Multi-Layer Computation Offloading in Distributed Heterogeneous Mobile Edge Computing Networks

Pengfei Wang, *Graduate Student Member, IEEE*, Boya Di, *Member, IEEE*, Lingyang Song, *Fellow, IEEE*, and Nicholas R. Jennings, *Fellow, IEEE*

***Abstract*—In this paper, we consider distributed heterogeneous multi-layer mobile edge computing (HetMEC) networks, where resource-poor edge devices (EDs) upload computing tasks for processing to the mobile edge computing (MEC) servers and a cloud center (CC). To reduce total energy consumption, computation offloading and resource allocation are independently performed by each device and each server. However, due to the partial information available at each device and server, the offloading strategies may overwhelm the layers above. This may lead to network congestion, i.e., so many tasks are offloaded to the same node that this node is overloaded. To address this problem, we develop a smart pricing mechanism to coordinate the computation offloading of multi-layer devices, where the CC charges the MEC servers and EDs for computing services and network congestion. In particular, to satisfy the latency constraints of each task, we construct a Lagrangian framework where multi-agent reinforcement learning is utilized by each MEC server to determine its offloading strategies and resource allocation, so that the total energy consumption is reduced. Simulation results show that our algorithm achieves an energy consumption reduction of 28% and a decrease in congestion probability of between 28% and 100% compared to the state of the art.**

***Index Terms*—Minimum energy control, distributed computing, resource management.**

I. Introduction

With the emergence of various mobile applications propelled by the increasing popularity of mobile devices, vast numbers of computation-intensive and energy-consuming tasks are generated at edge devices (EDs), i.e., mobile devices at the edge of the communication network [2]. Typical applications include virtual and augmented reality, face recognition and monitoring data analysis [2]. To overcome the challenges of the resource-poor and battery-limited mobile devices for processing such tasks, mobile edge computing (MEC) has been proposed to enable the computation to be performed at the MEC servers [3]. However, tremendous computation burden is placed on the MEC servers which are often equipped with limited computing capacities. To further mitigate this issue, heterogeneous multi-layer MEC (HetMEC) has been proposed [4], where the computing tasks of EDs can be first offloaded to the MEC servers, and then the tasks that cannot be processed by MEC servers are further offloaded to a cloud center (CC). Effectively exploring the computation offloading among multiple layers in the HetMEC network allows the computing resources of both MEC servers and CC to be fully exploited to support the computation-intensive tasks of EDs.

To extend battery lifetime of devices, energy saving is a key goal in HetMEC networks. Since the energy consumption of the computing and transmission varies at different devices (i.e., ED, MEC server, and CC), effective energy saving requires the suitable computation offloading and resource allocation among multiple devices on different layers [5]. However, such coordination introduces a heavy signaling overhead for the *centralized* management of computation offloading strategies. They require *global information* about all devices at the central node [4], [6], [7], which is not practical enough especially when the MEC servers belong to different operators. Thus, a distributed scheme is desirable in which each device determines its computation offloading and resource allocation individually. However, such distributed approaches bring two main challenges:

- It is hard to achieve the overall energy saving goal in a distributed HetMEC network. Servers with different computing capacities, serving areas, and energy consumption models only make independent strategies to minimize their own energy consumption, which usually deviates from the goal of reducing the total energy consumption.
- The individual decisions made by different servers are likely to induce the data aggregation due to the partial information available at each other. The network congestion occurs when the amount of aggregated data surpasses that servers' computing capacity.

To handle these challenges, we adopt a pricing-based distributed approach, where the CC charges EDs and MEC servers for providing computing services and compensating

Manuscript received October 27, 2021; revised January 31, 2022; accepted March 18, 2022. Date of publication March 24, 2022; date of current version June 9, 2022. This work was supported by the National Nature Science Foundation of China under grant number 61941101 and 61931019, and in part by Beijing Natural Science Foundation under Grant L212027. This current submission is an extension of our conference paper [1] published at the IEEE ICC, Dublin, Ireland, June 2020 [DOI: 10.1109/ICC40277.2020.9148801]. Our conference paper [1] only describes the basic model and presents some preliminary results. We have rewritten the major parts of the paper to present the problem statement and proposed algorithm more clearly. We have enriched the related works on the distributed methods. The theoretical analysis and simulation results have been greatly extended. The associate editor coordinating the review of this article and approving it for publication was N. Zhang. *(Corresponding authors: Lingyang Song; Boya Di.)*

Pengfei Wang, Boya Di, and Lingyang Song are with the National Engineering Laboratory for Big Data Analysis and Applications, Department of Electronics, Peking University, Beijing 100871, China (e-mail: wangpengfei13@pku.edu.cn; diboya@pku.edu.cn; lingyang.song@pku.edu.cn).

Nicholas R. Jennings is with the Department of Computer Science, Loughborough University, Loughborough LE11 3TU, U.K. (e-mail: n.r.jennings@lboro.ac.uk).

Digital Object Identifier 10.1109/TCCN.2022.3161955

for the loss of network congestion they cause. When a high price is charged, the EDs and MEC servers are driven to process computing tasks locally for lower cost rather than to upload tasks to the upper-layer. However, in traditional pricing mechanisms [10], [11], given an undifferentiated price for various devices, similar offloading decisions may be made by MEC servers, which leads to the computation overload at one node. To avoid the congestion induced by computation overload, we develop a multi-layer smart pricing mechanism that uses differentiated prices for various devices on multiple layers, so as to adjust the offloading behaviors of different device respectively. Moreover, coordinated by the developed smart pricing mechanism, the interests of each ED, MEC server and CC are aligned to the overall energy saving target. Thus, the global energy consumption reduction can be realized by the distributed computation offloading and resource allocation of different devices.

Several existing works investigate distributed energy-efficient computation offloading and resource allocation in either *single-server* [12]–[16] or *multi-server* [17]–[20] two-layer MEC networks. For single-server MEC networks, [13]–[15] utilize game theory for the distributed computation offloading and resource allocation. A Nash equilibrium is achieved via the distributed computation offloading or multi-agent stochastic learning of multiple interacting devices. By charging mobile users a fee for the congestion they cause, [12] proposes a pricing mechanism for a single MEC server which ensures the selfish mobile users make socially beneficial offloading decisions. For multi-server MEC networks, [19] proposes a coalitional game-based pricing scheme to arrange computation offloading between EDs and MEC servers, where each scheduled ED selects the MEC server within the same coalition and pays for the MEC service.

Unfortunately, these aforementioned works do not fully consider the interactions of computation offloading and resource allocation between the CC and multiple MEC servers in a distributed HetMEC network. The self-interested decisions may cause the concentrated computation offloading that overloads the receiving node, but the network congestion loss induced by such self-interested decisions is not fully investigated, thus resulting in the waste of energy and resources [13]–[20]. Though congestion games could be utilized to model the competition of selfish players for resources [21], this line of work traditionally assumes that the cost functions of all players for a resource are the same. However, given the heterogeneous nature[1] of devices in the HetMEC network, the energy consumption of the ED, MEC server and CC differs when processing the same task. Thus, the congestion games cannot resolve our difficulty.

Against this background, the main contributions of our work are summarized as below:

- We design a pricing-based multi-layer computation offloading scheme to reduce the total energy consumption of the distributed HetMEC network. A novel method is developed for the CC to adjust prices discriminatively for multiple MEC servers and EDs on different layers. Such developed pricing mechanism aligns the targets of these self-interested devices to the overall energy saving in that distributed network.
- Different MEC servers determine their own computation offloading and resource allocation schemes using the multi-agent reinforcement learning (RL) in a Lagrangian framework. Given such developed Lagrangian-based multi-agent RL framework, each MEC server can be guaranteed to achieve a feasible offloading strategy satisfying all constraints, thus greatly alleviating the network congestion.
- Theoretical analysis shows that our mechanism can always meet all latency constraints even though only partial environment information is known to each node. Simulation results show that our algorithm obtains a 28% energy consumption reduction and a maximum 100% network congestion alleviation compared to the state of the art schemes.

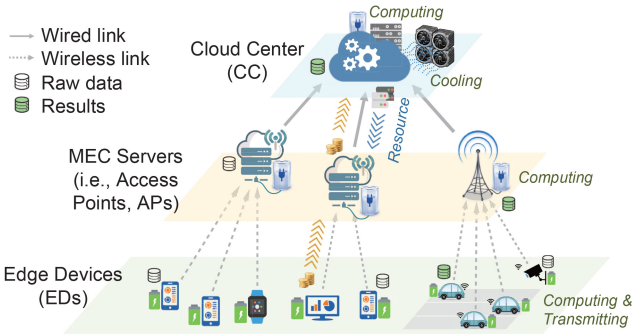


Fig. 1. The multi-layer HetMEC network architecture.

The rest of the paper is organized as follows. In Section II we describe the system model of the distributed HetMEC network. In Section III we formulate the distributed energy consumption minimization problem. To solve this problem, we design a pricing based distributed task offloading and resource allocation algorithm, and analyze the feasibility as well as offloading strategies in Section IV. Simulation results are given in Section VI. Finally the conclusions are drawn in Section VII.

II. System Model

We first describe the pricing based distributed HetMEC network, and then present the computing model, transmission model and energy consumption model of each device.

A. Scenario Description

As shown in Fig. 1, we consider a three-layer HetMEC network consisting of one CC on the top layer, M MEC servers[2] (integrated with access points(APs)) in the middle

[1]Although the heterogeneous nature of HetMEC has been discussed in [4], the authors only investigate a centralized latency minimization approach without considering the distributed feature of servers or discussing the energy saving.

[2]Given a three-layer HetMEC network, we use APs (Access Points) to represent MEC servers in the radio access network in this paper, because the MEC servers and APs are usually integrated to realize both the computing and data transmission.

layer and *N* EDs on the bottom layer. The EDs connect with the APs via wireless links according to the distance-aware[3] criteria [22], and the APs communicate with the CC via wired links.

A typical *uplink* computation offloading scenario is considered, where the computing tasks are generated at the EDs and the task can be processed at the ED, AP or CC. For each task, each ED chooses to process locally or offload to the AP via wireless link individually. Similarly, the AP either processes the received task itself or continues to offload the task to the CC via a wired link [4]. The CC then processes all tasks received from the APs. Binary offloading is performed.[4] The processing results are fed back to the ED.[5] However, due to the incomplete view of the environment for each device in such a distributed network, the individual decisions of devices may not avoid network congestion or satisfy latency constraints.

To resolve this issue, we use a smart pricing based multi-layer computation offloading strategy in the Lagrangian framework.[6] The strategy maintains the stable functioning of the distributed HetMEC network while minimizing each device's offloading cost (e.g., energy consumption) during task processing. Specifically, the CC charge other devices (including APs and EDs) to compensate for its resources usage and congestion cost[7] for processing their offloaded tasks, thereby motivating other devices to adjust their strategies. According to the prices announced by the CC, the EDs and APs make their offloading decisions by weighing the cost of processing themselves and offloading to the upper layer. Therefore, the CC coordinates the offloading decisions of EDs and APs dynamically by adjusting the prices according to the real-time data generation speed of tasks.

Remark 1: In the discussed HetMEC network, we focus on reducing the network congestion induced by improper computation offloading strategies in approach of adjusting the multi-layer computation offloading and resource allocation scheme. For the case where the network is congested whatever the offloading strategies due to the large amount of data, we have discussed in our other work [9], which is not the focus of this paper.

B. Computing Model

1) Edge Device: The EDs, including the smart phones, cameras, sensors and so on, communicate with the AP via wireless links. The set of *N* EDs is denoted by $\mathcal{N}$. We consider the data generation process in *each time unit* T_0 at the ED. The amount of data generated in each time unit at ED *i* is denoted by λ_i, and the number of total CPU cycles required per unit of time to process these data, i.e., the computing load, is denoted by b_i. The data generation speed of ED *i* can be expressed by λ_i / T_0.

The computation offloading indicator at the ED *i* on the bottom layer is denoted by $x_i^b \in \{0, 1\}$, where the data are processed at the ED when $x_i^b = 1$ and offloaded to the AP when $x_i^b = 0$. Thus, the amount of data processed to the ED *i* is expressed by $\lambda_i x_i^b$. Therefore, the computing capacity of ED *i*, denoted by θ_i^b, should ensure that

$$\theta_i^b \geq x_i^b b_i / T_0, \ \forall 1 \leq i \leq N, \tag{1}$$

$$\theta_i^b \leq \Theta_i^b, \ \forall 1 \leq i \leq N, \tag{2}$$

where Θ_i^b represents the upper bound of the computing capacity of ED *i*. Therefore, the computing time of ED *i* can be expressed by

$$t_{i,comp}^b = \frac{b_i x_i^b}{\theta_i^b}. \tag{3}$$

2) Access Point: The AP receives the uploaded tasks from the connected EDs via wireless links, and chooses to process them itself or to offload to the CC. Let $\mathcal{M}$ denote the set of APs. The number of EDs connected to AP *j* is denoted by N_j, the set of which is expressed by $\mathcal{N}_j$.

We introduce a binary variable x_i^m to indicate whether the AP on the *middle* layer processes the task delivered from ED *i*, and thus, the amount of data that the AP processes is $\lambda_i x_i^m$. Similarly, the computation offloading indicator at the CC on the *top* layer for the data generated at the ED *i* is denoted by x_i^t. Different tasks assigned to the same AP are processed in parallel using their allocated computing capacity. The computing capacity of AP *j* allocated to the task uploaded from ED *i* is denoted by $\theta_{j,i}^m$, which should satisfy

$$\theta_{j,i}^m \geq x_i^m b_i / T_0, \ \forall i \in \mathcal{N}_j, 1 \leq j \leq M, \tag{4}$$

$$\sum_{i \in \mathcal{N}_j} \theta_{j,i}^m \leq \Theta_j^m, \ \forall 1 \leq j \leq M, \tag{5}$$

where $x_i^m b_i$ represents the required CPU cycles for the AP, and Θ_j^m is the upper bound of the computing capacity of the AP *j*. Therefore, the computing time of the data from ED *i* at AP *j* can be expressed by [23]

$$t_{i,comp}^m = \frac{b_i x_i^m}{\theta_{j,i}^m}. \tag{6}$$

3) Cloud Center: Different tasks received by the CC are processed in parallel using the allocated computing capacity. The amount of data offloaded to the CC from ED *i* is $\lambda_i x_i^t$, and the computing capacity of the CC allocated to ED *i* for processing its data is denoted by θ_i^t, which should satisfy that

$$\theta_i^t \geq x_i^t b_i / T_0, \tag{7}$$

$$\sum_{i=1}^{N} \theta_i^t \leq \theta_0^t, \tag{8}$$

where θ_0^t denotes the upper bound of the computing capacity of the CC. Therefore, to avoid the congestion, the total amount

[3]In the distance-aware criteria, the ED is associated with the AP with the minimum distance.

[4]Binary offloading model is generally applied for the highly integrated task that is impartible and has to be processed as a whole at the ED, AP or CC, e.g., the image recognition task that require all information of the whole image.

[5]Since the amount of results is usually much smaller than that of the raw data, we only consider the uplink transmissions.

[6]Our proposed pricing based offloading strategy can also be expanded to the partial offloading case.

[7]The congestion cost refers to the additional resource consumption cost of the CC resulted from the congestion.

TABLE I
SUMMARY OF KEY NOTATION

Constant	Definition
N	The number of EDs
M	The number of MEC servers (i.e., APs)
N_j	The number of EDs connected with AP j
λ_i	The amount of data generated in each time unit at ED i
b_i	The required number of CPU cycles in each time unit of the task at ED i
τ_i	The latency requirement for the task generated at ED i
Θ_j^m	The upper bound of the computing capacity of the AP j (at the middle layer)
ϕ_0	The total wired transmission resources of the CC
Variable	**Definition**
x_i^b, x_i^m, x_i^t	The offloading indicator of the task at ED i on the bottom (ED) / middle (AP) / top (CC) layer
θ_i^b	The computing capacity of ED i
$\theta_{j,i}^m$	The computing capacity of AP j allocated to the task uploaded from ED i
θ_i^t	The computing capacity of the CC allocated to the task generated at ED i
p_i	The transmitting power of ED i
ϕ_i	The wired transmission rate for the AP to upload the task generated at ED i to the CC
$t_{i,comp}^{b/m/t}$	The computing time of the ED / AP / CC for processing the task generated at ED i
$t_{i,tran}^b$	The transmission time of ED i for uploading the task
$t_{i,tran}^m$	The transmission time of the AP for uploading the task generated at ED i
E_i^b, E_j^m, E^t	The energy consumption ED i, AP j and CC

of offloaded data should be smaller than the total computing capacity of the CC, expressed by

$$\theta_0^t \geq \sum_{i=1}^{N} x_i^t b_i / T_0. \tag{9}$$

Considering the task offloaded from ED *i*, the computing time at the CC can be calculated by

$$t_{i,comp}^t = \frac{b_i x_i^t}{\theta_i^t}. \tag{10}$$

C. Transmission Model

We consider the linear correlated transmission resources, e.g., the bandwidth resources in the Ethernet and the frequency resources in the wireless network [24], which represent the fact that the transmission data rate is linearly related to the amount of transmission resources. Therefore, we model the delays in a linear way as the load over the allocated capacity.

1) Edge Device: Orthogonal channels with bandwidth *W* are allocated to the wireless links between the EDs and the AP, and the set of these channels are denoted by $\mathcal{K} = \{1, 2, \ldots, K\}$. The transmission power of ED *i* is denoted by p_i. The transmission data rate from ED *i* to AP *j* can be expressed by

$$r_{i,j}(p_i) = W \cdot \log\Big(1 + \frac{p_i g_{i,j}}{\sigma^2}\Big), \tag{11}$$

where $g_{i,j}$ is the channel gain of the wireless channel between ED *i* and AP *j*, and σ^2 denotes the variance of the additive white Gaussian noise (AWGN). The channel gain is defined by

$$g_{i,j} = \frac{G|h_0^{(k)}|^2}{(d_{i,j})^\alpha}, \tag{12}$$

where *G* is the power gain constant introduced by the amplifier and antenna, $h_0^{(k)} \sim \mathcal{CN}(0,1)$ is a complex Gaussian variable representing the Rayleigh fading,[8] and $d_{i,j}$ is the distance from ED *i* to AP *j*, and α is the fading factor [26]. The transmission rate between ED *i* and AP *j* should satisfy that

$$r_{i,j}(p_i) \geq \Big(1 - x_i^b\Big)\lambda_i / T_0, \ \forall 1 \leq i \leq N, 1 \leq j \leq M, i \in \mathcal{N}_j, \tag{13}$$

where $\lambda_i(1 - x_i^b)$ is the amount of raw data transmitted to the AP. The transmission latency from the ED *i* to the AP is given by

$$t_{i,tran}^b = \frac{\lambda_i\Big(1 - x_i^b\Big)}{r_{i,j}(p_i)}. \tag{14}$$

2) Access Point: Considering the task uploaded from ED *i*, when the AP *j* continues to deliver these data up to the CC, the required wired transmission rate ϕ_i should satisfy that

$$\phi_i \geq x_i^t \lambda_i / T_0, \tag{15}$$

where $\lambda_i x_i^t$ is the amount of raw data offloaded to the CC from ED *i*. Therefore, the transmission time of these data from AP *j* to the CC can be calculated by

$$t_{i,tran}^m = \frac{\lambda_i x_i^t}{\phi_i}. \tag{16}$$

The wired transmission resources of all APs allocated by the CC are constrained by the amount of total transmission resources possessed by the CC, denoted by ϕ_0, expressed by

$$\phi_0 \geq \sum_{i=1}^{N} \phi_i. \tag{17}$$

[8]The Rayleigh fading is reasonable for modeling the effect of heavily built-up urban environments on radio signals, especially when there is no dominant propagation along a line of sight between the transmitter and receiver [25].

D. Energy Consumption Model

1) Edge Device: The energy consumption of the ED includes the computing and transmission consumption [14]. The computing energy consumption of ED *i* is expressed by

$$E_{i,c}^{b} = \kappa_b b_i x_i^b \left(\theta_i^b\right)^2, \tag{18}$$

where κ_b is the effective switched capacitance depending on the chip architecture at the ED, and θ_i^b denotes the computing capacity (CPU cycles per second) of ED *i*. The transmission energy consumption of ED *i* is expressed by

$$E_{i,t}^{b} = p_i t_{i,tran}^{b}, \tag{19}$$

which is proportional to the transmission power and transmission time of ED *i*. Therefore, the energy consumption of ED *i* is expressed by

$$E_i^b = E_{i,c}^b + E_{i,t}^b. \tag{20}$$

2) Access Point: For AP *j*, e.g., the base station equipped with an MEC server, we consider the computation energy consumption with low CPU voltage [27], [28], which is given by[9]

$$E_j^m = \kappa_m \left(\sum_{i\in\mathcal{N}_j} b_i x_i^m\right)\left(\sum_{i\in\mathcal{N}_j} \theta_{j,i}^m\right)^2, \tag{21}$$

where κ_m is the effective switched capacitance depending on the chip architecture at the AP, $\sum_{i\in\mathcal{N}_j} b_i x_i^m$ represents the total required CPU cycles of AP *j*, and $\theta_{j,i}^m$ denotes the computing capacity (CPU cycles per second) of AP *j* allocated to the data from ED *i* [27].

3) Cloud Center: The energy consumption of the CC can be expressed by

$$E^t = \sum_{i=1}^{N}\left[\left(\beta_1\left(\theta_i^t\right)^{\varepsilon} + \beta_2\right)\cdot t_{i,comp}^t\right], \tag{22}$$

where β_1 and β_2 are positive constants, and the parameter ε ranges from 2.5 to 3 [27], and $t_{i,comp}^t$ represents the computing time for the CC to process the task offloaded from ED *i*.

III. Problem Formulation

In this section, we aim to minimize the total energy consumption of all entities in the distributed HetMEC network via the multi-layer computation offloading and resource allocation. Due to the distributed nature, each entity (ED, AP, or CC) optimizes its own component and only limited information can be exchanged among these entities. The smart pricing mechanism is utilized for aligning the respective objectives of different entities with the global objective (total energy consumption minimization) in the Lagrangian framework.

[9]Here we neglect the unimportant wired transmission energy consumption, which is very common in the literature [27].

The total energy consumption of the whole network includes the energy consumption of the CC, the APs and the EDs, which can be expressed[10] by

$$E(\boldsymbol{x},\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{\phi}) = E^t(\boldsymbol{x},\boldsymbol{\theta}) + \sum_{j=1}^{M} E_j^m(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\phi}) + \sum_{i=1}^{N} E_i^b(\boldsymbol{x},\boldsymbol{p},\boldsymbol{\theta}), \tag{23}$$

where E^t is the energy consumption of the CC as shown in (22), E_j^m denotes the energy consumption of AP *j* as shown in (21), and E_i^b represents the energy consumption of ED *i* as shown in (20). According to the different computation offloading strategies, the **latency** of processing the task generated at ED *i* can be classified to three cases:

1) When the task of ED *i* is processed locally, i.e., $x_i^b = 1, x_i^m = 0, x_i^t = 0$, the latency comes from the computing time of ED;
2) When the task of ED *i* is processed at the AP, i.e., $x_i^b = 0, x_i^m = 1, x_i^t = 0$, the latency comes from the transmission time of ED and the computing time of AP;
3) When the task of ED *i* is processed at the CC, i.e., $x_i^b = 0, x_i^m = 0, x_i^t = 1$, the latency consists of the transmission time of ED and AP, as well as the computing time of CC.

Therefore, we present the latency of processing the task generated by ED *i* as below.

$$\begin{aligned} t_i = {} & x_i^b t_{i,comp}^b + x_i^m \cdot \left(t_{i,tran}^b + t_{i,comp}^m\right) \\ & + \; x_i^t \cdot \left(t_{i,tran}^b + t_{i,tran}^m + t_{i,comp}^t\right), \\ & x_i^b + x_i^m + x_i^t = 1, \; x_i^b, x_i^m, x_i^t \in \{0,1\}. \end{aligned} \tag{24}$$

A. Smart Pricing Based Energy Consumption Minimization

While satisfying the task latency requirements, we aim to minimize the total energy consumption of the whole network by adjusting the computation offloading decision $\boldsymbol{x}$, the power control for the wireless transmission $\boldsymbol{p}$, the computing resource allocation of each device $\boldsymbol{\theta}$, and the wired transmission resource allocation of the CC $\boldsymbol{\phi}$. The total energy consumption minimization problem is formulated as

$$\min_{\boldsymbol{x},\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{\phi}} \; E(\boldsymbol{x},\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{\phi}) \tag{25}$$

$$s.t. \;\; (1),\; (2),\; (4),\; (5),\; (7),\; (8),(9),\; (13),\; (15),\; (17),\; (24),$$
$$x_i^b + x_i^m + x_i^t = 1, \; x_i^b, x_i^m, x_i^t \in \{0,1\}, \; \forall 1 \le i \le N, \tag{25a}$$

$$t_i \le \tau_i, \; \forall 1 \le i \le N, \tag{25b}$$

where constraint (25a) indicates that the data generated by ED *i* needs to be processed thoroughly at ED *i*, the corresponding AP, or the CC. Constraint (25b) implies that the latency of processing the task generated by each ED *i* should be no longer than the requirement τ_i.

[10]The bold form $\boldsymbol{x}$ (similar for $\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{\phi}$) denotes the vector of all computation offloading indicators $x_i^{b/m/t}$.

Based on the Lagrangian multiplier theory [29], we convert problem (25) into an equivalent Lagrange duality problem (26), as described below. For the distributed HetMEC network where each entity handles its own Lagrangian component and individually minimizes its own energy consumption, we then decompose the converted problem to each ED, AP and the CC using the smart pricing mechanism. To alleviate the congestion in the HetMEC network during the distributed optimization, a monetary penalty is charged by the CC for the EDs and APs when their offloading decisions induce network congestion. Specifically, the Lagrangian multiplier method is applied to reformulate the problem (25), where the Lagrange multipliers $\boldsymbol{\omega} = \{\omega_1, \omega_2, \omega_3^{(i)}, \omega_4^{(i)}, \omega_5^{(i)}\}, 1 \le i \le N$, refer to the prices for the EDs and APs. To simplify the expression, we substitute 1 s into the time unit duration T_0 in the next parts. The pricing based energy consumption minimization and smart pricing mechanism design problem is formulated as

$$\begin{aligned}\min_{\boldsymbol{x},\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{\phi}} \max_{\boldsymbol{\omega}} \; & L(\boldsymbol{x},\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\omega}) = E(\boldsymbol{x},\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{\phi}) \\ & + \omega_1\left(\sum_{i=1}^{N}\lambda_i x_i^t - \phi_0\right) + \omega_2\left(\sum_{i=1}^{N} b_i x_i^t - \theta_0^t\right) \\ & + \sum_{i=1}^{N}\left[\omega_3^{(i)}\left(b_i x_i^t - \theta_i^t\right) + \omega_4^{(i)}\left(\lambda_i x_i^t - \phi_i\right)\right] + \sum_{j=1}^{M}\sum_{i\in\mathcal{N}_j}\omega_5^{(i)} \\ & \times\left[\frac{x_i^b b_i}{\theta_i^b} + \frac{\left(1-x_i^b\right)\lambda_i}{W\cdot\log\left(1+\frac{p_i g_{i,j}}{\sigma^2}\right)} + \frac{x_i^m b_i}{\theta_{j,i}^m} + \frac{x_i^t\lambda_i}{\phi_i} + \frac{x_i^t b_i}{\theta_i^t} - \tau_i\right] \end{aligned} \tag{26}$$

$$s.t.\;\; (1),\ (2),\ (4),\ (5),\ (8),\ (13),\ (17),$$

$$x_i^b + x_i^m + x_i^t = 1,\; x_i^b, x_i^m, x_i^t \in \{0,1\},\; \forall 1 \le i \le N, \tag{26a}$$

$$\omega_1 \ge 0, \omega_2 \ge 0, \omega_3^{(i)} \ge 0, \omega_4^{(i)} \ge 0, \omega_5^{(i)} \ge 0,\; \forall 1 \le i \le N, \tag{26b}$$

where $x_i^t = (1 - x_i^b - x_i^m)$ is determined by the offloading decisions of both ED and AP. Below we illustrate the physical meaning of each Lagrangian multiplier in the designed pricing mechanism.

- The Lagrange multipliers ω_1 and ω_2 indicate the **monetary penalty for the network congestion**. The congestion occurs when the computing or transmission capacity of the CC cannot support the processing or transmission of the tasks offloaded by EDs and APs. In this case, both EDs and APs are charged by the CC for the congestion cost.
- The Lagrange multiplier $\omega_3^{(i)}$ represents the **price for the computing resources of the CC** imposed upon ED i and its connected AP. For processing the offloaded task, the CC allocates the computing resources for the tasks generated by the ED and delivered by the AP.
- The Lagrange multiplier $\omega_4^{(i)}$ represents the **price for the transmission resources of the CC**. The CC allocates the wired transmission resources to APs for computation offloading. The ED and AP need to pay for the allocated transmission resources.
- The Lagrange multiplier $\omega_5^{(i)}$ denotes the **monetary penalty** posed on ED i and its connected AP when the latency requirement is not satisfied, i.e., when the processing time $t_i > \tau_i$.

We then decompose the above global optimization problem (26) for each ED, AP and the CC, where each entity handles its own Lagrangian component.

B. ED Layer: Offloading Strategy Design and Power Control Problem

In response to the prices set by the CC, each ED determines whether to offload the task x_i^b, and its computing capacity θ_i^b and wireless transmission power p_i. We only consider the part that ED i controls in the pricing based energy consumption expressed in (26), expressed by

$$\begin{aligned}\min_{x_i^b,p_i,\theta_i^b} L_i^b\left(x_i^b,p_i,\theta_i^b\right) = {} & \kappa_b b_i x_i^b\left(\theta_i^b\right)^2 + \frac{p_i\lambda_i\left(1-x_i^b\right)}{W\cdot\log\left(1+\frac{p_i g_{i,j}}{\sigma^2}\right)} \\ & + \omega_1\left(1-x_i^b\right)\lambda_i + \omega_2\left(1-x_i^b\right)b_i + \omega_3^{(i)}\left(1-x_i^b\right)b_i + \omega_4^{(i)}\left(1-x_i^b\right)\lambda_i \\ & + \omega_5^{(i)}\left[\frac{x_i^b b_i}{\theta_i^b} + \frac{\left(1-x_i^b\right)\lambda_i}{W\log\left(1+\frac{p_i g_{i,j}}{\sigma^2}\right)} + \frac{\left(1-x_i^b\right)\lambda_i}{\phi_i} + \frac{\left(1-x_i^b\right)b_i}{\theta_i^t}\right] \end{aligned} \tag{27}$$

$$s.t.\;\; x_i^b + x_i^m + x_i^t = 1,\; x_i^b, x_i^m, x_i^t \in \{0,1\}, \tag{27a}$$

$$x_i^b b_i \le \theta_i^b \le \Theta_i^b, \tag{27b}$$

$$r_i = W\cdot\log\left(1+\frac{p_i g_{i,j}}{\sigma^2}\right) \ge \left(1-x_i^b\right)\lambda_i. \tag{27c}$$

Constraint (27a) guarantees that the task generated at ED i can be processed thoroughly, derived from constraint (26a). Constraints (27b) and (27c) are consistent with constraints (1), (2) and (13).

C. AP Layer: Offloading Strategy Design and Computing Resource Allocation Problem

According to the prices set by the CC, each AP chooses to deliver the received tasks to the CC or to process them by itself, and $\boldsymbol{x}_i^m, i \in \mathcal{N}_j$, denotes the computation offloading decisions of AP j for the tasks uploaded from its connected EDs. Accordingly, the computing capacity allocation $\boldsymbol{\theta}_{j,i}^m$ for the tasks from different EDs is performed. We only consider the part that AP j determines in the pricing based energy consumption expressed in (26). Therefore, the distributed energy consumption minimization problem for AP j can be formulated by

$$\begin{aligned}\min_{\boldsymbol{x}_i^m,\boldsymbol{\theta}_{j,i}^m} L_j^m\left(\boldsymbol{x}_i^m,\boldsymbol{\theta}_{j,i}^m\right) = {} & \kappa_m\left(\sum_{i\in\mathcal{N}_j} b_i x_i^m\right)\left(\sum_{i\in\mathcal{N}_j}\theta_{j,i}^m\right)^2 \\ & + \sum_{i\in\mathcal{N}_j}(1-x_i^m)(\omega_1\lambda_i + \omega_2 b_i) + \sum_{i\in\mathcal{N}_j}\left[\omega_3^{(i)}(1-x_i^m)b_i\right. \\ & \left. + \omega_4^{(i)}(1-x_i^m)\lambda_i + \omega_5^{(i)}\left(\frac{x_i^m b_i}{\theta_{j,i}^m} + \frac{\left(1-x_i^m\right)\lambda_i}{\phi_i} + \frac{\left(1-x_i^m\right)b_i}{\theta_i^t}\right)\right] \end{aligned} \tag{28}$$

$$s.t.\;\; x_i^b + x_i^m + x_i^t = 1,\; x_i^b, x_i^m, x_i^t \in \{0,1\}, \forall i \in \mathcal{N}_j, \tag{28a}$$

$$x_i^m b_i \le \theta_{j,i}^m,\ \forall i \in \mathcal{N}_j, \tag{28b}$$

$$\sum_{i\in\mathcal{N}_j} \theta_{j,i}^m \le \Theta_j^m, \tag{28c}$$

where constraints (28a)-(28c) are consistent with constraints (26a), (4) and (5).

D. CC Layer: Smart Pricing Mechanism Design and Resource Allocation Problem

The CC allocates the wired transmission resources and computing capacity to the received tasks, and designs the prices $\boldsymbol{\omega}$ for EDs and APs. The CC's energy consumption minimization and smart pricing mechanism design problem can be formulated by

$$\begin{aligned}\min_{\boldsymbol{\theta}_i^t,\boldsymbol{\phi}_i}\max_{\boldsymbol{\omega}} L^t\left(\boldsymbol{\theta}_i^t,\boldsymbol{\phi}_i,\boldsymbol{\omega}\right) &= \sum_{i=1}^{N}\left(\beta_1\left(\theta_i^t\right)^{\varepsilon}+\beta_2\right)\frac{b_i x_i^t}{\theta_i^t}\\ &+\omega_1\left(\sum_{i=1}^{N} x_i^t\lambda_i-\phi_0\right)+\omega_2\left(\sum_{i=1}^{N} x_i^t b_i-\theta_0^t\right)\\ &+\sum_{i=1}^{N}\omega_3^{(i)}\left(x_i^t b_i-\theta_i^t\right)+\sum_{i=1}^{N}\omega_4^{(i)}\left(x_i^t\lambda_i-\phi_i\right)+\sum_{j=1}^{M}\sum_{i\in\mathcal{N}_j}\omega_5^{(i)}\\ &\times\left[\frac{x_i^b b_i}{\theta_i^b}+\frac{\left(1-x_i^b\right)\lambda_i}{W\log\left(1+\frac{p_i g_{i,j}}{\sigma^2}\right)}+\frac{x_i^m b_i}{\theta_{j,i}^m}+\left(1-x_i^b-x_i^m\right)\left(\frac{\lambda_i}{\phi_i}+\frac{b_i}{\theta_i^t}\right)\right]\end{aligned} \tag{29}$$

$$s.t.\ \ x_i^b+x_i^m+x_i^t=1,\ x_i^b,x_i^m,x_i^t\in\{0,1\},\ \forall 1\le i\le N, \tag{29a}$$

$$\sum_{i=1}^{N}\phi_i\le\phi_0,\quad \sum_{i=1}^{N}\theta_i^t\le\theta_0^t, \tag{29b}$$

$$\omega_1\ge 0,\omega_2\ge 0,\omega_3^{(i)}\ge 0,\omega_4^{(i)}\ge 0,\omega_5^{(i)}\ge 0,\forall 1\le i\le N. \tag{29c}$$

Constraints (29a)-(29c) are consistent with constraints (26a), (17), (8) and (26b). The CC adjusts the prices $\boldsymbol{\omega}$ to maximize the fees charged from the EDs and APs, and adjusts the computing and transmission resource allocation $\boldsymbol{\theta},\boldsymbol{\phi}$ for smaller energy consumption.

IV. Pricing Based Multi-Layer Computation Offloading Algorithm Design

In this section, we first illustrate the diagram of pricing based multi-layer computation offloading and resource allocation,[11] as well as the signaling between different devices on different layers in the distributed HetMEC network. The energy consumption minimization approaches for the ED, AP and CC are then proposed, respectively. Finally, we summarize the whole algorithm.

A. Overview of the Pricing Based Multi-Layer Computation Offloading (P-MCO)

The energy consumption optimization is performed *round by round* according to the updated prices until the final solutions are reached within the feasible set. As illustrated in Fig. 2, in each round, the ED, AP and CC perform their optimization successively as follows.

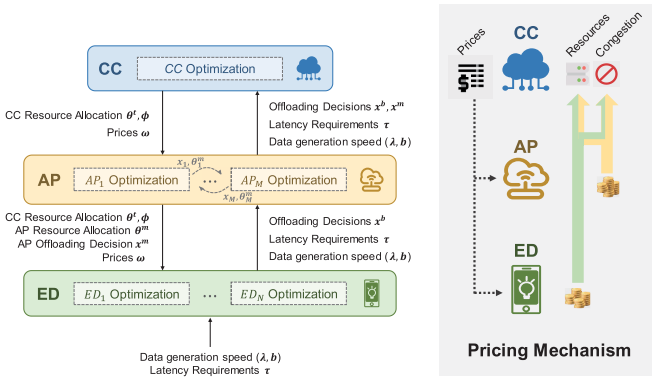


Fig. 2. Pricing based multi-layer computation offloading strategy design.

- *ED Strategy Design:* Each ED first minimizes its own cost (27) given the prices of the computing and transmission resources, as well as the network congestion and task processing timeout. The computing capacity and wireless transmission power are determined according to the computation offloading decision. If the minimum latency of local computing cannot satisfy the latency requirement of the corresponding task, the data can only be offloaded to the AP; otherwise, the ED makes the offloading decision to achieve a lower cost.
- *Signaling from ED to AP:* After the optimization of EDs, they inform the connected APs of their offloading decisions. For those offloaded tasks, the data generation speed $\boldsymbol{\lambda}$ and required CPU cycles $\boldsymbol{b}$ and remaining task processing time[12] are transmitted to the AP.
- *AP Strategy Design:* Each AP then determines its action, i.e., the computation offloading matrix $\boldsymbol{x}_i^m$ and computing capacity allocation $\boldsymbol{\theta}_i^m$. A multi-agent Q-learning method [33] is utilized to solve (28) given current data arrival speed and remaining task processing time.[13]
- *Signaling from AP to CC:* After the optimization of APs, each AP transmits its offloading decision to the CC. For those offloaded tasks, the data generation speed $\boldsymbol{\lambda}$ and required CPU cycles $\boldsymbol{b}$ and remaining task processing time are also transmitted to the CC.
- *CC Price Design and Resource Allocation:* Receiving the tasks from APs, the CC then adjusts the transmission resource allocation and computing capacity allocation. Prices for resources, as well as penalty charges for congestion and task processing timeout are updated according to the smart pricing mechanism.

[11]The time scale of the resource allocation and computation offloading is much smaller than the pace of environmentąs change.

[12]The remaining task processing time for the AP refers to latency requirement minus the transmission time of the ED.

[13]Using the multi-agent Q-learning, the state in the current round of pricing based multi-layer computation offloading (P-MCO) algorithm is influenced by the joint actions of all APs mainly via the binary variable ξ, since the data generation speed and latency requirements remains unchanged. $\xi=0$ when the maximum amount of CC's transmission resources ϕ_0 is enough, i.e., $\sum_{i=1}^{N}(1-x_i^b-x_i^m)\lambda_i\le\phi_0$; Otherwise, $\xi=1$. Therefore, the state is mainly determined by the actions of all APs.

- *Price Update and Feedback:* The CC then launches these prices to the APs and EDs, together with its adjusted computing and transmission resource allocation schemes $\boldsymbol{\theta}^t$ and $\boldsymbol{\phi}$. The AP then delivers these messages as well as its own offloading decisions and computing resource allocation to the corresponding EDs. Based on the updated information, the EDs and APs adjust their offloading decisions and resource allocation scheme. The optimization goes on round by round until the final solutions converge under the Lagrangian framework.

Being the mechanism designer, we aim to minimize the total energy consumption from a global perspective in the distributed HetMEC network where each entity handles its own Lagrangian component. However, the CC, APs, and EDs are independent entities having their own objectives. Information exchange among these cooperative entities are agreed to align their respective objectives with the global objective in the Lagrangian framework. But given that different entities belong to different parties in such a distributed network, only limited information rather than the overall information can be exchanged among different components.

B. ED Layer: Computation Offloading Algorithm Design

On the ED layer, each ED adjusts its computation offloading strategy based on the prices determined by the CC in order to minimize its cost. For the offloading strategy design of the ED, two choices are available, i.e., computing locally $x_i^b = 1$ or offloading to the AP $x_i^b = 0$. When the task is processed at the ED, the cost of ED *i* as shown in (27) can be simplified to

$$L_i^b\left(1, p_i, \theta_i^b\right) = L_i^b\left(\theta_i^b\right) = \kappa_b b_i \left(\theta_i^b\right)^2 + \omega_5^{(i)} \frac{b_i}{\theta_i^b}.$$

Proposition 1: The objective function $L_i^b(x_i^b = 1, p_i, \theta_i^b)$ of ED *i* for local computing is only related to the computing capacity θ_i^b, and the function is convex in the feasible set, and the problem can be solved by convex optimization.

Proof: The second derivative of $L_i^b(x_i^b = 1, p_i, \theta_i^b)$ is $\frac{d^2 L_i^b(\theta_i^b)}{d(\theta_i^b)^2} = 2\kappa_b b_i + 2\omega_5^{(i)} b_i (\theta_i^b)^{-3} > 0$, and thus, the objective function $L_i^b(x_i^b = 1, p_i, \theta_i^b)$ is convex. Moreover, since the feasible set $b_i \le \theta_i^b \le \theta_i^0$ is the convex set, the problem can be solved by convex optimization. ∎

When the task is offloaded to the AP, the cost of ED *i* as shown in (27) can be simplified by

$$L_i^b\left(0, p_i, \theta_i^b\right) = \frac{p_i \lambda_i}{W \log\left(1 + \frac{p_i g_{i,j}}{\sigma^2}\right)} + \omega_1 \lambda_i + \omega_2 b_i + \omega_3^{(i)} b_i + \omega_4^{(i)} \lambda_i + \omega_5^{(i)} \left[\frac{\lambda_i}{W \log\left(1 + \frac{p_i g_{i,j}}{\sigma^2}\right)} + \frac{\lambda_i}{\phi_i} + \frac{b_i}{\theta_i^t}\right].$$

Proposition 2: The objective function $L_i^b(x_i^b = 0, p_i, \theta_i^b)$ of ED *i* for an uploading task is only related to transmission power p_i, and reaches a minimum under the conditions of $\omega_5^{(i)} g_{i,j} \ge \sigma^2$.

Proof: The differential coefficient of the ED's energy consumption when $x_i^b = 0$ is

$$\frac{dL_i^b(p_i)}{d(p_i)} = \frac{\log\left(1 + \frac{p_i g_{i,j}}{\sigma^2}\right) - \ln 2 \cdot \frac{p_i g_{i,j} + \omega_5^{(i)} g_{i,j}}{p_i g_{i,j} + \sigma^2}}{\left(\log\left(1 + \frac{p_i g_{i,j}}{\sigma^2}\right)\right)^2} \quad (30)$$

Therefore, in the feasible set that $p_i \ge 0$, the numerator of $\frac{dL_i^b(p_i)}{d(p_i)}$ is incremental, while the denominator is always positive. Given that $\frac{dL_i^b(p_i)}{d(p_i)}|_{p_i=0} = -\frac{2\omega_5^{(i)} g_{i,j} - \sigma^2}{2\sigma^2 \ln 2} < 0$, and obviously $\frac{dL_i^b(p_i)}{d(p_i)}|_{p_i \to \infty} > 0$, there must exist a $p_i^* > 0$ satisfying that $\frac{dL_i^b(p_i)}{d(p_i)}|_{p_i = p_i^*} = 0$. Thus, the cost $L_i^b(p_i)$ of ED *i* reaches a minimum when $p_i = p_i^*$. ∎

Therefore, the optimal power for problem $\min L_i^b(x_i^b = 0, p_i, \theta_i^b)$ can be obtained by solving

$$\log\left(1 + \frac{p_i g_{i,j}}{\sigma^2}\right) - \ln 2 \cdot \frac{p_i g_{i,j} + \omega_5^{(i)} g_{i,j}}{p_i g_{i,j} + \sigma^2} = 0. \quad (31)$$

C. AP Layer: Multi-Agent Q-Learning Algorithm Design

On the AP layer, according to the prices determined by the CC and the offloading decisions of the connected EDs, the AP optimizes its computation offloading and computing resource allocation strategies. It is worth noting that different APs are interactive and jointly constrained by the computing and transmitting capacity of the CC. In this subsection, we solve the energy consumption minimization problems of APs in (28) via multi-agent reinforcement learning (RL).

1) Motivations of Multi-Agent Reinforcement Learning: Reinforcement learning provides a way for an agent to interact with an unknown environment, and learn the optimal solution via repeated actions and observations of the rewards [32]. In multi-agent RL (MARL), multiple agents share the same environment, aiming to maximize their individual interest [33], and the environment can only be partially observed by each agent [34].

In the distributed HetMEC network, each AP can only observe a part of the environment, i.e., the data generation speed, the historical prices and resource allocation scheme of the CC, and the offloading decisions of connected EDs. At the AP layer, different APs make decisions simultaneously, and thus, only the historical actions of other APs can be observed by each AP, while the real-time actions of other APs are invisible. Therefore, the APs need to find the optimal actions via learning in the presence of limited environment information. Therefore, MARL is well suited to solve this problem.[14] In particular, different APs aim to minimize their cost individually, and the offloading decisions of the APs are unaware of each other. However, different APs are jointly bound by the transmission resource amount and the computing capacity of the CC. Thus, their decisions are inter-dependent. By utilizing

[14]Moreover, for each AP, the energy consumption minimization problem (28) is a mixed integer nonlinear optimization problem, for which it is hard to find the optimal solution by optimization methods [30].

a MARL algorithm,[15] APs can predict the actions of each other according to the historical decisions, and obtain the optimal actions jointly by learning round by round. The instability of environment of individual agents can be well solved by the MARL. Therefore, we use MARL for solving problem (28).

2) Reinforcement Learning Framework for HetMEC: In the three-layer HetMEC network including the CC, APs and EDs, each AP is viewed as an agent. The APs decide their action according to the prices determined by the CC and the offloading decisions of EDs. We define the state, action and reward of APs in the RL framework.

- *State:* The state of the environment observed by each AP is determined by: the data generation speed of connected EDs $\Lambda_j = [\boldsymbol{\lambda}_i], \mathcal{B}_j = [\boldsymbol{b}_i], i \in \mathcal{N}_j$; the latency requirements for task processing $\mathcal{T}_j = [\boldsymbol{\tau}_i], i \in \mathcal{N}_j$; and an transmission congestion indicator $\xi \in \{0, 1\}$ representing whether the total amount of APs' required transmission resources exceeds the maximum amount of CC. Thus, the state space of AP *j* is defined by $\mathcal{S}_j = (\Lambda_j, \mathcal{B}_j, \mathcal{T}_j, \xi)$.
- *Action:* The actions of the AP are determined by the offloading decisions $\mathcal{X}_j = [\boldsymbol{x}_i^m], i \in \mathcal{N}_j$, and the computing capacity allocation $\Theta_j = [\boldsymbol{\theta}_{j,i}^m], i \in \mathcal{N}_j$. Therefore, we denote the action of AP *j* as a_j, and the action space of AP *j* is defined by $\mathcal{A}_j = [\mathcal{X}_j, \Theta_j]$. It is worth noting that the computing capacity $\theta_{j,i}^m$ is a successive variable. That is, to limit the size of the action space, the computing capacity can be discretized into multiple units.
- *Reward:* Each AP will obtain a reward *R*(*s*, *a*) given the state *s* after performing the action *a* in each iteration. We aim to minimize the costs of APs, while the RL aims to maximize the reward. Therefore, the reward function should be negatively correlated to the cost function (28) of the AP. The reward of AP *j* is defined by $R_j(s, a_j) = \frac{1}{L_j^m(s, a_j)}$.

3) Multi-Agent Q-Learning Algorithm Design for the Multiple-AP Network: We design the multi-agent *Q*-learning (MAQL) algorithm for multiple APs to coordinate their computation offloading and resource allocation while considering their mutual effects. We calculate the *Q*-values *Q*(*s*, *a*) given all state-action pairs (*s*, *a*) and record them in the *Q*-table (stored at the corresponding AP), where the *Q*-value represents the long-term reward of taking an action in the current state, considering both immediate reward and future expected reward. It is worth noting that the offloading decisions of all APs are jointly constrained by the amount of resources of the CC. To reduce the congestion cost, each AP estimates the actions of others according to their historical actions and makes its own offloading decisions for a lower cost.

(i) Action Space Discretization and Available Action Space Reduction: The action a_j of AP *j* is determined by the offloading decisions $\boldsymbol{x}_i^m$ and computing capacity $\boldsymbol{\theta}_{j,i}^m$. We define $\boldsymbol{a}_{-j}$ as the actions of all other APs except AP *j*. We discretize the computing capacities $\boldsymbol{\theta}_{j,i}^m$ into *Y* levels, and thus, the action space is discrete. In order to reduce the size of the action space, we remove the unfeasible actions according to the strategies described below.

- The allocated computing capacity θ_i^m is not smaller than the required CPU cycles $b_i(1 - x_i^b)$.
- The AP does not need to allocate computing capacity unless the task is computed by itself.
- The total allocated computing capacity of one AP to all connected EDs cannot surpass its own maximum computing capacity.

Remark 2: The upper bound of the performance loss induced by the computing capacity discretization of the AP is proportional to the square of the discretization granularity, so the loss can be significantly reduced by increasing the number of the discretization levels *Y*.

(ii) Action Selection Strategy: To achieve the trade-off between exploitation and exploration [35], the AP selects the action based on an $\epsilon-$greedy strategy. In each cycle, the AP selects a random action with the probability ϵ, aiming to explore new actions, and the AP selects the optimal action a_j^* given the *Q*-table with the probability $1 - \epsilon$. The optimal action can be expressed as

$$a_j^* = \pi_j(s) = \underset{a_j}{\operatorname{argmax}} \sum_{\boldsymbol{a}_{-j}} \frac{\Phi\big(s, \boldsymbol{a}_{-j}\big)}{n(s)} Q_j\big(s, \big(a_j, \boldsymbol{a}_{-j}\big)\big), \tag{32}$$

where the policy $\pi_j(s)$ maps the state to the optimal action, and $\Phi(s, \boldsymbol{a}_{-j})$ represents the number of actions $\boldsymbol{a}_{-j}$ of the APs other than AP *j* are selected in state *s*, and *n*(*s*) is the total number of times that state *s* is visited [35].

(iii) Q-table Updating Mechanism: After AP *j* observes the actions of other APs $\boldsymbol{a}_{-j}$, and the reward in the previous iteration, it updates the *Q*-value as follows

$$\begin{aligned} Q_j\big(s, \big(a_j, \boldsymbol{a}_{-j}\big)\big) = (1 - \eta) Q_j\big(s, \big(a_j, \boldsymbol{a}_{-j}\big)\big) \\ + \ \eta\big[R_j\big(s, \big(a_j, \boldsymbol{a}_{-j}\big)\big) + \mu V_j\big(s'\big)\big], \end{aligned} \tag{33}$$

where η represents the learning rate, and μ is a constant that satisfies $0 \leq \mu \leq 1$ representing the proportion of the future expected reward. $V_j(s') = \max_{a_j'} \sum_{\boldsymbol{a}'_{-j}} \frac{\Phi(s', \boldsymbol{a}'_{-j})}{n(s')} Q(s', (a_j', \boldsymbol{a}'_{-j}))$ represents the fact that AP *j* selects the action a_j' in new state s' to maximize the expected discount reward according to distribution of the historical actions of other APs.

Finally, we summarize the MAQL algorithm for the multiple-AP networks in *each P-MCO round* in Algorithm 1. We first initialize the *Q*-tables and action selection policies of each AP. In each iteration of the MAQL algorithm, each AP selects the action following an $\epsilon-$greedy strategy. After the APs perform their actions, the *Q*-tables are updated according to (33), and the action selection policies are updated according to (32). The *worst-case complexity* is analyzed.

Proposition 3: The **worst-case complexity** of Algorithm 1 is $O(V \cdot 2^{N_{max}} (\frac{Y}{N_{max}})^{N_{max}})$, where *Y* denotes the discretization levels of the computing capacity, and *V* implies the maximum iteration times, and $N_{max} = \max\{N_1, \ldots, N_M\}$

[15]The multi-agent multi-armed bandit [31] is not suitable for solving our problem, because it cannot describe the changes of states, and the future rewards cannot be taken into account.

Algorithm 1 Multi-Agent Q-Learning (MAQL) Algorithm for Multiple-AP Networks

Input: Data generation speed $\boldsymbol{\lambda}_i$; Require CPU cycles $\boldsymbol{b}_i$; Latency requirements $\boldsymbol{\tau}_i$; Offloading decisions of EDs $\boldsymbol{x}_i^b$; Prices $\boldsymbol{\omega}$; Historical computing and transmission resource allocation of CC $\boldsymbol{\theta}_i^t$ and $\boldsymbol{\phi}_i$; Max iteration time V.

Output: computation offloading decision $\boldsymbol{x}_i^m$, computing resources allocation scheme $\boldsymbol{\theta}_{j,i}^m$ of all APs.

Remove the unfeasible actions and reduce the action space of AP j to $\mathcal{A}_j, 1 \leq j \leq M$.
Initialize $Q(s,(a_j,\boldsymbol{a}_{-j}))$ for state $s \in \prod_{j'=1}^{M} \mathcal{S}_{j'}$ and $a_j \in \mathcal{A}_j, \boldsymbol{a}_{-j} \in \prod_{j'=1,j'\neq j}^{M} \mathcal{A}_{j'}$.
Initialize the action selection polity $\pi_j(s) = \frac{1}{|\mathcal{A}_j|}$.
for *iteration* $v < V$, *each AP* j **do**
    With probability $\epsilon^{(q)}$, AP j randomly selects an action, otherwise, it selects action optimally following $\pi_j(s)$.
    Perform the selected action a_j.
    Observe the obtained reward R_j.
    Update the Q-table according to (33) with the learning rate $\eta^{(v)} = v^{-\frac{2}{3}}$ according to the actions.
    Update the polity $\pi_j(s)$ based on (32) according to the actions in this iteration.
    $v \leftarrow v + 1$.

indicates the maximum number of EDs connected with the AP. When $N_1 = \cdots = N_M$, the worst-case complexity can be simplified into $O(V(\frac{2MY}{N})^N)$.

Proof: See the Appendix. ∎

D. CC Layer: The Resource Allocation and Price Update Mechanism Design

Based on the offloading decisions of the EDs and APs in response to previous prices, the CC updates the prices and allocates the wired transmission and computing resources. We can solve the min-max problem in (29) by solving its dual problem. It has been proved that the original problem and the dual problem are equivalent for a convex function [36]. The dual objective is

$$H(\boldsymbol{\omega}) = \min_{\boldsymbol{\theta}_i^t, \boldsymbol{\phi}_i} L^t\big(\boldsymbol{\theta}_i^t, \boldsymbol{\phi}_i, \boldsymbol{\omega}\big). \tag{34}$$

Therefore, the dual problem, i.e., the smart pricing mechanism design problem, is expressed by

$$\max_{\boldsymbol{\omega}} \ H(\boldsymbol{\omega}), \tag{35}$$

$$s.t. \ \ \omega_1 \geq 0, \omega_2 \geq 0, \omega_3^{(i)} \geq 0, \omega_4^{(i)} \geq 0, \omega_5^{(i)} \geq 0, \forall 1 \leq i \leq N, \tag{35a}$$

The resource allocation of the CC is determined by solving problem (34) with constraints (29a) and (29b), and the price updating mechanism can be obtained by solving the dual problem (35).

1) Computing and Transmission Resource Allocation of CC: We can prove that the resource allocation problem $\min_{\boldsymbol{\theta}_i^t, \boldsymbol{\phi}_i} L^t(\boldsymbol{\theta}_i^t, \boldsymbol{\phi}_i, \boldsymbol{\omega})$ is convex. Therefore, the computing and transmission resource allocation scheme can be obtained by convex optimization.

Proposition 4: The computing and transmission resource allocation problem in (34) is convex.

Proof: Given that the value $2.5 \leq \varepsilon \leq 3$, we can obtain that $\frac{d^2 L^t(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i, \boldsymbol{\omega})}{d(\phi_i^t)^2} = \lambda_i x_i^t \omega_5^{(i)} (\phi_i)^{-3} \geq 0$ and $\frac{d^2 L^t(\boldsymbol{\theta}_i^t, \boldsymbol{\phi}_i, \boldsymbol{\omega})}{d(\theta_i^t)^2} = b_i x_i^t [\beta_1(\varepsilon-1)(\varepsilon-2)(\theta_i^t)^{\varepsilon-3} + 2(\beta_2 + \omega_5^{(i)})(\theta_i^t)^{-3}] \geq 0$. Therefore, for a given price $\boldsymbol{\omega}$, function $L^t(\boldsymbol{\theta}_i^t, \boldsymbol{\phi}_i, \boldsymbol{\omega})$ is convex for the variable θ_i^t and ϕ_i. We note that constraints (29a) and (29b) are also convex. Thus, problem (34) can be solved by convex optimization. ∎

2) Price Update: The prices announced by the CC enable it to adjust the offloading strategies of the APs and EDs in an economic way. For example, when network congestion occurs, the prices for computing and transmission resources and the monetary penalty for congestion increase to motivate the APs and EDs to process more tasks locally for lower cost. The prices are updated utilizing the subgradient method [36], which can be expressed by

$$\omega[k+1] = \left[\omega[k] + l[k] \cdot \frac{\partial H(\boldsymbol{\omega})}{\partial \omega}\Big|_{\omega=\omega[k]}\right]^+, \tag{36}$$

where k is the iteration time, and $\omega[k]$ represents the price in the k-th round, and $l[k]$ is a sequence of scalar stepsizes, and $[\cdot]^+ = \max\{\cdot, 0\}$. It has been proved that the subgradient method can converge once the stepsize $l[k]$ is sufficiently small [36]. Therefore, we design the stepsize by $l[k] = \frac{\iota}{k}$, where ι is a positive constant and k is the iteration time.

E. Summary of Pricing Based Multi-Layer Computation Offloading Algorithm

The pricing based multi-layer computation offloading (P-MCO) algorithm is summarized in Algorithm 2. By executing the P-MCO algorithm, we minimize the total energy consumption in a distributed way, where EDs, APs and the CC adjust their solutions iteratively. Prices are determined by the CC in each round, to coordinate the offloading strategies of EDs and APs for lower network congestion probability.

As described in Algorithm 2, in each round, according to the prices set by the CC in the previous round, the ED individually determines its computation offloading strategy by comparing its cost of local computing and computation offloading. The offloading decisions $\boldsymbol{x}_i^b$ of all EDs are transmitted to the connected APs, together with the environment information and latency requirements. Given $\boldsymbol{x}_i^b$ and prices set by the CC, the APs obtain their computation offloading strategies and computing resource allocation schemes by Algorithm 1. According to the received offloading decisions of the EDs and APs, the CC allocates the computing resources and wired transmission resources. The prices are updated given the actions of all devices according to (36) for the next round. The loop ends when the maximum price change is smaller than threshold ξ.

V. Theoretical Analysis

In this section, we first analyze the feasible solution of the energy consumption minimization problem, and then analyze the computation offloading strategies of the APs and derive the

Algorithm 2 Pricing Based Multi-Layer Computation Offloading (P-MCO) Algorithm

Input: Data generation speed $\boldsymbol{\lambda}_i$; Require CPU cycles $\boldsymbol{b}_i$; Latency requirements $\boldsymbol{\tau}_i$.

Output: computation offloading decision $\boldsymbol{x}_i^b, \boldsymbol{x}_i^m$; Computing resources allocation scheme $\boldsymbol{\theta}_i^b, \boldsymbol{\theta}_{j,i}^m, \boldsymbol{\theta}_i^t$; Wireless transmission power $\boldsymbol{p}$ and wired transmission resource allocation $\boldsymbol{\phi}$.

Initialize maximum iteration times *MaxIter*, threshold ξ.
for $k <$ *MaxIter* **do**
  EDs minimize the cost $L_i^b(x_i^b = 1)$ via convex optimization for locally processing.
  EDs minimize the cost $L_i^b(x_i^b = 0)$ by solving equation (31) for uploading tasks.
  Compare $L_i^b(x_i^b = 1)$ and $L_i^b(x_i^b = 0)$ and select the offloading strategy x_i^b for lower cost.
  ED Signaling: The EDs transmit $\boldsymbol{x}_i^b, \boldsymbol{\lambda}, \boldsymbol{b}$ and the remaining latency requirements to the connected APs.
  APs minimize the cost by MAQL algorithm described in **Algorithm 1**.
  AP Signaling: The APs transmit $\boldsymbol{x}_i^m, \boldsymbol{x}_i^b, \boldsymbol{\lambda}, \boldsymbol{b}$ and the remaining latency requirements to the CC.
  CC optimizes the resource allocation by solving (34) utilizing convex optimization.
  CC updates the pricing according to (36).
  if $\max(|\omega(k+1) - \omega(k)|) < \xi$ **then**
    break
  CC updates the information to APs.
  APs updates the information to the connected EDs.
  $k \leftarrow k + 1$.

conditions under which the APs prefer processing all tasks to uploading them to the CC.

A. Feasible Set of the Total Energy Consumption Minimization Problem

Unlike most reinforcement learning algorithms where the obtained solution may be out of the feasible set and the feasible solution cannot be guaranteed, our algorithm guarantees that the results obtained by P-MCO algorithm are always within the feasible set.

Proposition 5: When the feasible set is nonempty, the P-MCO algorithm can always find a feasible solution.

Proof: According to the pricing based energy consumption minimization problem described in (26), the constraints that are consistent with the problem (25) can be guaranteed by the EDs, APs and CC respectively. We focus on the constraints (7), (9), (15), (17) and (25b). The feasible set is denoted by $\mathcal{F}$. For each solution *f*, we define that

$$O_1(f) = \sum_{i=1}^{N} \lambda_i x_i^t - \phi_0, \quad O_2(f) = \sum_{i=1}^{N} b_i x_i^t - \theta_0^t,$$
$$O_3^{(i)}(f) = b_i x_i^t - \theta_i^t, \quad O_4^{(i)}(f) = \lambda_i x_i^t - \phi_i,$$
$$O_5^{(i)}(f) = t_i - \tau_i,$$

and $\mathcal{O} = \{O_1, O_2, O_3^{(i)}, O_4^{(i)}, O_5^{(i)}\}$. The Lagrangian based energy consumption can be expressed by $L(f) = E(f) + \omega_1 O_1(f) + \omega_2 O_2(f) + \sum_{i=1}^{N} [\omega_3^{(i)} O_3^{(i)}(f) + \omega_4^{(i)} O_4^{(i)}(f) + \omega_5^{(i)} O_5^{(i)}(f)]$.

When the solution is within the feasible set, i.e., $f \in \mathcal{F}$, we also consider to update price ω_1. Since $O_1(f) \leq 0$ and given that $f \in \mathcal{F}$, we note that $E(f) + \omega_1 O_1 \leq E(f)$. According to the price updating mechanism (36), after the *k*-th iteration, the updated price is expressed by $\omega_1[k+1] = [\omega_1[k] + l[k] \cdot O_1(f)]^+ \leq \omega_1[k]$. Therefore, $E(f) + \omega_1 O_1(f)$ increases with the iteration times and towards *E(f)*. Similarly, we can obtain that $L(f)|_{k\to\infty} = E(f)$.

When the solution is out of the feasible set, i.e., $f' \notin \mathcal{F}$, we consider the updating of price ω_1. Since $O_1(f') > 0$ given that $f' \notin \mathcal{F}$, we note that $E(f') + \omega_1 O_1 > E(f')$. According to the price updating mechanism (36), after the *k*-th iteration, the updated price is expressed by $\omega_1[k+1] = [\omega_1[k] + l[k] \cdot \frac{\partial H(\boldsymbol{\omega})}{\partial \omega_1}|_{\omega_1=\omega_1[k]}]^+ = \omega_1[k] + l[k] O_1(f') > \omega_1[k]$. Therefore, $E(f) + \omega_1 O_1(f')$ increases with the iteration times and towards infinity. Similarly, we can obtain that $L(f')|_{k\to\infty} = \infty$. Hence, when minimizing the Lagrangian based energy consumption *L*, the final solution is always within the feasible set $\mathcal{F}$ when $\mathcal{F} \neq \emptyset$. ■

B. Computation Offloading Strategy of the AP

We now investigate the offloading strategy of the AP. When the required number of CPU cycles exceeds the computing capacity of the EDs, i.e.,

$$b_i > \Theta_i^b, \forall i \in \mathcal{N}_j, \tag{37}$$

the EDs offload the tasks to the connected AP *j*. Let p_i^* denote the optimal transmission power of ED *i*. Three sufficient conditions that ensure AP *j* chooses to process all the tasks itself rather than to upload to the CC are then given below.

First, the energy consumption at the AP is smaller than that at the CC, expressed by

$$\kappa_m b_i \left(\theta_{j,i}^m\right)^2 < \left(\beta_1 \left(\theta_i^t\right)^{\varepsilon} + \beta_2\right) \cdot \frac{b_i}{\theta_i^t}. \tag{38}$$

According to the pricing based energy consumption minimization problem for AP shown in (28), when the energy consumption at the CC is larger, the CC can raise the prices ω_4 and ω_5 to encourage the AP to process the task itself for lower cost.

Second, the computing capacity of the AP is sufficient for processing the tasks without congestion, which can be expressed by

$$\Theta_j^m \geq \sum_{i \in \mathcal{N}_j} b_i. \tag{39}$$

Third, the task processing latency of the tasks should satisfy their latency requirements when processed at the AP. The minimum required computing capacity for processing task *i* is expressed by $\frac{b_i}{\tau_i - (t_{i,tran}^b)^*}$, where $(t_{i,tran}^b)^* = \frac{\lambda_i}{W \log(1 + p_i^* g_{ij}/\sigma^2)}$ implies the transmission duration from the ED to the AP. Therefore, the total required computing capacity cannot surpass the maximum computing capacity θ_0^m of the AP, described by

$$\theta_0^m \geq \sum_{i \in \mathcal{N}_j} \frac{b_i}{\tau_i - \left(t_{i,tran}^b\right)^*}. \tag{40}$$

TABLE II
HETMEC NETWORK PARAMETERS

Parameters	Value	Parameters	Value
CPU clock frequency of each ED	1 GHz	AP energy consumption coefficient κ_m	10^{-19}
CPU clock frequency of each AP	4 GHz	CC energy consumption coefficient β_1	$2 * 10^{-19}$
CPU clock frequency of CC	50 GHz	CC energy consumption coefficient β_2	7
Maximum transmission power of ED	200 mW	CC energy consumption coefficient ε	3
Transmission resource amount of CC	5 Mbps	ED energy consumption coefficient κ_b	10^{-19}
Noise density	10^{-11}mW/Hz	Bandwidth of the wireless channel	200KHz

Based on the above three statements, when the task processing latency requirements $\tau_i, i \in \mathcal{N}_j$, satisfy the constraint (40) under the conditions (37)-(39), AP j will process the task locally.

VI. SIMULATION RESULTS

In this section, we evaluate the total energy consumption and latency performance of the whole network when executing the P-MCO algorithm. We first verify the convergence of the MAQL algorithm and the P-MCO algorithm by the simulation. The total energy consumption and network congestion probability are then evaluated, compared to the traditional distributed method [37] with no pricing mechanism. In the *traditional distributed* method [37], the APs determine the actions via RL in a distributed manner. Specifically, each ED decides to offload or to compute locally for lower energy consumption, and CC allocates the transmission and computing resources according to the decisions of EDs and APs.

A. Parameters Setting

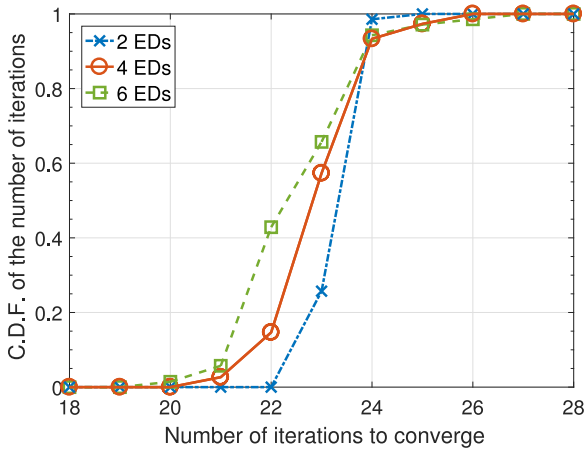
We consider a HetMEC network consisting of one CC, $M = 10$ APs[16] and $N = 60$ EDs. The parameters are listed in Table II according to [14], [27], [28]. We assume that the EDs and APs are randomly located with a distance within 100m. The EDs are the smartphones with single-core 1 GHz of maximum CPU clock frequency, and the AP is the IBM server with a quad-core 4 GHz CPU. The CC consists of 10 quad-core 5 GHz CPUs, so the equivalent CPU clock frequency of the CC is 50 GHz [27], [28]. The required number of CPU cycles is proportional to the amount of raw data for the same kind of tasks [38].

The performance of the P-MCO algorithm is evaluated based on the following metrics

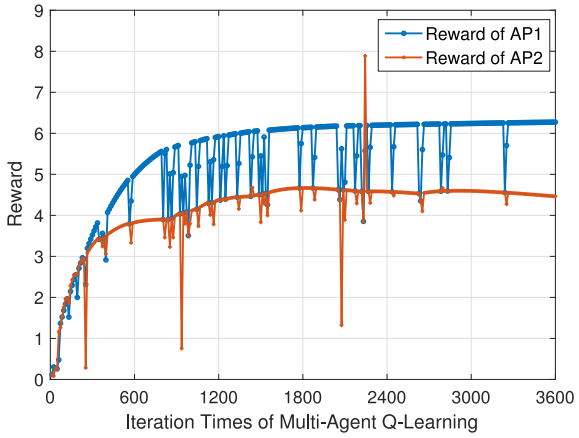
- *Total Energy Consumption:* The total energy consumption of all EDs, APs and the CC.
- *Congestion Probability:* The probability that the network experiences congestion.

B. Convergence Performance

We illustrate the convergence performance of the P-MCO and MAQL algorithms in Fig. 3. Fig. 3(a) presents the cumulative distribution function (CDF) of the number of iterations to converge for the P-MCO algorithm given different numbers of EDs. We observe that the outcomes of the P-MCO algorithm all converge within 27 iterations in different cases. As the number of EDs increases, the range of iteration numbers to converge also increases. The most frequent iteration number to converge is 24, 23, and 22 when the number of EDs is 2, 4, and 6, respectively. Fig. 3(b) shows the reward of each AP versus the iterations of the MAQL algorithm in a round of P-MCO algorithm. Each AP searches for the best solution and reinforces its learning results as the number of iteration times increases. The sudden changes of the reward results from the random action selected in those iterations. Since the offloading decisions of connected EDs are different for APs, their rewards converge to different values.



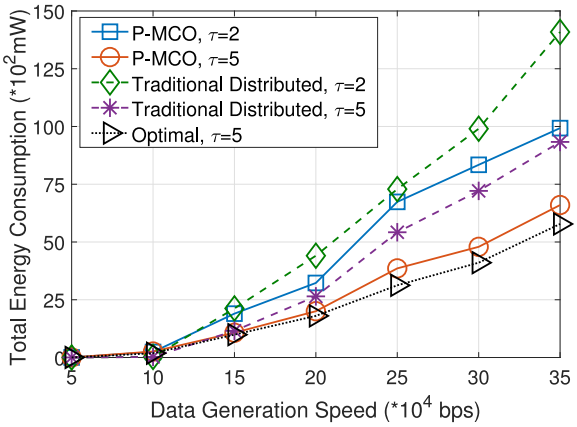
(a) Convergence performance of P-MCO algorithm.



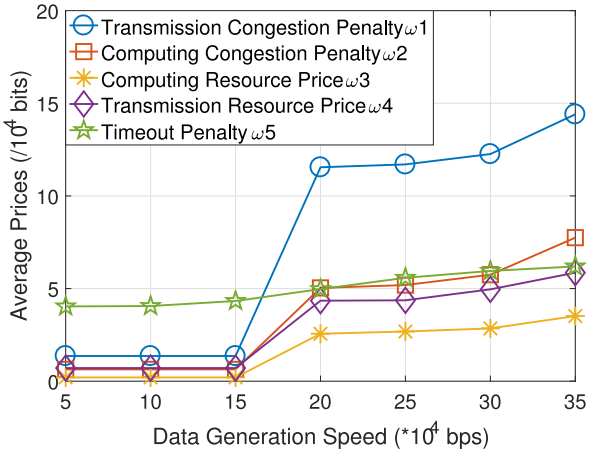
(b) Convergence performance of MAQL algorithm.

Fig. 3. Algorithm Convergence Performance.

[16]Our proposed algorithm is scalable for multiple APs. For the ultra-dense networks, multi-layer APs can be considered for the scalability, where MARL can be performed within each group of lower-layer APs connected with the same upper-layer AP.

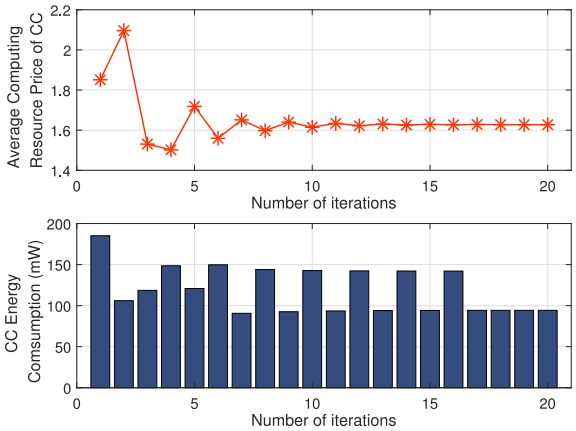


(a) Total power consumption v.s. data generation speed

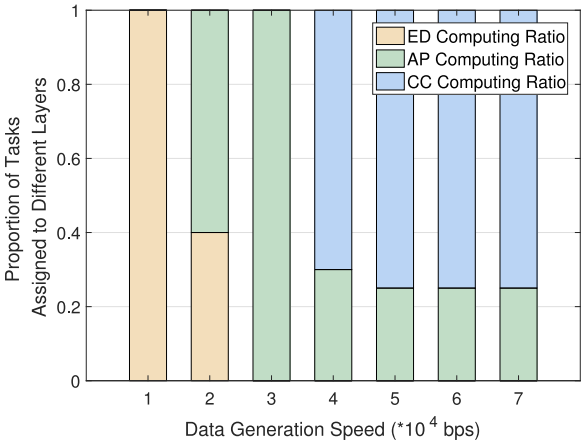


(b) Average prices by performing P-MCO algorithm v.s. data generation speed when $\tau = 5$

Fig. 4. Energy consumption performance v.s. the change of prices.



(a) Average computing resource price and energy consumption of the CC varying the number of iterations.



(b) Proportion of Tasks assigned to different layers by performing P-MCO algorithm v.s. data generation speed

Fig. 5. Influence of the price on energy consumption and computation offloading results analysis.

C. Energy Consumption Performance

Fig. 4(a) shows the total energy consumption versus the data generation speed in different cases. We assume that one CPU cycle is required for processing each bit of raw data, i.e., the computation load also increases with the data generation speed. Given the latency requirement for task processing, the total energy consumption increases with the data generation speed, as more data needs to be processed. When the required latency decreases, the total energy consumption grows, since larger computing capacity is utilized and more transmission power is required for reducing the task processing latency. We compare our proposed P-MCO algorithm with the traditional distributed method [37]. Our proposed algorithm achieves around 28% lower energy consumption[17] , because the offloading strategies are adjusted by prices for reducing the total energy consumption, implying the significance of the pricing mechanism for energy saving. We observe that the performance gap to the optimal solution is smaller than 14%, which is induced by distributed decision making and the computing capacity's discretization.

Fig. 4(b) presents the final prices determined by the CC given different data generation speeds when the latency requirement $\tau = 5$ time periods. For a given data generation speed λ, the prices are updated round by round to coordinate the actions of EDs, APs and CC. We note that the average prices increase with the data generation speed, so as to encourage more EDs and APs to process the task themselves for lower latency and network congestion probability. Since no task is offloaded to the CC when $\lambda \leq 15$, the resources of the CC are not utilized. Thus, the CC's resource prices (i.e., ω_3 and ω_4) and congestion penalties (i.e., ω_1 and ω_2) remain unchanged. When $\lambda > 15$, these prices increase significantly to discourage tasks being offloaded to the CC.

Fig. 5(a) illustrates how the average price of computing resources ω_3 influences the energy consumption of the CC. During the iterations, the CC updates the prices based on the offloading decisions of the EDs and APs, and the prices in turn influence their strategies. We note that when the price ω_3 increases, the CC's energy consumption decreases, as more

[17]We note that when data generation speed $\lambda = 10 * 10^4$ bps, the energy consumption gained by the P-MCO is slightly larger than that gained by the traditional distributed method, because the computation offloading strategy is adjusted in P-MCO so as to satisfy the latency requirements of task processing (which can be seen in Fig. 7 when $\lambda = 10$).

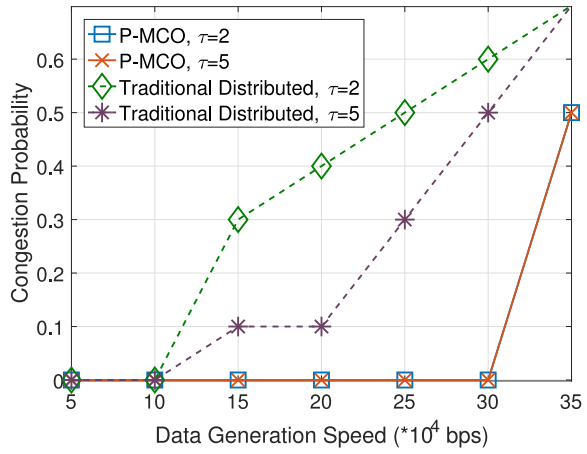


Fig. 6. Congestion Probability v.s. data generation speed.

tasks are encouraged to be processed at the EDs and APs. When the resource price ω_3 reduces, it provides the incentive for EDs and APs to upload more tasks, and thus, the energy consumption of the CC increases.

Fig. 5(b) shows the task assignment results of our P-MCO algorithm given different data generation speeds. As the data load increases, the tasks are gradually offloaded to upper layers. When $\lambda > 2$, EDs do not process tasks anymore due to limited computing capacity. When $\lambda = 3$, tasks are all processed at the APs as analyzed in Section V. When $\lambda > 3$, the AP cannot handle all tasks without congestion, so the CC shares the computation load. The computing capacities of multiple devices are fully used to save energy as well as to reduce the network congestion.

D. Network Congestion Performance

Fig. 6 presents the network congestion probability v.s. the data generation speed given different latency requirements. With our P-MCO algorithm, the network remains non-congested until the data generation speed reaches $\lambda = 35$. However, in the traditional distributed method, the network exhibits congestion after $\lambda > 10$, and the congestion probability increases gradually with the data generation speed. When the smart pricing mechanism operates, prices increase with the data generation speed. Thus, the offloading cost grows, motivating more EDs and APs to process tasks locally. Therefore, the computation load is distributed and network congestion can be alleviated.

VII. Conclusion

We studied the energy consumption minimization problem in distributed HetMEC networks. A pricing based multi-layer computation offloading (P-MCO) strategy is designed to coordinate the offloading behaviors and resource allocation schemes of the EDs, APs and CC, so that the network congestion can be alleviated while minimizing the total energy consumption. In the P-MCO algorithm, the CC updates the prices according to the real-time data generation speed of tasks, and adjusts the resource allocation scheme based on the offloading decisions of the EDs and APs. Given the prices, the EDs make their offloading decisions via convex optimization, and the APs utilize a MAQL algorithm for the computation offloading and computing resource allocation. The feasibility of the obtained solutions is guaranteed if the feasible set is not empty, and the timeout percentage is reduced when no feasible solution can be obtained.

Simulation results show that when performing the proposed P-MCO algorithm the total energy consumption decreases by 28% and a maximum 100% network congestion reduction is achieved compared with the traditional distributed method. As the computation load grows, resource prices and monetary penalty increase to discourage EDs and APs to upload tasks, which in turn alleviates the network congestion.

Appendix
Proof of Proposition 3

When performing Algorithm 1, the maximum number of AP's actions is $(2\,Y)^{N_{max}}$ after the discretization of the computing capacity, where Y denotes the number of the discretization levels and $N_{max} = \max\{N_1, \ldots, N_M\}$ denotes the maximum number of EDs connected with each AP. Benefit from the action space reduction described in *Section IV-C-3)-(i)*, the maximum number of actions can be reduced. The total allocated computing capacity of each AP cannot surpass its own maximum computing capacity, i.e., $\sum_{i\in\mathcal{N}_j} y_{j,i}^m \leq Y$, where $y_{j,i}$ denotes the discretized computing capacity of ED i. Therefore, according to the inequality of arithmetic and geometric means, we can obtain that $\prod_{i\in\mathcal{N}_j} y_{j,i} \leq (\frac{\sum_{i\in\mathcal{N}_j} y_{j,i}}{N_j})^{N_j} = (\frac{Y}{N_j})^{N_j}$. Therefore, the maximum number of actions for each AP is expressed by $2^{N_{max}}(\frac{Y}{N_{max}})^{N_{max}}$, which can be simplified into $(\frac{2MY}{N})^{\frac{N}{M}}$ when $N_1 = N_2 = \cdots = N_M = N_{max} = \frac{N}{M}$.

In each iteration, the complexity of obtaining the optimal action is $O(V \cdot 2^{N_{max}}(\frac{Y}{N_{max}})^{N_{max}})$ according to (32), and the complexity of Q-table updating is $O(M)$. Therefore, the worst-case complexity of multi-agent Q-learning algorithm is $O(V \cdot 2^{N_{max}}(\frac{Y}{N_{max}})^{N_{max}})$, where V implies the maximum iteration times. When $N_1 = N_2 = \cdots = N_M$, the worst-case complexity of Algorithm 1 can be simplified into $O(V(\frac{2MY}{N})^{\frac{N}{M}})$.

References

[1] P. Wang, B. Di, Z. Zheng, and L. Song, "Distributed energy saving for heterogeneous multi-layer mobile edge computing," in *Proc. IEEE ICC*, Dublin, Ireland, Jun. 2020, pp. 1–6.
[2] Y. Yu, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges," *China Commun.*, vol. 13, no. 2, pp. 89–99, 2016.
[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," ETSI, Sophia Antipolis, France, Rep. 11, Sep. 2015.
[4] P. Wang, Z. Zheng, B. Di, and L. Song, "HetMEC: Latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4942–4956, Oct. 2019.
[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[6] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[7] Q. Liu, T. Han, and N. Ansari, "Joint radio and computation resource management for low latency mobile edge computing," in *Proc. IEEE GLOBECOM*, Abu Dhabi, UAE, Dec. 2018, pp. 1–7.

[8] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, Sep. 2021.

[9] P. Wang, C. Yao, Z. Zheng, G. Sun, and L. Song, "Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2872–2884, Apr. 2019.

[10] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "Smart data pricing: Using economics to manage network congestion," *Commun. ACM*, vol. 58, no. 12, pp. 86–93, 2015.

[11] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Optimal pricing-based edge computing resource management in mobile blockchain," in *Proc. IEEE ICC*, Kansas City, MO, USA, 2018, pp. 1–6.

[12] L. Li, M. Siew, and T. Q. S. Quek, "Learning-based pricing for privacy-preserving job offloading in mobile edge computing," in *Proc. IEEE ICASSP*, Brighton, U.K., May 2019, pp. 4784–4788.

[13] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[14] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[15] J. Zheng, Y. Cai, Y. Wu, and X. S. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[16] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Proc. IEEE/ACM IWQoS*, Vilanova i la Geltrú, Spain, Jun. 2017, pp. 1–10.

[17] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. ACM MSWiM*, Cancun, Mexico, Nov. 2015, pp. 271–278.

[18] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[19] T. Zhang, "Data offloading in mobile edge computing: A coalition and pricing based approach," *IEEE Access*, vol. 6, pp. 2760–2767, 2018.

[20] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, Apr. 2018, pp. 207–215.

[21] F. Zhang and M. M. Wang, "Stochastic congestion game for load balancing in mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 778–790, Jan. 2021.

[22] E. Hossain, M. Rasti, H. Tabassum, and A. Abdelnasser, "Evolution toward 5G multi-tier cellular wireless networks: An interference management perspective," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 118–127, Jun. 2014.

[23] P. Barham *et al.*, "Xen and the art of virtualization," in *Proc. ACM Symp. Oper. Syst. Principles (SOSP)*, Bolton Landing, NY, USA, Oct. 2003, pp. 164–177.

[24] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[25] B. Sklar, "Rayleigh fading channels in mobile digital communication systems .I. Characterization," *IEEE Commun. Mag.*, vol. 35, no. 7, pp. 90–100, Jul. 1997.

[26] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall PTR, 1996.

[27] L. Rao, X. Liu, M. D. Ilic, and J. Liu, "Distributed coordination of Internet data centers under multiregional electricity markets," *Proc. IEEE*, vol. 100, no. 1, pp. 269–282, Jan. 2012.

[28] T. D. Burd, and R. W. Broderson, "Processor design for portable systems," *J. VLSI Singapore Process.*, vol. 13, nos. 2–3, pp. 203–221, Aug. 1996.

[29] R. T. Rockafellar, "Augmented lagrange multiplier functions and duality in nonconvex programming," *SIAM J. Control*, vol. 12, no. 2, pp. 268–285, 1974.

[30] C. A. Floudas, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. New York, NY, USA: Oxford Univ. Press, 1995.

[31] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, U.K.: MIT Press, 2018.

[32] J. Hu, H. Zhang, and L. Song, "Reinforcement learning for decentralized trajectory design in cellular UAV networks with sense-and-send protocol," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6177–6189, Aug. 2019.

[33] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Auton. Agents Multi-Agent Syst.*, vol. 11, no. 3, pp. 387–434, Nov. 2015.

[34] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Dept. Psychol., Kings Coll., Cambridge, U.K., 1989.

[35] E. R. Gomes and R. Kowalczyk, "Dynamic analysis of multiagent Q-learning with $\epsilon-$greedy exploration," in *Proc. ACM ICML*, Montreal, QC, Canada, Jun. 2009, pp. 369–376.

[36] W. Yu and R. Lui, "Dual methods for nonconvex spectrum optimization of multicarrier systems," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1310–1322, Jul. 2006.

[37] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," 2017, *arXiv:1711.09012*.

[38] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX Conf. Hot Topics Cloud Comput. (HotCloud)*, Jun. 2010, pp. 1–7.

Pengfei Wang (Graduate Student Member, IEEE) received the B.S. degree in electronics engineering from Peking University, Beijing, China, in 2017, where he is currently pursuing the master's degree with the Department of Electronics. His current research interest includes wireless communications, vehicular networks, and edge computing.

Boya Di (Member, IEEE) received the B.S. degree in electronic engineering from Peking University, China, in 2014, and the Ph.D. degree from the Department of Electronics, Peking University in 2019. She was a Postdoctoral Researcher with Imperial College London and is currently an Assistant Professor with Peking University. Her current research interests include reconfigurable intelligent surfaces, multiagent systems, edge computing, vehicular networks, and aerial access networks. She received the Best Doctoral Thesis Award from China Education Society of Electronics in 2019. She is also the recipient of 2021 IEEE ComSoc Asia–Pacific Outstanding Paper Award. She serves as an Associate Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY since June 2020. She has also served as the Workshop Co-Chair for IEEE WCNC 2020&2021.

Lingyang Song (Fellow, IEEE) received the Ph.D. degree from the University of York, U.K., in 2007, where he received the K. M. Stott Prize for Excellent Research. He worked as a Research Fellow with the University of Oslo, Norway, until rejoining Philips Research U.K. in March 2008. In May 2009, he joined the Department of Electronics, School of Electronics Engineering and Computer Science, Peking University, and is currently a Boya Distinguished Professor. His main research interests include wireless communication and networks, signal processing, and machine learning. He was the recipient of the IEEE Leonard G. Abraham Prize in 2016 and the IEEE Asia–Pacific Young Researcher Award in 2012. He has been an IEEE Distinguished Lecturer since 2015.

Nicholas R. Jennings (Fellow, IEEE) is the Vice-Chancellor and the President of Loughborough University. He is an internationally-recognised authority in the areas of AI, autonomous systems, cyber-security, and agent-based computing. He is a member of the U.K. government's AI Council, the governing body of the Engineering and Physical Sciences Research Council, and the Chair of the Royal Academy of Engineering's Policy Committee. Before Loughborough, he was the Vice-Provost for Research and Enterprise and a Professor of Artificial Intelligence with Imperial College London, the U.K.'s first Regius Professor of Computer Science (a post bestowed by the monarch to recognise exceptionally high quality research) and the U.K. Government's first Chief Scientific Advisor for National Security.