about 2.7 to 2.2, which shows that the impact of

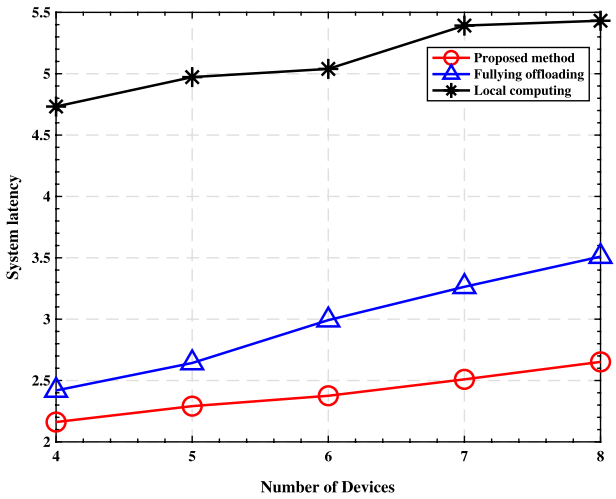

Fig. 4 System latency versus the number of devices

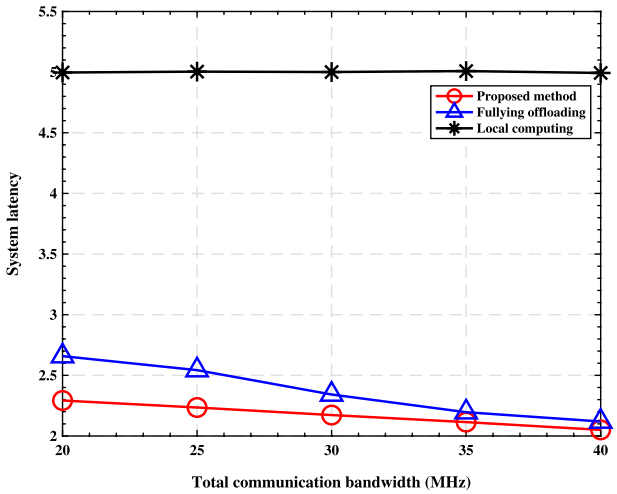

Fig. 5 System latency versus the total communication bandwidth

the increase in total system bandwidth on the fullying offloading scheme is more significant than that of the proposed method. This is because the proposed method will reserve part of the computational data streams for local computing as necessary, but the fullying offloading scheme will completely offload the computational data streams to the CAP. In contrast, the local computing scheme has no computational data stream transmission process. Thus, the system latency of the local computing scheme remains unchanged. Among the three schemes, the proposed method always has the best system latency, which also indicates the effectiveness of this proposed scheme.

Figure 6 depicts the system latency versus the smallest value of the computational data stream, which increases from 2 to 10. According to results in Fig. 6, we can find that the system latency increases as the smallest value of the computational data stream increases. Specifically, the system latency of the local computing scheme

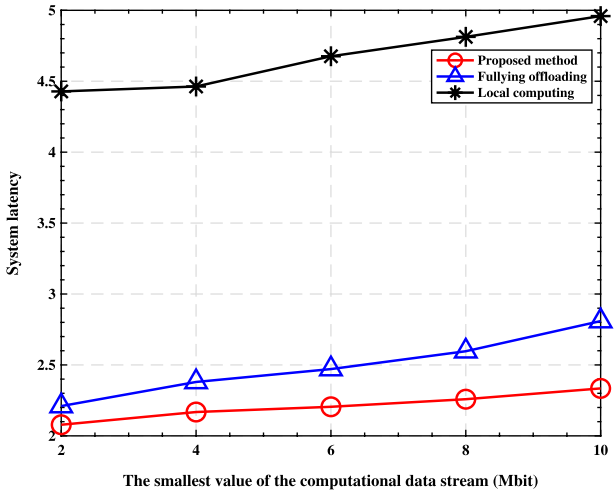

Fig. 6 System latency versus the smallest value of the computational data stream

ranges from 4.4 to 5, while the system latency of the fullying offloading scheme ranges from 2.2 to 2.8. By comparison, the system latency of the proposed method varies from 2.1 to 2.3. This is because larger computational data streams impose a bigger computational and energy consumption burdens at devices. From Fig. 6, we also note that the proposed method invariably outperforms other schemes regarding system latency because this method effectively utilizes the computation capability of both the local and the CAP. Specifically, as the computational data stream increases, our proposed method can help the device transfer part of the computational data stream to the CAP for computation when the channel conditions are favourable, which makes its system latency comparable to the fullying offloading method. Our proposed method will compute locally when the channel conditions are unfavourable, making its system latency comparable to the local computing method. Therefore, our proposed approach is always superior to the other two approaches, which can be considered as exceptions to the proposed method.

Figure 7 delineates the relation between the system latency and the computational capability at the CAP. From the results in Fig. 7, we can witness that the system latency of the fullying offloading scheme and the proposed method show a decreasing tendency as the computational capability at the CAP increases. Specifically, the system latency of the fullying offloading scheme ranges from 4.4 to 1.6, while the system latency of the proposed method varies from 3.8 to 1.4. The increasing computational capacity of the CAP leads to a more efficient execution of computational data streams by the CAP, reducing the system latency. The local computing scheme's line remains unchanged due to the lack of a transmission phase for the computational data stream in this scheme. Among the three schemes, the system latency of the proposed method is always better than the other two, which further demonstrates the method's validity.

From Fig. 3, 4, 5, 6 and 7, we can summarize the proposed method, which jointly optimizes the offloading and charging strategies, efficiently minimizes the system latency, assisting in processing multi-source data streams from intelligent devices for the considered network.

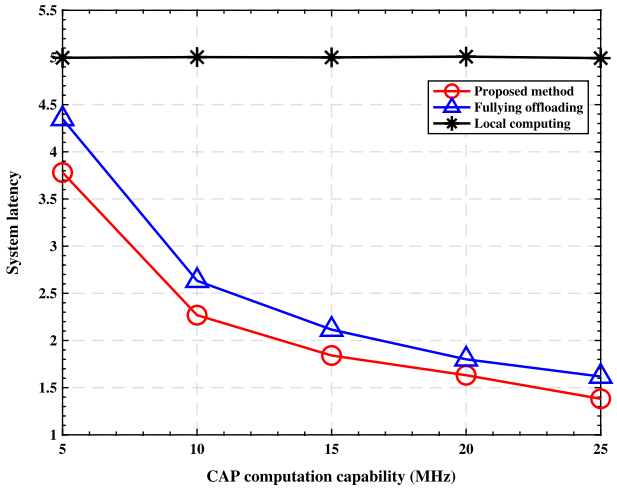

Fig. 7 System latency versus the CAP computational capability

6 Conclusion

This paper investigated a MEC network aided by WPT, which incorporated multi-source data streams. The device's wireless channel parameter and characteristic of the data stream varied in each time slot. Moreover, we considered a practical communication scenario, where the devices with limited battery capacity could not support the executing and transmitting of computational data streams within a given latency. Thus, WPT was adopted for this MEC network to allow the devices to harvest energy from the PB. Considering the device's energy consumption and latency constraints, we proposed a constrained optimization problem. To solve this problem, we designed a customized PSO-based algorithm which minimized the latency of the device processing computational data stream by jointly optimizing the charging and offloading strategies. Simulation results illustrated that the proposed method outperformed other benchmark schemes in minimizing latency, which illustrated the proposed method's superiority in processing the multi-source data stream.

Abbreviations

MEC	Mobile edge computing
WPT	Wireless power transfer
PB	Power beacon
PSO	Particle swarm optimization
IoT	Internet of things
UAV	Unmanned aerial vehicle
CAP	Computing access point
DQN	Deep Q-network
NOMA	Nonorthogonal multiple access
CEE	Computational energy efficiency
HetNets	Heterogeneous networks
D2D	Device-to-device
EE	Energy efficiency
AWGN	Additive white Gaussian noise

Author contributions

XZ proposed the model and performed the simulation experiments, FZ interpreted the experimental outcomes and checked the derivation process, JX modified the initial draft grammar issues, CG helped to complete the problem formulation, TC assisted with the Introduction, and SL contributed to the PSO algorithm. All authors read and support the final manuscript.

Funding
This work was supported by the Key-Area Research and Development Program of Guangdong Province, China (no.2019B090904014), and by Science and Technology Program of Guangzhou (No. 202201010047).

Availability of data and materials
The author promises that the data in this article are available.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interest
The authors promise that there are no competing interests related to the publication of this paper.

Received: 28 December 2022 Accepted: 27 March 2023
Published online: 04 May 2023

References

1. 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)
2. 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)
3. R. Zhao, M. Tang, Profit maximization in cache-aided intelligent computing networks. Phys. Commun. **99**, 1–10 (2022)
4. 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)
5. 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)
6. 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. Topics Signal Process. **abs/2206.00422** (2023)
7. 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. Inf. **18**(6), 4244–4253 (2022)
8. L. Zhang, C. Gao, Deep reinforcement learning based IRS-assisted mobile edge computing under physical-layer security. Phys. Commun. **55**, 101896 (2022)
9. W. Zhou, C. Li, M. Hua, Worst-case robust MIMO transmission based on subgradient projection. IEEE Commun. Lett. **25**(1), 239–243 (2021)
10. W. Zhou, X. Lei, Priority-aware resource scheduling for UAV-mounted mobile edge computing networks. IEEE Trans. Veh. Tech. **99**, 1–6 (2023)
11. 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)
12. L. Chen, X. Lei, Relay-assisted federated edge learning: performance analysis and system optimization. IEEE Trans. Commun. **99**, 1–12 (2022)
13. 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)
14. 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)
15. M.M. Sadeeq, N.M. Abdulkareem, S.R. Zeebaree, D.M. Ahmed, A.S. Sami, R.R. Zebari, Iot and cloud computing issues, challenges and opportunities: a review. Qubahan Acad. J. **1**(2), 1–7 (2021)
16. 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. Inf. **17**(5), 3391–3400 (2021)
17. J. Lu, M. Tang, Performance analysis for IRS-assisted MEC networks with unit selection. Phys. Commun. **55**, 101869 (2022)
18. J. Ling, C. Gao, DQN based resource allocation for NOMA-MEC aided multi-source data stream. to appear in EURASIP J. Adv. Signal Process. **2023**(1) (2023)
19. L. Chen, Physical-layer security on mobile edge computing for emerging cyber physical systems. Comput. Commun. **194**(1), 180–188 (2022)
20. H. Sun, F. Zhou, R.Q. Hu, Joint offloading and computation energy efficiency maximization in a mobile edge computing system. IEEE Trans. Veh. Technol. **68**(3), 3052–3056 (2019)
21. L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, Y. Zhang, Reinforcement learning-based mobile offloading for edge computing against jamming and interference. IEEE Trans. Commun. **68**(10), 6114–6126 (2020)
22. X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. **24**(5), 2795–2808 (2015)

23. Y. Wu, C. Gao, Task offloading for vehicular edge computing with imperfect CSI: a deep reinforcement approach. Phys. Commun. **55**, 101867 (2022)
24. Y. Wu, C. Gao, Intelligent resource allocation scheme for cloud-edge-end framework aided multi-source data stream. to appear in EURASIP J. Adv. Signal Process. **2023**(1) (2023)
25. W. Zhou, F. Zhou, Profit maximization for cache-enabled vehicular mobile edge computing networks. IEEE Trans. Veh. Tech. **99**, 1–6 (2023)
26. Y. Guo, R. Zhao, S. Lai, L. Fan, X. Lei, G.K. Karagiannidis, Distributed machine learning for multiuser mobile edge computing systems. IEEE J. Sel. Top. Signal Process. (2022)
27. Z. Zhou, C. Zhang, J. Wang, B. Gu, S. Mumtaz, J. Rodriguez, X. Zhao, Energy-efficient resource allocation for energy harvesting-based cognitive machine-to-machine communications. IEEE Trans. Cogn. Commun. Netw. **5**(3), 595–607 (2019)
28. W. Zhou, L. Xing, J. Xia, L. Fan, A. Nallanathan, Dynamic computation offloading for MIMO mobile edge computing systems with energy harvesting. IEEE Trans. Veh. Technol. **70**(5), 5172–5177 (2021)
29. L. Shi, Y. Ye, X. Chu, G. Lu, Computation energy efficiency maximization for a NOMA-based WPT-MEC network. IEEE Internet Things J. **8**(13), 10731–10744 (2021)
30. E. Mustafa, J. Shuja, A.I. Jehangiri, S. Din, F. Rehman, S. Mustafa, T. Maqsood, A.N. Khan et al., Joint wireless power transfer and task offloading in mobile edge computing: a survey. Clust. Comput. **25**(4), 2429–2448 (2022)
31. B. Li, Z. Fei, J. Shen, X. Jiang, X. Zhong, Dynamic offloading for energy harvesting mobile edge computing: architecture, case studies, and future directions. IEEE Access **7**, 79877–79886 (2019)
32. G. Lu, L. Shi, Y. Ye, Maximum throughput of TS/PS scheme in an AF relaying network with non-linear energy harvester. IEEE Access **6**, 26617–26625 (2018)
33. L. He, X. Tang, Learning-based MIMO detection with dynamic spatial modulation. IEEE Trans. Cogn. Commun. **99**, 1–12 (2023)
34. S. Tang, Dilated convolution based CSI feedback compression for massive MIMO systems. IEEE Trans. Vehic. Tech. **71**(5), 211–216 (2022)
35. J. Li, S. Dang, M. Wen, Index modulation multiple access for 6G communications: principles, applications, and challenges. IEEE Net. (2023)
36. M. Jain, V. Saihjpal, N. Singh, S.B. Singh, An overview of variants and advancements of PSO algorithm. Appl. Sci. **12**(17), 8392 (2022)
37. L. Zhang, S. Tang, Scoring aided federated learning on long-tailed data for time-varying IoMT based healthcare system. IEEE J. Biomed. Health Inform. **99**, 1–12 (2023)
38. J. Ren, X. Lei, Z. Peng, X. Tang, O.A. Dobre, Ris-assisted cooperative NOMA with SWIPT. IEEE Wireless Communications Letters (2023)
39. 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)
40. S. Tang, X. Lei, Collaborative cache-aided relaying networks: performance evaluation and system optimization. IEEE J. Sel. Areas Commun. **41**(3), 706–719 (2023)

Publisher's Note