Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3365279, 11 pages
https://doi.org/10.1155/2022/3365279

Research Article

Machine Learning-Based Two-Stage Task Offloading Optimization for Power Distribution Internet of Things

Chenzidu He, Yuanyuan Wu, Mengxing Huang, Siling Feng, and Feng Shu

College of Information and Communication Engineering, Hainan University, Haikou 570228, China

Correspondence should be addressed to Yuanyuan Wu; wyuanyuan82@163.com

Received 23 July 2022; Revised 20 September 2022; Accepted 5 October 2022; Published 28 October 2022

Academic Editor: Wenjuan Li

The increase in the number of services in the power distribution grid leads to a massive increase in task data. Power distribution internet of things (PDIoT) is the specific application of internet of things (IoT) in the power distribution grid. By deploying a large number of PDIoT devices, the voltage, active power, reactive power, and harmonic parameters are collected to support distribution grid services such as fault identification and status detection. Therefore, PDIoT utilizes massive devices to collect and offload tasks to the edge server through 5G network for real-time data processing. However, how to dynamically select edge servers and channels to meet the energy-efficient and low-latency task offloading requirements of PDIoT devices still faces several technical challenges such as task offloading decisions coupling among devices, unobtainable global state information, as well as interrelation of various quality of service (QoS) metrics such as energy efficiency and delay. To this end, we firstly construct a joint optimization problem to maximize the weighted difference between energy efficiency and delay of devices in PDIoT. Then, the joint optimization problem is decomposed into a large-timescale server selection problem and a small-timescale channel selection problem. Next, we propose an ML-based two-stage task offloading algorithm, where the large-timescale problem is solved by two-side matching in the first stage, and the small-timescale problem is solved by adaptive ε-greedy learning in the second stage. Finally, simulation results show that compared with the task offloading delay-first matching algorithm and the matching theory-based task offloading strategy, the proposed algorithm performs superior in terms of energy efficiency and delay.

1. Introduction

With the rapid development of new power systems, the number of services in the power distribution grid has gradually increased, resulting in a massive increase in task data. Power distribution internet of things (PDIoT) is the specific application of internet of things (IoT) in the power distribution grid. By deploying a large number of PDIoT devices, the voltage, active power, reactive power, and harmonic parameters are collected to support power distribution grid services such as fault location and status detection [1]. For example, for the status estimation service, task data such as real-time power and voltage information are collected to supplement the load curve data, load forecast data, and meter reading data. By offloading the task data to the server for processing, the real-time load of power distribution grid can be obtained. However, traditional cloud computing with a long data transmission distance to the cloud server results in high delay, large energy consumption, and severe congestion [2, 3]. Edge computing integrated with 5G provides a new paradigm shift for real-time computing services, where PDIoT devices offload data to nearby edge servers to reduce delay and energy consumption [4–6].

Task offloading is a key enabler to realize efficient edge computing for PDIoT. On the one hand, the devices need to select the optimal one among the deployed edge servers. On the other hand, due to the spectrum shortage, the devices need to dynamically select channels according to available spectrum resources. Meanwhile, PDIoT services have strict requirements on energy efficiency and delay performances [7]. For instance, the delay requirement of control services is millisecond level, while the monitoring devices collect and transmit massive data to improve energy efficiency under limited battery capacity [8]. Therefore, how to achieve energy-efficient and low-latency task offloading by jointly

optimizing server and channel selection remains an open issue. The joint optimization problem still faces the following challenges.

First, the task offloading decisions among massive devices are coupled with each other. Meanwhile, server selection needs to be optimized according to the change of server computing resources, and channel selection needs to be optimized according to the change of channel state. Since the change of server computing resources is not in the same timescale as that of channel state, task offloading needs to be optimized in different timescales. Particularly, the large-timescale server selection is optimized in the first stage, while the small-timescale channel selection is optimized in the second stage. Therefore, two-stage task offloading problem is constructed. Second, the wireless channels are interfered by electromagnetic interference in PDIoT, and face channel fading caused by multipath transmission. It is not feasible to obtain global state information (GSI) for task offloading. Finally, the optimization of energy efficiency, transmission delay, and processing delay are coupled with each other tightly, leading to a more complex optimization problem.

Task offloading has gained considerable attention from both academia and industry. In [9], Mitsis et al. proposed a data offloading framework for UAV-assisted multiaccess edge computing systems based on resource pricing and user risk perception. A usage-based pricing mechanism for users was introduced to utilize the computing power of MEC server. In [10], Chen et al. proposed an alternating minimization algorithm to achieve energy-optimal fog computing offloading by jointly optimizing offloading ratio, transmission power, local CPU computation speed, and transmission time. In [11], Maray et al. surveyed the latest research on task offloading from the aspects of offloading mechanism, offloading granularity, and offloading technology. Various task offloading mechanisms and optimization methods in different environments were discussed. In [12], Mustafa et al. divided the computation offloading into four categories, i.e., static, dynamic, full, and partial offloading, and compared the existing research from seven aspects, i.e., contribution, computation offloading, energy/battery lifetime, resource/task scheduling, cooperation, user fairness, and transmission/computation latency. In [13], Wu et al. proposed an energy-efficient dynamic task offloading (EEDTO) algorithm to control the computation and communication costs for different types of applications and dynamic changes in the wireless environment. However, the above works neglected the coupling of task offloading decisions among massive devices and cannot solve access conflicts among devices. Task offloading problem can be constructed as a two-side matching problem to obtain stable task offloading strategies and cope with the access conflicts among devices.

Matching theory provides an effective approach to solve the two-side matching problem by defining the preferences of matching subjects to address access conflicts among devices, which has been widely used in solving task offloading problems [14, 15]. In [16], Shi et al. proposed a two-side matching-based server selection algorithm to maximize the efficiency of device-to-device content sharing. In [17], Zhou et al. proposed a task offloading algorithm based on vehicle-device matching to maximize the utility function of the base station (BS). In [18], Abedin et al. proposed a two-side matching game to solve the problem of server selection, aiming to maximize the efficiency of user resource allocation. In [19], Wang et al. considered the impact of channel selection on task offloading, and proposed a matching-based channel selection algorithm to minimize the total energy consumption. The above works used matching theory to solve task offloading conflicts among devices, but they rely on the perfect GSI such as server states and channel states, which cannot be applied to scenarios where the global information changes rapidly and is unknown. Moreover, the above works do not consider the multitimescale joint optimization of server selection and channel selection, and the establishment of the preference list is influenced by the optimization results of other dimensions.

To solve the task offloading problem under incomplete GSI, machine learning (ML) has been applied to intelligent task offloading decision making. ML includes deep learning (DL), reinforcement learning (RL), deep reinforcement learning (DRL), and support vector machine (SVM). In [20], Jehangiri et al. proposed a mobility prediction and offloading framework that offloads computationally intensive tasks to predicted user locations using artificial neural networks. In [21], Zhou et al. proposed a task offloading strategy based on SVM, which minimizes energy consumption by optimizing clock frequency control, transmission power allocation, as well as offloading and receive power allocation strategies in edge computing scenarios. In [22], Wu et al. proposed a distributed DL-driven task offloading (DDTO) algorithm to jointly optimize the system utility and bandwidth allocation. In [23], Qu et al. proposed a deep metareinforcement learning-based offloading (DMRO) algorithm to solve the problem of limited computing resources of IoT devices and improve task processing efficiency. In [24], Chen et al. proposed a cloud-edge collaborative mobile computing offloading (DRL-CCMCO) algorithm based on DRL to solve the joint optimization problem of execution delay and energy consumption. The above works adopted ML algorithms to optimize task offloading decisions and further improve task offloading performances. However, the above algorithms have high computation complexity and high requirements for CPU performance, while PDIoT devices with limited CPU put forward lightweight requirements for the algorithm. Therefore, the above algorithms are not suitable for the scenario mentioned in the article. RL, as an important branch of ML, has low computation complexity, which can meet the needs of lightweight task offloading [25]. The task offloading problem can be regarded as a multi-armed bandit (MAB) problem and solved by RL [26]. ϵ-greedy learning algorithm is a low-complexity RL algorithm that can balance the tradeoff between exploration and exploitation through the adjustment of ϵ. In [27], Li et al. proposed an interference-aware RL algorithm to solve the joint problem of multichannel selection and data scheduling. In [28], Talekar and Terdal proposed a solution for optimal channel selection and routing and applied RL to select the best channel for routing. However, these works

do not take into account the coupling between server selection and channel selection and cannot dynamically adjust the learning parameters according to the dynamic and complex communication environment to improve learning performance.

Motivated by the aforementioned challenges, we firstly construct a two-stage task offloading problem, including the server selection in the first stage and the channel selection in the second stage, which have different timescales. The objective is to maximize the weighted difference between energy efficiency and delay through joint optimization of server selection and channel selection, considering the influence of electromagnetic interference and stringent quality of service (QoS) constraint. Then, we propose an ML-based two-stage task offloading algorithm. Specifically, a two-side matching-based server selection algorithm is proposed to obtain large-timescale device-edge stable matching. For the channel selection in the second stage, we propose an adaptive ε-greedy learning algorithm to dynamically learn optimal channel selection strategies. The main contributions of this paper are summarized as follows.

(i) *Energy-Efficient and Low-Latency Task Offloading*. Since the optimization of energy efficiency and delay are coupled, we construct the weighted difference between energy efficiency and delay to achieve the joint optimization of different QoS metrics

(ii) *Two-Stage Task Offloading*. We propose a two-side matching-based server selection algorithm and an adaptive ε-greedy learning algorithm to optimize large-timescale server selection in the first stage and small-timescale channel selection in the second stage under incomplete GSI

(iii) *Extensive Performance Evaluation*. Compared with two advanced algorithms, simulation results demonstrate that the proposed algorithm has superior performance of energy efficiency, transmission delay, and processing delay

The rest of the paper is organized as follows. Section 2 demonstrates the system model. Section 3 presents the ML-based two-stage task offloading algorithm for PDIoT. The simulation results are shown in Section 4 to verify the effectiveness of the proposed algorithm. Section 5 concludes the paper.

2. System Model

The considered task offloading scenario of PDIoT is shown in Figure 1, which consists of multiple BSs and PDIoT devices. Each BS is equipped with an edge server to provide overlapping communication coverage and computing resources for devices. Each device needs to offload its task data to an edge server through a BS for processing, aiming to reduce delay and improve energy efficiency. There are I PDIoT devices, J edge servers, and N channels. The sets are $\mathcal{M} = \{m_1,\cdots,m_i,\cdots,m_I\}$, $\mathcal{S} = \{s_1,\cdots,s_j,\cdots,s_J\}$, and $\mathcal{C} = \{c_1,\cdots,c_n,\cdots,c_N\}$, respectively.

We adopt a two-timescale model with period and slot [29]. The large timescale is period, and the small timescale is slot. There are L equal periods, which are large timescales, and the set is $\mathcal{L} = \{1, 2,\cdots,l,\cdots,L\}$. Each large timescale contains T_0 small timescales, i.e., slots, and the set of slots in the l-th period is $\mathcal{T}_l = \{(l-1)T_0 + 1, (l-1)T_0 + 2,\cdots,lT_0\}$. The total number of slots is T, i.e., $T = T_0L$, and the set is $\mathcal{T} = \{1, 2,\cdots,t,\cdots,T\}$. Task offloading includes two stages, i.e., large-timescale server selection and small-timescale channel selection. The server selection variable of the device m_i towards s_j in the l-th period is defined as $z_{i,j}(l) = \{0, 1\}$. $z_{i,j}(l) = 1$ indicates m_i select s_j, and $z_{i,j}(l) = 0$ otherwise. Define the quota of s_j as q_j, which represents the maximum number of devices that can be served by s_j in each period. The channel selection variable is defined as $x_{i,n}(t) = \{0, 1\}$. $x_{i,n}(t) = 1$ indicates that m_i selects c_n for data transmission, and $x_{i,n} = 0$ otherwise. An example of two-stage task offloading is shown in Figure 1. m_1 selects s_2 to offload data in the first stage, and selects c_2 for data transmission in the second stage. The main notation used in this paper is given in Table 1.

2.1. Transmission Model. Based on orthogonal frequency division multiplexing (OFDM), each device selects an orthogonal channel to offload tasks in each slot. The data transmission rate from m_i to s_j on c_n in the t-th slot is given by

$$R_{i,j,n}(t) = B\log_2\left[1 + SINR_{i,j,n}(t)\right], \tag{1}$$

where B is channel bandwidth. $SINR_{i,j,n}$ is the signal-to-interference-plus-noise ratio (SINR), which is given by

$$SINR_{i,j,n}(t) = \frac{P^{TX}g_{i,j,n}(t)}{\sigma^2 + \xi_{i,j,n}}, \tag{2}$$

where P^{TX} and σ^2 represent transmission power and noise power. $g_{i,j,n}(t)$ is the channel gain between m_i and s_j on c_n in the t-th slot. $\xi_{i,j,n}$ is the electromagnetic interference power.

2.2. Delay Model. Denoting the total computing resources of s_j as $\psi_j(t)$, the computing resources allocated by s_j to m_i in the t-th slot is given by

$$\alpha_j(t) = \frac{\psi_j(t)}{q_j}. \tag{3}$$

Denote the size of offloaded data from m_i in the t-th slot is denoted as $U_i(t)$. Then, the transmission delay of m_i offloading data to s_j on c_n, and the processing delay required by s_j to process the offloaded data of m_i in the t-th slot are

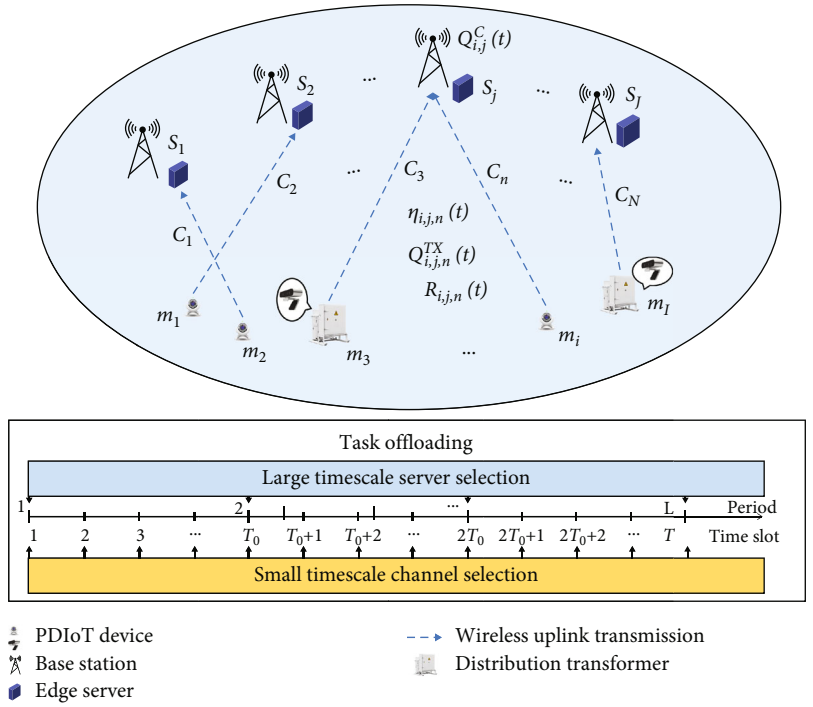

Figure 1: The task offloading scenario of PDIoT.

Table 1: Main notation table.

Notation	Meaning
I	The number of PDIoT devices
J	The number of edge servers
N	The number of channels
L	The number of periods
T	The number of slots
$\mathcal{M}$	The set of PDIoT devices
$\mathcal{S}$	The set of edge servers
$\mathcal{C}$	The set of channels
$\mathcal{L}$	The set of periods
$\mathcal{T}_l$	The set of slots in the l-th period
$\mathcal{T}$	The set of slots
$z_{i,j}(l)$	The server selection variable
$x_{i,n}(t)$	The channel selection variable
B	The channel bandwidth (MHz)
P^{TX}	The transmission power (W)
σ^2	The noise power (dBm/Hz)
$g_{i,j,n}(t)$	The channel gain
$\xi_{i,j,n}$	The electromagnetic interference power (dBm/Hz)
$\psi_j(t)$	The total computing resources (GHz)
f_j	The CPU cycles (cycle/bit)
P_0	The circuit power of device operation (W)

given by

$$Q^{TX}_{i,j,n}(t) = \frac{U_i(t)}{R_{i,j,n}(t)},$$
$$Q^{C}_{i,j}(t) = \frac{f_j U_i(t)}{\alpha_j(t)}, \tag{4}$$

where f_j (cycle/bit) is the CPU cycles required by s_j to process one bit of data.

The result feedback delay is negligible compared with transmission delay and processing delay [30]. Therefore, the total delay is the sum of transmission delay and processing delay, which is given by

$$Q_{i,j,n}(t) = Q^{TX}_{i,j,n}(t) + Q^{C}_{i,j}(t). \tag{5}$$

2.3. Energy Efficiency Model. The transmission energy consumption of m_i in the t-th slot is given by

$$E^{TX}_{i,j,n}(t) = P^{TX} Q^{TX}_{i,j,n}(t). \tag{6}$$

The operation energy consumption of m_i is given by

$$E^{O}_{i,j,n}(t) = P_0 Q^{TX}_{i,j,n}(t), \tag{7}$$

where P_0 is the circuit power of device operation.

The total energy consumption is the sum of transmission energy consumption and operation energy consumption,

Input: $\mathcal{M}$, $\mathcal{S}$, $\mathcal{C}$.
Output: $\{z_{i,j}(l)|\forall m_i \in \mathcal{M}, \forall s_j \in \mathcal{S}, \forall l \in \mathcal{L}\}$ and $\{x_{i,n}(t)|\forall m_i \in \mathcal{M}, \forall c_n \in \mathcal{C}, \forall t \in \mathcal{T}\}$
Phase 1. Initialization
  Initialize $k_{i,n}(1) = 0, \forall m_i \in \mathcal{M}, \forall c_n \in \mathcal{C}$.
For $l = 1 : L$ **do**
  Phase 2. Large-Timescale First-Stage Server Selection
  Step 1:
  Initialize $\phi = \varnothing$, $\Theta = \mathcal{M}$, and $\Gamma = \mathcal{S}$.
  Step 2:
    m_i and s_j calculate the preference values $\tilde{\eta}_{i,j}(l)$ and $\tilde{Q}_{i,j}(l)$ based on (11) and (12) and establish the preference lists $\mathcal{F}_i$ and $\mathcal{F}_j$.
  Step 3:
  While $\Omega \neq \varnothing$ and $\mathcal{F}_i \neq \varnothing$ **do**
    m_i proposes to its most preferred server based on $\mathcal{F}_i$.
    For $s_j \in \Gamma$ **do**
      If the sum of temporary matches and new proposals for s_j is less than quota q_j **then**
        Temporarily match s_j with the devices, update $z_{i,j}(l)$, and remove the matched devices from Θ.
      else
        Temporarily match s_j with its most preferred q_j devices and update $z_{i,j}(l)$. Remove matched devices from Θ and add unmatched devices into Θ. Unmatched devices remove s_j from $\mathcal{F}_i$.
      End if
      If the sum of matches for s_j is equal to q_j **then**
        Remove s_j from Γ.
      End if
    End for
  End while
  For $t = (l-1)T_0 + 1 : lT_0$ **do**
    Phase 3. Small-Timescale Second-Stage Channel Selection
    m_i makes the action decision $a_i(t)$ based on (18).
    m_i calculates $r_{i,n}(t)$ and $\bar{r}_{i,n}(t)$ based on (14) and (15)
    Update $\bar{r}_K(t+1)$ and $\boldsymbol{\epsilon}_{t+1}$ based on (16) and (17).
  End for
End for

Algorithm 1: ML-based two-stage task offloading optimization algorithm.

Table 2: Simulation parameters.

Parameter	Value	Parameter	Value
I	120	J	3
N	50	τ	0.5
B	5 MHz	σ^2	−170 dBm/Hz
P^{TX}	0.4 W	P_0	0.1 W
Ψ_j	60 – 180 GHz	U_i	0.8 – 1.2 Mbits
f_j	800 cycle/bit	V	0.5 – 3

which is given by

$$E_{i,j,n}(t) = \left(P^{TX} + P_0\right) Q^{TX}_{i,j,n}(t) = \left(P^{TX} + P_0\right) \frac{U_i(t)}{R_{i,j,n}(t)}. \quad (8)$$

Based on [31], we define energy efficiency as the amount of data transmitted per unit of energy and per unit of bandwidth (bit/(J·Hz)). Therefore, the energy efficiency of m_i offloading data to s_j on c_n in the t-th slot is given by

$$\eta_{i,j,n}(t) = \frac{U_i(t)}{BE_{i,j,n}(t)} = \frac{R_{i,j,n}(t)}{B\left(P^{TX} + P_0\right)}. \quad (9)$$

2.4. Problem Formulation. In this paper, we aim to address the energy-efficient and low-latency task offloading problem

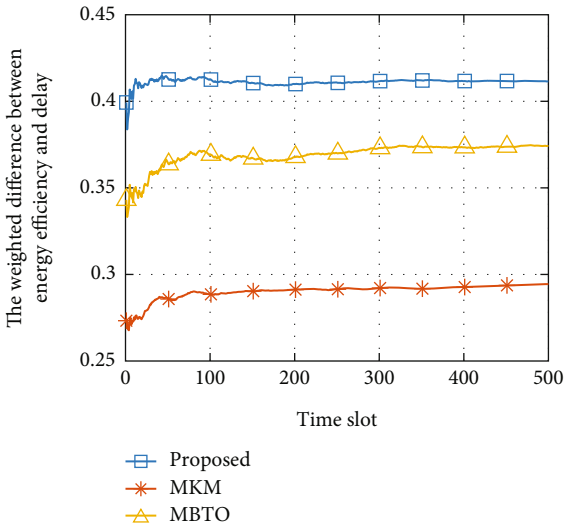

Figure 2: The weighted difference between energy efficiency and delay versus time slots.

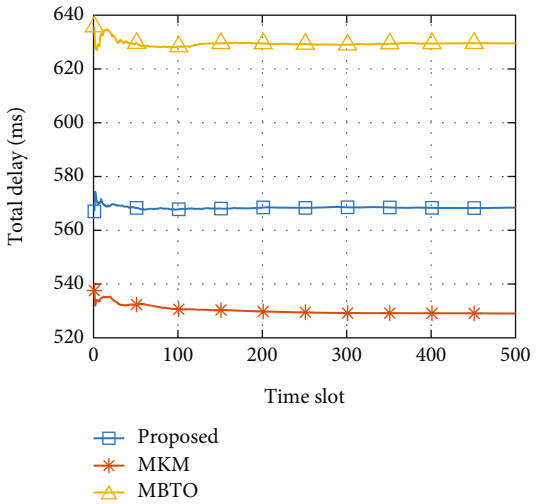

Figure 3: The total delay versus time slots.

in PDIoT. The objective is to maximize the weighted difference between energy efficiency and delay through joint optimization of large-timescale server selection in the first stage and small-timescale channel selection in the second stage. The two-stage task offloading problem is formulated as

$$\begin{aligned}
\mathrm{P1}: &\max_{\{z_{i,j}(l),x_{i,n}(t)\}} \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N} z_{i,j}(l)x_{i,n}(t)\left[V\eta_{i,j,n}(t) - Q_{i,j,n}(t)\right],\\
s.t. C_1: &\sum_{i=1}^{I} z_{i,j}(l) \le q_j,\ \sum_{j=1}^{J} z_{i,j}(l) = 1, s_j \in \mathcal{S}, m_i \in \mathcal{M}, l \in \mathcal{L},\\
C_2: &\sum_{n=1}^{N} x_{i,n}(t) = 1, m_i \in \mathcal{M}, t \in \mathcal{T},\\
C_3: &\mathrm{SINR}_{i,j,n}(t) \ge \mathrm{SINR}_{\min}, s_j \in \mathcal{S}, m_i \in \mathcal{M}, c_n \in \mathcal{C}, t \in \mathcal{T},
\end{aligned} \tag{10}$$

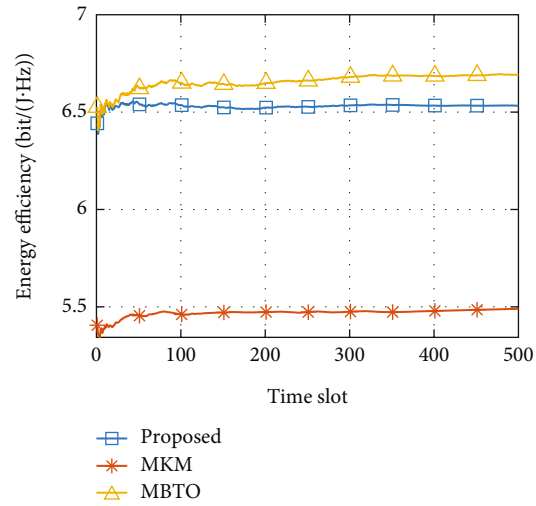

Figure 4: The energy efficiency versus time slots.

where V is used to achieve the tradeoff between energy efficiency and delay. Specifically, when V is large, the influence of energy efficiency is dominant in the optimization objective, and the proposed algorithm tends to maximize energy efficiency. When V is small, the influence of delay is dominant in the optimization objective, the proposed algorithm tends to minimize delay. C_1 represents the server selection constraints. C_2 represents the channel selection constraints. C_3 denotes the task offloading reliability constraints in terms of SINR, where $\mathrm{SINR}_{\min}$ is the threshold.

3. ML-Based Two-Stage Task Offloading Optimization for PDIoT

In this section, we introduce the problem transformation and the proposed ML-based two-stage task offloading optimization algorithm for PDIoT.

3.1. Problem Transformation. We transform the first-stage large-timescale server selection problem of P1 into a many-to-one matching problem between devices and servers. Then, we propose a stable server selection algorithm based on two-side matching with quota to solve it. Next, the second-stage small-timescale channel selection problem of P1 is solved by the proposed adaptive ε-greedy learning algorithm.

3.2. First-Stage Server Selection Based on Two-Side Matching with Quota. Based on the many-to-one two-side matching with quota [32–34], each device and server need to obtain the preference values towards each other. Then, based on the obtained two-side preference values, the first-stage server selection problem is solved according to the many-to-one two-side matching with quota to maximize the weighted difference between energy efficiency and delay.

Theorem 1. *A matching ϕ with quota q_j is defined as ϕ: $\mathcal{M} \cup \mathcal{S} \longrightarrow \mathcal{S} \cup \mathcal{M}$. When $\phi(m_i) = s_j$ and $m_i \in \phi(s_j)$ in the l-th period, m_i and s_j establish a matching relationship, i.e., $z_{i,j}(l$*

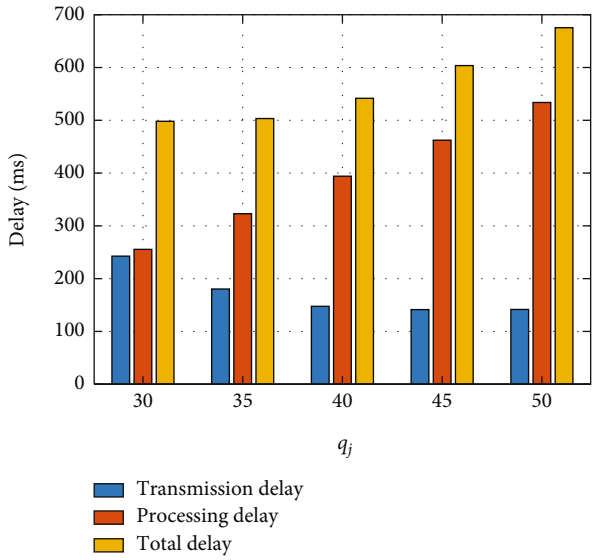

Figure 5: The impact of the quota q_j on transmission delay, processing delay, and total delay.

$) = 1$. *Particularly,* $|\phi(s_j)| \le q_j$, *where* $|\phi(s_j)|$ *is the size of* $\phi($ $s_j)$.

3.2.1. Preference List Construction. Based on P1, both devices and servers establish their matching preference lists. The preference value $\omega_{i,j}(l)$ of m_i for server s_j is defined as the weighted energy efficiency, and the preference value $\gamma_{j,i}(l)$ of s_j for m_i is defined as the negative of total delay, which are given by

$$\omega_{i,j}(l) = V\tilde{\eta}_{i,j}(l), \tag{11}$$

$$\gamma_{j,i}(l) = -\tilde{Q}_{i,j}(l), \tag{12}$$

where $\tilde{\eta}_{i,j}(l)$ and $\tilde{Q}_{i,j}(l)$ are the empirical statistical estimates of energy efficiency and total delay in the l-th period, i.e.,

$$\begin{aligned}\tilde{\eta}_{i,j}(l) &= \frac{\sum_{p=1}^{l-1}\sum_{t=(p-1)T_0+1}^{pT_0}\sum_{n=1}^{N} z_{i,j}(p)x_{i,n}(t)\eta_{i,j,n}(t)}{\sum_{k=1}^{l-1} z_{i,j}(p)},\\ \tilde{Q}_{i,j}(l) &= \frac{\sum_{p=1}^{l-1}\sum_{t=(p-1)T_0+1}^{pT_0}\sum_{n=1}^{N} z_{i,j}(p)x_{i,n}(t)Q_{i,j,n}(t)}{\sum_{k=1}^{l-1} z_{i,j}(p)}.\end{aligned} \tag{13}$$

Based on (11) and (12), m_i and s_j calculate $\omega_{i,j}(l)$ and $\gamma_{j,i}(l)$ and establish their preference lists $\mathscr{F}_i$ and $\mathscr{F}_j$ by sorting preference values in descending order.

3.2.2. Implementation of Two-Side Matching with Quota. The implementation of two-side matching with quota consists of three steps, which are introduced as follows.

Step 1. Initialize the sets of server selection strategy, unmatched devices, and unmatched servers as $\phi = \varnothing$, $\Theta = \mathscr{M}$, and $\Gamma = \mathscr{S}$.

Step 2. m_i, $\forall m_i \in \Theta$ and s_j, $\forall s_j \in \Gamma$, calculate the preference values according to (11) and (12) to obtain the preference lists $\mathscr{F}_i$ and $\mathscr{F}_j$.

Step 3. First, m_i proposes to its most preferred server based on $\mathscr{F}_i$. Afterward, s_j calculates the sum of temporary matches and new proposals. If the sum of temporary matches and new proposals is less than q_j, s_j establishes temporary matches with the devices which have proposed to it. The matched devices are temporarily removed from Θ. Otherwise, based on $\mathscr{F}_j$, s_j establishes temporary matches with only the top q_j devices which have proposed to it. Next, the unmatched devices are added into Θ, and s_j is removed from their preference lists. The matched devices are removed from Θ. If the sum of matches for s_j is equal to q_j , remove s_j from Γ. Finally, return to Step 2, and the unmatched devices make new proposals based on the updated preference lists.

Matching iteration ends until m_i, $\forall m_i \in \mathscr{M}$ establishes a match with a server or its preference list $\mathscr{F}_i = \varnothing$.

3.3. Second-Stage Channel Selection Based on Adaptive ε-Greedy Learning. The second-stage channel selection optimization problem is transformed into an MAB problem, which is addressed by the proposed adaptive ε-greedy learning algorithm. The MAB problem is mainly composed of decision maker, arm, and reward, which are introduced as follows:

(i) *Decision Maker.* The decision maker generates selection decisions and constantly updates the decision by learning the reward from historical feedbacks. We define PDIoT devices as decision makers.

(ii) *Arm.* Based on server selection, the device needs to select a channel for data transmission. Denote the set of arms as $\mathscr{C} = \{c_1, \cdots, c_n, \cdots, c_N\}$.

(iii) *Reward.* Define $r_{i,n}(t)$ as the reward of m_i selecting c_n in the t-th slot, which is given by

$$r_{i,n}(t) = \sum_{j=1}^{J} z_{i,j}(l)\left(V\eta_{i,j,n}(t) - Q_{i,j,n}(t)\right) \tag{14}$$

.

The traditional ε-greedy algorithm uses a linear method to adjust the exploration factor ε, which has a certain degree of blindness [35, 36]. In order to improve the exploration efficiency, we propose an adaptive ε-greedy learning algorithm, which uses the average cumulative reward to dynamically adjust the adaptive factor ε_t to balance exploration and exploitation.

Define $\bar{r}_{i,n}(t)$ as the historical average reward of m_i selecting c_n, which is given by

$$\bar{r}_{i,n}(t) = \frac{\bar{r}_{i,n}(t-1)k_{i,n}(t-1) + x_{i,n}(t)r_{i,n}(t)}{k_{i,n}(t-1) + x_{i,n}(t)}, \tag{15}$$

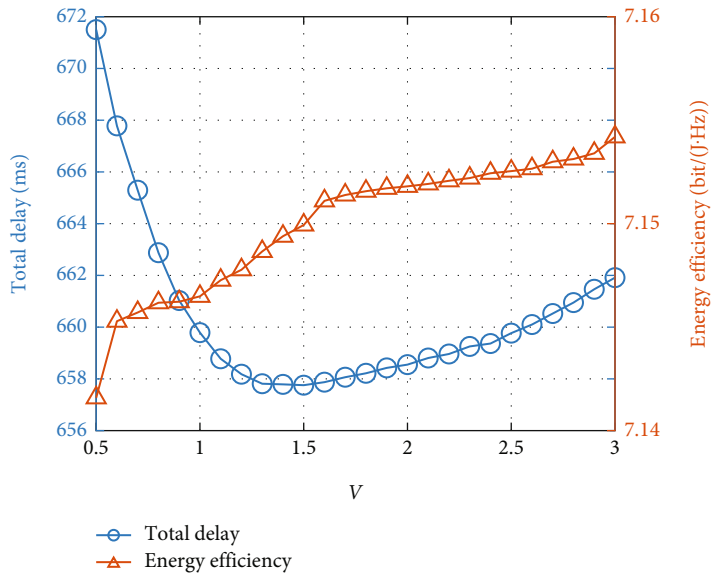

Figure 6: The impact of the weight V on total delay and energy efficiency.

where $k_{i,n}(t)$ represents the total times that m_i selects c_n until the t-th slot. The average cumulative reward under the previous K, (Kt), task offloading strategies is calculated as

$$\bar{r}_K(t) = \frac{1}{K}\sum_{k=t-K}^{t-1}\sum_{i=1}^{I}\sum_{n=1}^{N} x_{i,n}(k)\bar{r}_{i,n}(k). \tag{16}$$

Then, the adaptive exploration factor ε_t is updated as

$$\boldsymbol{\varepsilon}_t = \frac{1}{1+\log_v(1+\bar{r}_K(t))}, \tag{17}$$

where v represents the base of the logarithmic function, and $v > 1$.

The action decision $a_i(t)$ of the adaptive ε-greedy learning algorithm is given by

$$a_i(t) = \begin{cases} \arg \max_{c_n \in C} \bar{r}_{i,n}(t-1), & \mu > \varepsilon_t, \\ \text{random selection}, & \mu \le \varepsilon_t, \end{cases} \tag{18}$$

where $\mu \in (0,1)$ is a random number. When $\mu > \varepsilon_t$, the device selects the channel with the largest historical average reward. When $\mu \le \varepsilon_t$, the device randomly selects a channel. Specifically, $a_i(t) = c_n$ is equivalent to $x_{i,n}(t) = 1$.

3.4. Implementation of Two-Stage Task Offloading. The proposed ML-based two-stage task offloading optimization algorithm consists of three phases, which are summarized in Algorithm 3.3.

Phase 1. Initialization. Initialize $k_{i,n}(1) = 0, \forall m_i \in \mathcal{M}, \forall c_n \in \mathcal{C}$.

Phase 2. Large-Timescale First-Stage Server Selection. At the beginning of each period, each device and server construct their preference lists $\mathcal{F}_i$ and $\mathcal{F}_j$, respectively. Then, perform the two-side matching process with quota based on Section 3.2 and obtain $z_{i,j}(l)$.

Phase 3. Small-Timescale Second-Stage Channel Selection: In each slot, m_i, $\forall m_i \in \Theta$ makes action decision based on (18) and selects the corresponding channel. Then, m_i calculates the $r_{i,n}(t)$ and $\bar{r}_{i,n}(t)$ based on (14) and (15). Finally, update $\bar{r}_K(t+1)$ and $\boldsymbol{\varepsilon}_{t+1}$ based on (16) and (17).

The algorithm ends until $t > T$.

3.5. Computation Complexity. For the first-stage server selection, the computation complexity is $\mathcal{O}(I + 2J + \log(IJ))$, while the computation complexity of the exhaustive-based server selection algorithm is $\mathcal{O}(I! \times J!)$. When I and J are large enough, the computation complexity of the proposed algorithm is much lower than that of the exhaustive-based server selection algorithm. For the second-stage channel selection, the computation complexity is $\mathcal{O}(N \log(N) + N + 2)$. In some ML-based channel selection algorithms such as DRL-based channel selection algorithm and DL-based channel selection algorithm, the computation complexity of the training of deep neural networks is $\mathcal{O}(E + |\mathcal{D}|/K + J)$. Here, E is the number of training epochs, $|\mathcal{D}|$ and K are, respectively, the dataset size and batch size, and J is the computation complexity of each training epoch. $\mathcal{O}(J)$ is related to many free variables such as the computation complexity of each layer, the size of convolution kernel, the number of input and output channels, and spatial dimensions of input and output feature maps. Therefore, the computation complexity of these DRL and DL-based channel selection algorithms are much higher than the proposed algorithm.

4. Simulation Results

We consider a 800m $\times$ 20m transmission line monitoring scenario in PDIoT, which includes 120 devices and 3 edge servers colocated with BSs. The number of channels is 50. The devices are randomly distributed along the transmission

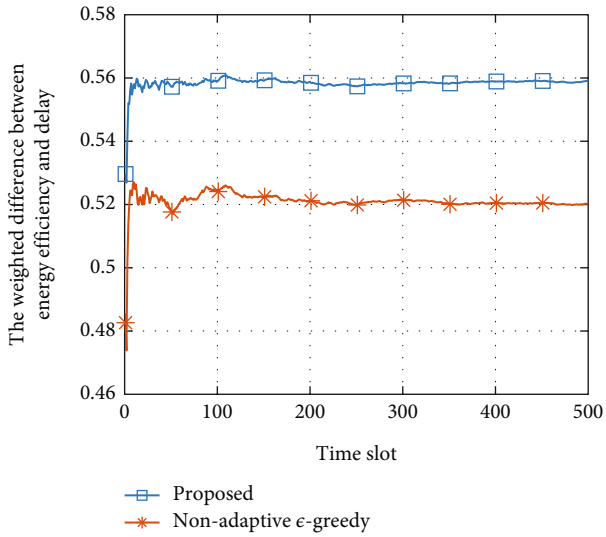

Figure 7: The comparison between the proposed algorithm and the nonadaptive ε-greedy algorithm.

line. Simulation parameters are summarized in Table 2 [37–39]. Two algorithms are utilized for comparison. The first one is the matching based on Kuhn-Munkras (MKM) [40] which aims to minimize task offloading delay. The second one is the matching-based task offloading strategy (MBTO) [41] which aims to maximize energy efficiency through server selection optimization in each period. Both MKM and MBTO cannot achieve small-timescale channel selection optimization.

Figure 2 shows the weighted difference between energy efficiency and delay versus time slots. Compared with MKM and MBTO, the proposed algorithm improves the weighted difference by 39.74% and 9.96%, respectively. The reason is that the proposed algorithm can learn the optimal channel and server selection strategy to minimize the weighted difference based on dynamic network states.

Figures 3 and 4 show the total delay and energy efficiency versus time slots, respectively. Compared with MKM, the proposed algorithm improves energy efficiency by 18.99% but increases delay by 7.45%. Compared with MBTO, the proposed algorithm reduces delay by 9.71% but decreases energy efficiency by 2.38%. The reason is that MKM and MBTO only optimize one aspect of delay or energy efficiency. The proposed algorithm can make a well-balanced tradeoff between energy and delay by dynamically adjusting server and channel selection strategies.

Figure 5 shows the impact of the quota q_j on transmission delay, processing delay, and total delay. The transmission delay decreases with q_j while the processing delay increases with q_j. When q_j increases from 30 to 50, the transmission delay decreases from 242.6389 ms to 141.6561 ms while the processing delay increases by 278.2709 ms. The reason is that as q_j increases, more devices can access to nearby BSs with better transmission performance, thus reducing transmission delay. However, the server allocates less computing resources to each device, thus increasing the processing delay. The total delay increases with q_j since the processing delay has a greater impact on the total delay.

Table 3: Comparison of the computation complexity.

Algorithm	Computation complexity
Proposed	1.541348 s
MKM	0.404835 s
MBTO	0.317423 s
DRLTO	1840.079283 s
FLTO	2057.109636 s
MLTO	1642.427586 s

Figure 6 shows the impact of the weight V on total delay and energy efficiency. With the increase of V, the energy efficiency increases obviously and finally reaches 7.1542 bits/(J·Hz). Meanwhile, the total delay decreases first and then increases gradually, finally reaching 661.91 ms. The reason is that as V increases, the proposed algorithm puts more emphasis on energy efficiency improvement. Devices are inclined to select the channel which can achieve a higher transmission rate, thereby reducing the transmission delay and total delay at first. However, the increasing V enforces devices to select the nearby edge servers with less computing resources to improve energy efficiency, which then increases the processing delay and total delay. Therefore, the proposed algorithm can balance the tradeoff between energy efficiency and delay by adjusting the weight V.

Figure 7 compares the performance of the proposed algorithm and the nonadaptive ε-greedy algorithm. The proposed algorithm outperforms the nonadaptive ε-greedy algorithm by 7.45%. The reason is that the proposed algorithm can adjust ε_t based on the average cumulative reward $\bar{r}_K(t)$ to balance the tradeoff between exploration and exploitation. On the contrary, the nonadaptive ε-greedy algorithm with fixed ε cannot adaptively trade off exploration and exploitation based on current reward, thereby resulting in poor learning performance and lower weighted difference.

Table 3 shows the computation complexity of different algorithms. Due to the consideration of channel selection optimization, the computation complexity of the proposed algorithm is higher than MKM and MBTO, but the proposed algorithm, respectively, improves the weighted difference between energy efficiency and delay by 39.74% and 9.96% compared with MKM and MBTO. Due to the limited computing resources of PDIoT devices, the computation complexities of DRL-based task offloading algorithm (DRLTO), federated learning-based task offloading algorithm (FLTO), and meta learning-based task offloading algorithm (MLTO) are much higher than that of the proposed algorithm.

5. Conclusion

In this paper, the energy-efficient and low-latency task offloading in PDIoT was investigated. An ML-based two-stage task offloading optimization algorithm was proposed to maximize the weighted difference between energy efficiency and delay through joint optimization of large-

timescale server selection and small-timescale channel selection. Simulation results show that the proposed algorithm can improve the weighted difference by 39.74% and 9.96% compared with MKM and MBTO. In the future, the joint optimization of power control and computing resource allocation for multi-QoS guaranteed task offloading in PDIoT will be studied.

Data Availability

No data was used for this article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Key Science and Technology Program of Haikou City under grant number 2020-009, the Key Research and Development Project of Hainan Province under grant number ZDYF2021SHFZ243, the National Natural Science Foundation of China under grant number 62062030, the Scientific Research Fund Project of Hainan University under grant number KYQD (ZR)-21007 and KYQD (ZR)-21008.

References

[1] J. Hu, Y. Li, G. Zhao, B. Xu, Y. Ni, and H. Zhao, "Deep reinforcement learning for task offloading in edge computing assisted power IoT," *IEEE Access*, vol. 9, pp. 93892–93901, 2021.

[2] M. T. Islam, S. Karunasekera, and R. Buyya, "Performance and cost-efficient spark job scheduling based on deep reinforcement learning in cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 7, pp. 1695–1710, 2022.

[3] Z. Zhou, M. Dong, K. Ota, G. Wang, and L. Yang, "Energy-efficient resource allocation for D2D communications underlaying cloud-RAN-based LTE-A networks," *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 428–438, 2016.

[4] W. Duan, J. Gu, M. Wen, G. Zhang, Y. Ji, and S. Mumtaz, "Emerging technologies for 5G-IoV networks: applications, trends and opportunities," *IEEE Network*, vol. 34, no. 5, pp. 283–289, 2020.

[5] D. Jiang, F. Wang, Z. Lv et al., "QoE-aware efficient content distribution scheme for satellite-terrestrial networks," *IEEE Transactions on Mobile Computing*, no. 99, pp. 1–16, 2021.

[6] X. Niu, S. Shao, C. Xin et al., "Workload allocation mechanism for minimum service delay in edge computing-based power internet of things," *IEEE Access*, vol. 7, pp. 83771–83784, 2019.

[7] J.-M. Liang, K.-R. Wu, J.-J. Chen, P.-Y. Liu, and Y.-C. Tseng, "Energy-efficient uplink resource units scheduling for ultra-reliable communications in NB-IoT networks," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 4079017, 17 pages, 2018.

[8] J. Du, C. Jiang, Z. Han, H. Zhang, S. Mumtaz, and Y. Ren, "Contract mechanism and performance analysis for data transaction in mobile social networks," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 2, pp. 103–115, 2019.

[9] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems: a resource-based pricing and user risk-awareness approach," *Sensors*, vol. 20, no. 8, p. 2434, 2020.

[10] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-Optimal dynamic computation offloading for industrial IoT in fog computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 566–576, 2020.

[11] M. Maray and J. Shuja, "Computation offloading in mobile cloud computing and mobile edge computing: survey, taxonomy, and open issues," *Mobile Information Systems*, vol. 2022, Article ID 1121822, 17 pages, 2022.

[12] E. Mustafa, J. Shuja, A. I. Jehangiri et al., "Joint wireless power transfer and task offloading in mobile edge computing: a survey," *Cluster Computing*, vol. 25, no. 4, pp. 2429–2448, 2022.

[13] H. Wu, K. Wolter, P. Jiao, Y. Deng, and M. Xu, "EEDTO: an energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2163–2176, 2021.

[14] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8322–8335, 2019.

[15] Y. Wei, H. Yang, J. Wang et al., "Delay and energy-efficiency-balanced task offloading for electric internet of things," *Electronics*, vol. 11, no. 6, pp. 2079–9292, 2022.

[16] X. Shi, D. Wu, C. Yue, C. Wan, and X. Guan, "Resource allocation for covert communication in D2D content sharing: a matching game approach," *IEEE Access*, vol. 7, pp. 72835–72849, 2019.

[17] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: a contract-matching approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3113–3125, 2019.

[18] S. Abedin, M. Alam, S. Kazmi, N. Tran, D. Niyato, and C. Hong, "Resource allocation for ultra-reliable and enhanced mobile broadband IoT applications in fog network," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 489–502, 2019.

[19] K. Wang, F. Fang, D. Costa, and Z. Ding, "Sub-channel scheduling, task assignment, and power allocation for OMA-based and NOMA-based MEC systems," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2692–2708, 2021.

[20] A. I. Jehangiri, T. Maqsood, A. I. Umar et al., "LiMPO: lightweight mobility prediction and offloading framework using machine learning for mobile edge computing," *Cluster Computing*, pp. 1–19, 2022.

[21] S. Zhou, W. Jadoon, and J. Shuja, "Machine learning-based offloading strategy for lightweight user mobile edge computing tasks," *Complexity*, vol. 2021, Article ID 6455617, 11 pages, 2021.

[22] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8099–8110, 2020.

[23] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: a deep meta reinforcement learning-based task offloading framework for

edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.

[24] S. Chen, J. Chen, Y. Miao, Q. Wang, and C. Zhao, "Deep reinforcement learning-based cloud-edge collaborative mobile computation offloading in industrial networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 364–375, 2022.

[25] S. Yin and F. R. Yu, "Resource allocation and trajectory design in UAV-Aided cellular networks based on multiagent reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2933–2943, 2022.

[26] T. Yang, S. Gao, J. Li et al., "Multi-armed bandits learning for task offloading in maritime edge intelligence networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4212–4224, 2022.

[27] W. Li, Y. Xu, Q. Guo et al., "Joint channel selection and data scheduling in HF jamming environment: An interference-aware reinforcement learning approach," *IEEE Access*, vol. 7, pp. 157072–157084, 2019.

[28] S. Talekar and S. Terdal, "Reinforcement learning based channel selection for design of routing protocol in cognitive radio network," in *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, Bengaluru, India, 2019.

[29] H. Liao, Z. Zhou, X. Zhao, B. Ai, and M. Guizani, "Learning-based queue-aware task offloading and resource allocation for space–air–ground-integrated power IoT," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5250–5263, 2021.

[30] H. Luo, I. A. Hiskens, and Z. Hu, "Stability analysis of load frequency control systems with sampling and transmission delay," *IEEE Transactions on Power Apparatus and Systems*, vol. 35, no. 5, pp. 3603–3615, 2020.

[31] A. Jahid, M. S. Hossain, M. K. H. Monju, M. F. Rahman, and M. F. Hossain, "Techno-economic and energy efficiency analysis of optimal power supply solutions for green cellular base stations," *IEEE Access*, vol. 8, pp. 43776–43795, 2020.

[32] Z. Zhou, K. Ota, M. Dong, and C. Xu, "Energy-efficient matching for resource allocation in D2D enabled cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5256–5268, 2017.

[33] H. Kawther, F. Mounir, and C. Tijani, "Access point backhaul resource aggregation as a many-to-one matching game in wireless local area networks," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 3523868, 11 pages, 2017.

[34] H. Liao, Z. Zhou, B. Ai, and M. Guizani, "Learning-based energy-efficient channel selection for edge computing-empowered cognitive machine-to-machine communications," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, Antwerp, Belgium, 2020.

[35] S. R. Nabavi, N. O. Eraghi, and J. A. Torkestani, "WSN routing protocol using a multiobjective greedy approach," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6664669, 12 pages, 2021.

[36] N. Kiran, C. Pan, S. Wang, and C. Yin, "Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks," *Journal of Communications and Networks*, vol. 22, no. 1, pp. 1–11, 2020.

[37] Y. Wang, G. Zheng, H. Ma, L. Yang, and J. Li, "A joint channel selection and routing protocol for cognitive radio network," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 6848641, 7 pages, 2018.

[38] L. de MBA Dib, V. Fernandes, M. D. Filomeno, and M. V. Ribeiro, "Hybrid PLC/wireless communication for smart grids and internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 655–667, 2018.

[39] M. Qiao, H. Zhao, S. Huang, L. Zhou, and W. Shan, "Optimal channel selection based on online decision and offline learning in multichannel wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 7902579, 13 pages, 2017.

[40] Y. Ren, X. Chen, S. Guo, S. Guo, and A. Xiong, "Blockchain-based VEC network trust management: a DRL algorithm for vehicular service offloading and migration," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8148–8160, 2021.

[41] C. Swain, M. N. Sahoo, A. Satpathy et al., "METO: Matching-Theory-Based efficient task offloading in IoT-fog interconnection networks," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12705–12715, 2021.