A Divide-and-Conquer Bilevel Optimization Algorithm for Jointly Pricing Computing Resources and Energy in Wireless Powered MEC

Pei-Qiu Huang, Yong Wang, *Senior Member, IEEE*, and Kezhi Wang, *Senior Member, IEEE*

***Abstract*—This paper investigates a wireless powered mobile edge computing (MEC) system, where the service provider (SP) provides the device owner (DO) with both computing resources and energy to execute tasks from Internet of Things devices. In this system, SP first sets the prices of computing resources and energy whereas DO then makes the optimal response according to the given prices. In order to jointly optimize the prices of computing resources and energy, we formulate a bilevel optimization problem (BOP), in which the upper level generates the prices of computing resources and energy for SP and then under the given prices, the lower level optimizes the mode selection, broadcast power, and computing resource allocation for DO. This BOP is difficult to address due to the mixed variables at the lower level. To this end, we first derive the relationships between the optimal broadcast power and the mode selection and between the optimal computing resource allocation and the mode selection. After that, it is only necessary to consider the discrete variables (i.e., mode selection) at the lower level. Note, however, that the transformed BOP is still difficult to solve because of the extremely large search space. To solve the transformed BOP, we propose a divide-and-conquer bilevel optimization algorithm (called DACBO). Based on device status, task information, and available resources, DACBO first groups tasks into three independent small-size sets. Afterward, analytical methods are devised for the first two sets. As for the last one, we develop a nested bilevel optimization algorithm that uses differential evolution and variable neighborhood search (VNS) at the upper and lower levels, respectively. In addition, a greedy method is developed to quickly construct a good initial solution for VNS. The effectiveness of DACBO is verified on a set of instances by comparing with other algorithms.**

***Index Terms*—Bilevel optimization, divide-and-conquer, mobile edge computing, differential evolution, variable neighborhood search**

I. Introduction

Internet of Things (IoT) equips massive limited-power devices with software, sensors, and network connectivity, with the aim of bringing the vision of a connected world into reality [1], [2]. However, IoT devices (IoTDs) cannot support some emerging applications, such as face recognition [3] and virtual reality [4], since these applications usually need to be completed in a short time and require a lot of computing resources and energy. Despite the growing capabilities of IoTDs, their computing and battery capabilities are still not sufficient due to physical size limitations, which poses a great challenge to execute tasks on IoTDs [5].

Mobile cloud computing (MCC) can relieve the above challenge by offloading tasks from IoTDs to MCC servers [6]. But MCC servers are typically far from IoTDs geographically, which gives rise to high latency and energy consumption for long-distance transmission. As an alternative, mobile edge computing (MEC) deploys computing resources at the edge of networks to avoid long-distance transmission, thereby reducing latency and prolonging the battery life of IoTDs [7]. However, it is still impossible to avoid manually charging or replacing batteries of IoTDs due to the finite battery capacity. In some scenarios, it is not a trivial task. For example, replacing the batteries of IoTDs located in remote areas, such as deserts and forests, requires high costs. Recently, wireless power transfer has been considered as a promising solution to continuously supply energy for IoTDs, which enables IoTDs to harvest energy from radio frequency (RF) signals radiated by radio frequency energy transmitters into the air [8], [9]. As a result, wireless powered MEC (WP-MEC) that combines MEC with wireless power transfer can provide both computing resources and energy to IoTDs [10], [11].

Because MEC servers are resource-constrained compared with MCC servers, effective resource management is critical for MEC systems [12], which generally includes two key techniques: mode selection and resource allocation. The former indicates whether or where to execute tasks, while the latter is to determine how many resources are allocated [5]. To date, a lot of methods have been designed. For instance, Chen *et al.* [13] adopted game theory method for the mode selection for a multi-channel MEC system. Pu *et al.* [14] developed a novel method for the mode selection in a device-to-device enabled MEC system, where idle IoTDs act as MEC servers to provide computing resources to nearby IoTDs. Considering time-varying network dynamics, Chen *et al.* [15] applied deep reinforcement learning to solve the mode selection. Due to the fact that both mode selection and resource allocation are related to the quality of service, Lyu *et al.* [16] researched joint mode selection and resource allocation problem for minimizing energy consumption. Wang *et al.* [17] studied a multi-unmanned aerial vehicle (multi-UAV) assisted MEC system and then proposed a bilevel optimization algorithm to jointly optimize the UAV deployment, mode selection, and resource allocation. In addition, some researches have

This work was supported by the National Natural Science Foundation of China under Grants 61976225. (Corresponding authors: Yong Wang.)

P.-Q. Huang and Y. Wang are with the School of Automation, Central South University, Changsha 410083, China (Email: pqhuang@csu.edu.cn, ywang@csu.edu.cn)

K. Wang is with the Department of Computer and Information Sciences, Northumbria University, Newcastle NE1 8ST, UK. (kezhi.wang@northumbria.ac.uk)

also designed resource management methods for WP-MEC systems. Bi *et al.* [18] employed the alternating direction method of multipliers to jointly optimize the mode selection and transmission time allocation for a WP-MEC system. Zhou *et al.* [19] studied joint mode selection and resource allocation problem for a UAV-enabled WP-MEC system. Du *et al.* [20] further considered the workflow scheduling for improving the energy efficiency of a UAV. Moreover, joint mode selection and resource allocation for WP-MEC systems are investigated in [21] to minimize the energy consumption and in [22] to maximize the residual energy.

However, these studies [13]–[22] assume that the service provider (SP) and the device owner (DO) belong to the same entity and they have the common objective. In fact, in complex MEC systems, multiple entities may be involved and have different objectives. For example, the objective of SP may be to maximize its profit or balance the load among MEC servers [23], while DO aims to minimize the energy consumption or the completion time [11]. Under this condition, the above methods are no longer applicable. To this end, some attempts have been made to use pricing schemes for resource management [24]–[26]. Pricing schemes are able to model and analyze complex interactions among different entities. Through these interactions, each entity can observe, learn, or predict actions or states of other entities and then make autonomous decisions to achieve the optimal resource allocation [27]. For instance, Wang *et al.* [28] analyzed the pricing scheme for computing resources in a UAV-enabled MEC system. Xiong *et al.* [29] optimized the price of computing resources for a MEC-assisted blockchain network. Wang *et al.* [30] studied the optimal price-based energy allocation algorithm. Han *et al.* [31] explored the pricing scheme in a vehicle assisted-MEC system. It is worth noting that these methods only study the price optimization of computing resources. For WP-MEC systems, DO needs to purchase not only computing resources but also energy from SP; thus, it is necessary to jointly optimize the prices of computing resources and energy.

In this paper, we study the price optimization of computing resources and energy in a WP-MEC system, where DO purchases computing resources and energy from SP to execute tasks from IoTDs. With the aim of maximizing their respective total profits, SP optimizes the prices of computing resources and energy, while DO optimizes the mode selection, broadcast power, and computing resource allocation under the given prices. To the best of our knowledge, it is the first attempt to jointly price computing resources and energy in a WP-MEC system. The main contributions of this paper are summarized as follows:

1) Based on the interaction between SP and DO, we formulate the price optimization of computing resources and energy as a bilevel optimization problem (BOP).
2) We derive the relationships between the optimal broadcast power and the mode selection and between the optimal computing resource allocation and the mode selection. As a result, the mixed-variable nonlinear optimization problem at the lower level is simplified to a discrete optimization problem.
3) A divide-and-conquer bilevel optimization algorithm (called DACBO) is devised to address the transformed BOP. DACBO groups tasks into three independent sets according to device status, task information, and available resources.
4) Analytical methods are developed for the first two sets. Regarding the last one, we design a nested bilevel optimization algorithm (called DE-VNS), which combines differential evolution (DE) with variable neighborhood search (VNS).
5) Extensive experiments have been conducted on a set of instances. The results demonstrate the effectiveness of DACBO by comparing it with other algorithms.

The rest of this paper is organized as follows. Section II introduces the background about bilevel optimization. The system model and problem formulation are given in Section III. Section IV describes the details of our proposed DACBO. The experimental studies are presented in Section V. Finally, Section VI concludes this paper.

II. Background

A. BOPs

Many optimization problems are hierarchical, where the performance of the solution produced by the upper-level entity is affected by the response of the lower-level entity. We refer to such optimization problems as BOPs [32]–[35], which can be formulated as follows [36]:

$$\begin{aligned} \min_{\mathbf{x}^u,\mathbf{x}^l} \ & f^u(\mathbf{x}^u,\mathbf{x}^l) \\ \text{s.t. } & g_j^u(\mathbf{x}^u,\mathbf{x}^l)\le 0, j=1,\dots,p^u \\ & \mathbf{x}^l=\arg\min_{\mathbf{x}^l} f^l(\mathbf{x}^u,\mathbf{x}^l) \\ & \quad \text{s.t. } g_j^l(\mathbf{x}^u,\mathbf{x}^l)\le 0, j=1,\dots,p^l \end{aligned} \tag{1}$$

where $\mathbf{x}^u=(x_1^u,\dots,x_{n^u}^u)$ and $\mathbf{x}^l=(x_1^l,\dots,x_{n^l}^l)$ represent the upper-level and lower-level solutions, respectively; x_i^u and x_i^l denote the ith upper-level variable and the ith lower-level variable, respectively; n^u and n^l denote the numbers of upper-level and lower-level variables, respectively; $f^u(\mathbf{x}^u,\mathbf{x}^l)$ and $f^l(\mathbf{x}^u,\mathbf{x}^l)$ represent the upper-level and lower-level objective functions, respectively; $g_j^u(\mathbf{x}^u,\mathbf{x}^l)$ and $g_j^l(\mathbf{x}^u,\mathbf{x}^l)$ represent the jth upper-level and the jth lower-level constraints, respectively; and p^u and p^l denote the numbers of upper-level and lower-level constraints, respectively.

Definition 1: A solution $\mathbf{x}=(\mathbf{x}^u,\mathbf{x}^l)$ is called a feasible solution of a BOP if and only if:

1) $\mathbf{x}$ satisfies all constraints at both levels;
2) $\mathbf{x}^l$ is the optimal lower-level solution corresponding to $\mathbf{x}^u$ [1].

Definition 2: A solution $\mathbf{x}^*=(\mathbf{x}^{u*},\mathbf{x}^{l*})$ is the optimal solution of a BOP if it is the feasible solution with the smallest value of the upper-level objective function .

[1]For some complex BOPs, the lower-level optimum may be difficult to obtain. In this case, the near-optimal solution of the lower-level optimization problem is also acceptable.

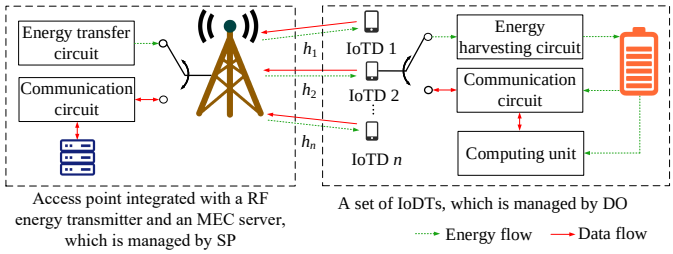

Fig. 1. WP-MEC system involving a set of n IoTDs and an access point.

B. Bilevel Optimization Algorithms

The traditional bilevel optimization algorithms usually adopt Karush-Kuhn-Tucker or other optimality conditions to transform BOPs into single-level optimization problems. Representative studies include branch and bound methods [37], simplex methods [38], and penalty function methods [39]. However, these methods are only effective for simple BOPs, such as linear or convex BOPs [36].

Because heuristic methods have the capability to address complex optimization problems, many attempts have been made on them for addressing BOPs. They can be roughly divided into two categories: (1) single-level reduction-based approaches and (2) nested structure-based approaches. In single-level reduction-based approaches, BOPs are transformed into single-level optimization problems and then solved by heuristic methods [40], [41]. In this kind of approach, it often requires strong assumptions on the mathematical properties of the lower-level optimization problems, such as linearity, convexity, or smoothness [36]. To solve more complex BOPs, nested structure-based approaches have been studied, which perform the upper-level and lower-level optimization in a nested manner [42], [43]. Since the assumptions on the mathematical properties of BOPs are not required, nested structure-based approaches are widely used [44], [45]. However, when the lower-level optimization problem of a BOP is highly complex, these approaches may suffer from low computational efficiency as the lower-level optimization usually needs to be performed for each upper-level solution [46].

III. System Model and Problem Formulation

As shown in Fig. 1, we consider a WP-MEC system involving an access point and a set of n IoTDs, which are managed by SP and DO, respectively. Since the access point is equipped with an MEC server and a RF energy transmitter, this WP-MEC system can provide computing and wireless charging services for IoTDs. In addition, each IoTD has a task waiting to be executed, denoted as $\mathcal{N} = \{1, \cdots, n\}$. For the sake of simplicity, we define the task of the ith IoTD by a 3-tuple: $U_i = (D_i, C_i, T_{i,max})$, where D_i and C_i represent the size of input data of U_i and the computing resources required to complete U_i, respectively; and $T_{i,max}$ indicates the maximum time allowed to complete U_i. In this WP-MEC system, DO needs to select one of three execution modes for each task, which are defined as

1) $m_i = -1$ $(i \in \mathcal{N})$, if U_i is executed locally (i.e., the local mode);

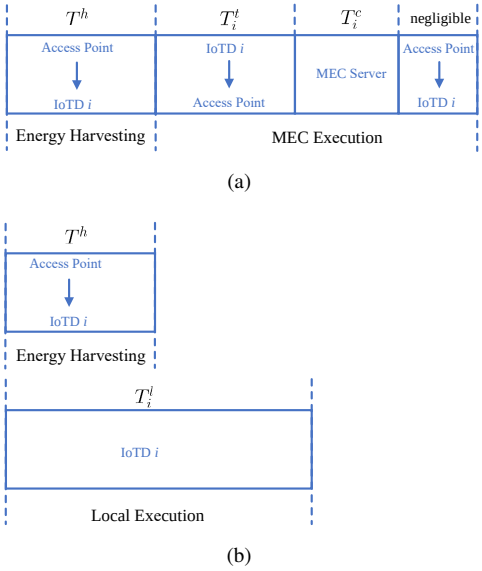

Fig. 2. Harvest-then-transmit protocol for the studied WP-MEC system. (a) MEC mode. (b) Local mode.

2) $m_i = 1$ $(i \in \mathcal{N})$, if U_i is executed on the MEC server (i.e., the MEC mode);
3) $m_i = 0$ $(i \in \mathcal{N})$, if U_i is not executed (i.e., the non-execution mode).

The execution modes of all tasks form mode set $\mathbf{m} = [m_1, m_2, \ldots, m_n]$.

Similar to [10], [22], and [47], this paper adopts the harvest-then-transmit protocol as shown in Fig. 2. Specifically, for the MEC mode shown in Fig. 2(a), the access point first broadcasts the energy to IoTDs, IoTDs then simultaneously offload their tasks to the access point for the MEC execution. However, for the local mode shown in Fig. 2(b), the local execution can be performed concurrently with the energy harvesting as IoTDs are equipped with batteries [47].

Remark 1: For ease of reference, the key notations used in this section are summarized in the supplementary file.

A. Energy Harvesting

For the energy harvesting in the MEC mode and the local mode, we assume that the access point simultaneously broadcasts energy to n IoTDs by pointing n radio beams to n IoTDs within a fixed duration of T^h [47]. We also assume that the channel gain between the access point and IoTDs follows the free-space path loss model [18]

$$h_i = 4.11 \left(\frac{3e{+}8}{4\pi f_c d_i} \right)^2, \forall i \in \mathcal{N} \tag{2}$$

where f_c denotes the carrier frequency and d_i denotes the distance between the access point and the ith IoTD.

Let $\mathbf{p}^b = [p_1^b, p_2^b, \cdots, p_n^b]$ represent the broadcast power allocated to all tasks. Then, the energy harvested by the ith IoTD is given by

$$E_i^h = \mu T^h h_i p_i^b, \forall i \in \mathcal{N} \tag{3}$$

where $\mu \in (0, 1)$ denotes the power conversion efficiency.

B. Local Execution

For the local execution in the local mode, the computing time and energy consumption of U_i are given by [5]

$$T_i^l = \frac{C_i}{r_i^l}, \forall i \in \mathcal{N} \tag{4}$$

and

$$E_i^l = k_0 (r_i^l)^3 T_i^l, \forall i \in \mathcal{N} \tag{5}$$

where r_i^l represents the computing capability of the ith IoTD and k_0 is the effective capacitance coefficient of IoTDs.

C. MEC Execution

For the MEC execution in the MEC mode, IoTDs need to transmit the input data of their tasks to the access point over the orthogonal channel based on orthogonal frequency-division multiple access, and the results of tasks are then returned to IoTDs after the MEC server completes these tasks. Due to the fact that the size of results is generally much smaller than that of the input data, we omit the time and energy consumption for receiving the results as in [7] and [13].

The transmission rate of the ith IoTD is expressed as

$$R_i = B \log_2 \left(1 + \frac{p_i^t h_i}{N_0}\right), \forall i \in \mathcal{N} \tag{6}$$

where B represents the channel bandwidth, N_0 is the variance of complex Gaussian channel noise, and p_i^t is the transmission power of the ith IoTD. Note that, p_i^t can be pre-configured according to the capability and location of the ith IoTD [48].

Then, the transmission time and energy consumption of U_i can be obtained by:

$$T_i^t = \frac{D_i}{R_i}, \forall i \in \mathcal{N} \tag{7}$$

and

$$E_i^t = \frac{p_i^t D_i}{R_i}, \forall i \in \mathcal{N}. \tag{8}$$

In addition, let $\mathbf{r}^c = [r_1^c, r_2^c, \ldots, r_n^c]$ represent the computing resources allocated to all tasks. Then, the computing time of U_i on the MEC server is expressed as

$$T_i^c = \frac{C_i}{r_i^c}, \forall i \in \mathcal{N}. \tag{9}$$

Besides, due to the fact that the energy consumption of the MEC server is generally related to the size of the input data [47], the energy consumption of U_i on the MEC server is given by

$$E_i^c = k_1 D_i, \forall i \in \mathcal{N} \tag{10}$$

where k_1 is the effective capacitance coefficient of the MEC server.

As shown in Fig. 2, the completion time of U_i is given by:

$$T_i = \begin{cases} \max\{T^h, T_i^l\}, & \text{if } m_i = -1, \\ T^h + T_i^t + T_i^c, & \text{if } m_i = 1, \\ 0, & \text{if } m_i = 0. \end{cases} \tag{11}$$

D. Profits of DO and SP

The profit of DO refers to the revenue from the reward after subtracting the cost of computing resources and energy. If U_i is successfully completed, DO can obtain a reward. In the local mode, DO only needs to purchase energy to cover the energy consumption for completing tasks locally. In the MEC mode, DO needs to purchase energy for transmitting the input data of tasks and computing resources for executing tasks on the MEC server. Note that, if a task is not executed, DO does not purchase any computing resources and energy. As a result, for U_i, the profit of DO is given by [2]

$$f_i^{do}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}, \mathbf{p}^b, \mathbf{r}^c) = \begin{cases} \alpha D_i - v_i^e p_i^b, & \text{if } m_i = -1, \\ \alpha D_i - v_i^e p_i^b - v_i^c r_i^c, & \text{if } m_i = 1, \\ 0, & \text{if } m_i = 0. \end{cases} \tag{12}$$

where αD_i is the reward and α is the reward coefficient [3]; v_i^c and v_i^e represent the unit prices of computing resources and energy for U_i, respectively; and $\mathbf{v}^c = [v_1^c, \cdots, v_n^c]$ and $\mathbf{v}^e = [v_1^e, \cdots, v_n^e]$ denote the prices of computing resources and energy for all tasks, respectively. In this paper, the discriminatory pricing scheme is studied, in which SP is able to set different unit prices of computing resources and energy for different tasks [29].

Therefore, the total profit of DO is expressed as

$$f^{do}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}, \mathbf{p}^b, \mathbf{r}^c) = \sum_{i \in \mathcal{N}} f_i^{do}. \tag{13}$$

The profit of SP refers to the revenue from selling computing resources and energy after subtracting its cost of energy. Therefore, for U_i, the profit of SP is expressed as:

$$f_i^{sp}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}, \mathbf{p}^b, \mathbf{r}^c) = \begin{cases} v_i^e p_i^b - v_0 p_i^b T^h, & \text{if } m_i = -1, \\ v_i^e p_i^b + v_i^c r_i^c - v_0 \left(p_i^b T^h + E_i^c\right), & \text{if } m_i = 1, \\ 0, & \text{if } m_i = 0. \end{cases} \tag{14}$$

where v_0 denotes the unit price of energy consumed by the access point.

Therefore, the total profit of SP is calculated as:

$$f^{sp}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}, \mathbf{p}^b, \mathbf{r}^c) = \sum_{i \in \mathcal{N}} f_i^{sp}. \tag{15}$$

E. Problem Formulation

In the studied WP-MEC system, the pricing process of computing resources and energy is the following:

1) SP first sets the prices of computing resources and energy (i.e., $\mathbf{v}^c$ and $\mathbf{v}^e$);
2) Based on device status, task information, and available resources, DO develops the optimal response, including the

[2]Since the energy harvesting time is constant, it can be omitted when calculating the profit of DO.

[3]In some scenarios, the reward may be related to D_i and/or F_i, or even a constant. The proposed algorithm in this paper can be easily extended to these scenarios.

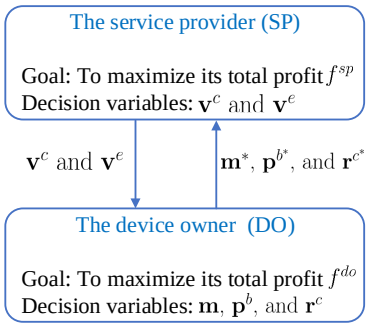

Fig. 3. Interaction between SP and DO.

optimal mode, broadcast power, and computing resource allocation (denoted as $\mathbf{m}^*$, $\mathbf{p}^{b^*}$, and $\mathbf{r}^{c^*}$), to maximize its total profit under the given $\mathbf{v}^c$ and $\mathbf{v}^e$;

3) Subsequently, SP evaluates its total profit based on $\mathbf{v}^c$, $\mathbf{v}^e$, $\mathbf{m}^*$, $\mathbf{p}^{b^*}$, and $\mathbf{r}^{c^*}$.
4) SP updates $\mathbf{v}^c$ and $\mathbf{v}^e$ and goes to Step 2 until the stopping criterion is reached;
5) Finally, the optimal $\mathbf{v}^c$ and $\mathbf{v}^e$ are output.

Based on the interaction between SP and DO in Fig. 3, a BOP is formulated, in which $\mathbf{v}^c$ and $\mathbf{v}^e$ are optimized at the upper level, and $\mathbf{m}$, $\mathbf{p}^b$, and $\mathbf{r}^c$ are optimized at the lower level under the given $\mathbf{v}^c$ and $\mathbf{v}^e$. The goals of both levels are to maximize the total profits of SP and DO, respectively. The BOP is formulated as:

$$\begin{aligned}
\mathcal{P}_0: &\max f^{sp}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}, \mathbf{p}^b, \mathbf{r}^c) \\
&\text{s.t. } \mathcal{C}1: v^c_{i,min} \le v^c_i \le v^c_{i,max},\ i \in \mathcal{N} \\
&\quad \mathcal{C}2: v^e_{i,min} \le v^e_i \le v^e_{i,max},\ i \in \mathcal{N} \\
&\quad \{\mathbf{m}, \mathbf{p}^b, \mathbf{r}^c\} = \arg\max f^{do}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}, \mathbf{p}^b, \mathbf{r}^c) \\
&\quad \text{s.t. } \mathcal{C}3: m_i \in \{-1, 1, 0\},\ i \in \mathcal{N} \\
&\qquad \mathcal{C}4: r^c_i \ge 0, \forall i \in \mathcal{N} \\
&\qquad \mathcal{C}5: \sum_{i \in \mathcal{N}} r^c_i \le r^c_{max} \\
&\qquad \mathcal{C}6: 0 \le p^b_i \le p^b_{i,max}, \forall i \in \mathcal{N} \\
&\qquad \mathcal{C}7: E^t_i \le E^h_i, \forall m_i = 1, i \in \mathcal{N} \\
&\qquad \mathcal{C}8: E^l_i \le E^h_i, \forall m_i = -1, i \in \mathcal{N} \\
&\qquad \mathcal{C}9: T_i \le T_{i,max}, \forall m_i = -1 \text{ or } 1, i \in \mathcal{N}
\end{aligned} \tag{16}$$

where $\mathcal{C}1$ specifies the lower and upper bounds of the price of computing resources; $\mathcal{C}2$ indicates the lower and upper bounds of the price of energy; $\mathcal{C}3$ states that DM needs to select a mode from the local mode, the MEC mode, and the non-execution mode for each task; $\mathcal{C}4$ represents that the computing resource allocated to each task cannot be less than 0; $\mathcal{C}5$ represents the computing resources allocated to all tasks cannot be greater than the computing capacity of the MEC server; $\mathcal{C}6$ specifies the lower and upper bounds of the broadcast power; $\mathcal{C}7$ ensures that the transmission energy consumption cannot be greater than the harvested energy if U_i is completed in the MEC mode; and $\mathcal{C}8$ ensures that the computing energy consumption cannot be greater than the harvested energy if U_i is completed in the local mode; and $\mathcal{C}9$ specifies the completion time of tasks should not be greater than the delay constraint.

It can be observed that the lower-level optimization problem of $\mathcal{P}_0$ in (16) is a mixed-variable nonlinear optimization problem since $\mathbf{m}$ is a discrete vector, and $\mathbf{p}^b$ and $\mathbf{r}^c$ are two continuous vectors. It is well known that the single-level mixed-variable nonlinear optimization problems are difficult to solve [36], not to mention BOPs involving a mixed-variable nonlinear lower-level problem. Considering the above challenge, we first transform $\mathcal{P}_0$ into a more tractable form.

F. Problem Transformation

If U_i can be completed in the local mode, $\mathcal{C}8$ should be satisfied. Since DO can obtain a higher profit by harvesting less energy according to (12) under the given v^e_i, the equality holds for $\mathcal{C}8$. Then, based on (3), we can obtain the optimal broadcast power as:

$$p^{b^*}_i = \frac{E^l_i}{\mu T^h h_i}, \forall m_i = -1, i \in \mathcal{N}. \tag{17}$$

Similarly, if U_i is completed in the MEC mode, the equality holds for $\mathcal{C}7$. Thus, the optimal broadcast power is given as:

$$p^{b^*}_i = \frac{E^t_i}{\mu T^h h_i}, \forall m_i = 1, i \in \mathcal{N}. \tag{18}$$

In addition, if U_i is completed in the MEC mode, based on $\mathcal{C}9$, the following constraint should be satisfied:

$$T^h + T^t_i + T^c_i \le T_{i,max}, \forall m_i = 1, i \in \mathcal{N}. \tag{19}$$

Re-arranging (19), we can obtain

$$r^c_i \ge \frac{C_i}{T_{i,max} - \frac{D_i}{R_i} - T^h}, \forall m_i = 1, i \in \mathcal{N}. \tag{20}$$

Under the given v^c_i, DO can obtain a higher profit by purchasing fewer computing resources according to (12). Therefore, when the equality holds for (20), the optimal computing resource allocation is obtained as:

$$r^{c^*}_i = \frac{C_i}{T_{i,max} - \frac{D_i}{R_i} - T^h}, \forall m_i = 1, i \in \mathcal{N}. \tag{21}$$

As a result, as long as $\mathbf{m}$ is known, we can obtain $\mathbf{p}^{b^*}$ and $\mathbf{r}^{c^*}$ based on (17), (18), and (21). It means that at the lower level, we only need to consider $\mathbf{m}$. Therefore, $\mathcal{P}_0$ can be reduced into the following equivalent form:

$$\begin{aligned}
\mathcal{P}_1: &\max\ f^{sp}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}) \\
&\text{s.t. } \mathcal{C}1, \mathcal{C}2 \\
&\quad \mathbf{m} = \arg\max f^{do}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}) \\
&\quad \text{s.t. } \mathcal{C}3 - \mathcal{C}6 \\
&\qquad \mathcal{C}10: T_i \le T_{i,max}, \forall m_i = -1, i \in \mathcal{N}.
\end{aligned} \tag{22}$$

Compared with $\mathcal{P}_0$, $\mathcal{P}_1$ has the following two advantages:

1) The lower-level optimization problem of $\mathcal{P}_1$ is a discrete optimization problem while that of $\mathcal{P}_0$ is a mixed-variable optimization problem. Clearly, $\mathcal{P}_1$ is easier to solve than $\mathcal{P}_0$.

2) The number of lower-level variables of $\mathcal{P}_0$ is $3n$, while that of $\mathcal{P}_1$ is n. Obviously, the search space of the lower-level optimization problem of $\mathcal{P}_1$ becomes smaller.

However, by further analyzing $\mathcal{P}_1$, we can find that the search space of $\mathcal{P}_1$ is still extremely large. For instance, for the given $\mathbf{v}^c$ and $\mathbf{v}^e$, all possible combinations of $\mathbf{m}$ are 3^n at the lower level of $\mathcal{P}_1$ due to $m_i \in \{-1, 0, 1\}$ ($\forall i \in \mathcal{N}$). It means that for each upper-level solution, the lower-level optimization has to face a huge search space. Thus, solving $\mathcal{P}_1$ directly may give rise to poor performance.

IV. DACBO

In this paper, we propose a divide-and-conquer bilevel optimization algorithm (called DACBO) to tackle $\mathcal{P}_1$, which includes two phases: task grouping and price optimization. Specifically, DACBO first groups tasks into three independent sets and then jointly optimizes the prices of computing resources and energy for each of them. In this way, a large-scale BOP is divided into three sub-problems.

A. Task Grouping

In the task grouping phase, we first introduce the conditions that need to be satisfied if the tasks can be completed in the local mode or the MEC mode. Then, the tasks are grouped into three independent sets based on these conditions.

1) Local mode: Based on $\mathcal{C}6$ and (17), if U_i can be completed in the local mode, the following constraint should be satisfied:

$$\frac{E_i^l}{\mu T^h h_i} \leq p_{i,max}^b, \forall m_i = -1, i \in \mathcal{N}. \tag{23}$$

Moreover, based on $\mathcal{C}10$ and (11), the following constraint should also be satisfied:

$$T_i^l = \frac{C_i}{r_i^l} \leq T_{i,max}, \forall m_i = -1, i \in \mathcal{N}. \tag{24}$$

Re-arranging (24), we have

$$\frac{C_i}{T_{i,max}} \leq r_i^l, \forall m_i = -1, i \in \mathcal{N}. \tag{25}$$

2) MEC mode: Based on $\mathcal{C}6$ and (18), if U_i can be completed in the MEC mode, the following constraint should be satisfied:

$$\frac{E_i^t}{\mu T^h h_i} \leq p_{i,max}^b, \forall m_i = 1, i \in \mathcal{N}. \tag{26}$$

In addition, according to $\mathcal{C}4$ and $\mathcal{C}5$, the computing resources available from the MEC server for each task are limited, that is:

$$0 \leq r_i^c \leq r_{max}^c, \forall m_i = 1, i \in \mathcal{N}. \tag{27}$$

Then, based on (21) and (27), if U_i can be completed in the MEC mode, the following constraint should also be satisfied:

$$0 \leq \frac{C_i}{T_{i,max} - \frac{D_i}{R_i} - T^h} \leq r_{max}^c, \forall m_i = 1, i \in \mathcal{N}. \tag{28}$$

Algorithm 1 Task Grouping

```
O = ∅, L = ∅, and R = ∅;
for eah task U_i (i ∈ N) do
    if (26) and (28) are satisfied then
        if (23), (25), and (29) are satisfied then
            L = L ∪ {i};
        else
            O = O ∪ {i};
        end if
    else if (23) and (25) are satisfied then
        L = L ∪ {i};
    else
        R = R ∪ {i};
    end if
end for
Output: O, L, and R.
```

It is worth noting that even if U_i can be completed in the MEC mode, DO is not of interest to this mode when U_i can be completed in the local mode and the following constraint is satisfied:

$$E_i^l \leq E_i^t, \forall i \in \mathcal{N}. \tag{29}$$

The reason is that for any given $\mathbf{v}^e$, when $E_i^l \leq E_i^t$, the energy cost of the MEC mode cannot be less than that of the local mode. In addition, in the MEC mode, DO also needs to purchase computing resources. Under this condition, the cost of the MEC mode is higher than that of the local mode. Therefore, DO is not of interest to the MEC mode.

As a result, according to device status, task information, and available resources, we can group tasks into the following three independent sets:

1) Set $\mathcal{R}$ includes tasks that cannot be completed in the local mode or the MEC mode. In this case, (23) or (25) is not satisfied, and (26) or (28) is also not satisfied.
2) Set $\mathcal{L}$ includes tasks that cannot be completed in the MEC mode but can be completed in the local mode. In this case, (26) or (28) is not satisfied, but (23) and (25) are satisfied. In addition, $\mathcal{L}$ also includes tasks for which DO is not of interest to the MEC mode although they can be completed in this mode. In this case, (23)-(29) are satisfied.
3) Set $\mathcal{O}$ includes tasks for which DO is interested in the MEC mode. In this case, (26) and (28) are satisfied, but (23), (25), and (29) cannot be satisfied simultaneously.

The above three sets $\mathcal{O}$, $\mathcal{L}$, and $\mathcal{R}$ can cover the whole tasks in $\mathcal{N}$ (i.e., $\mathcal{O} \cup \mathcal{L} \cup \mathcal{R} = \mathcal{N}$). The process of task grouping is summarized in **Algorithm** 1. Next, we explain why $\mathcal{O}$, $\mathcal{L}$, and $\mathcal{R}$ are three independent sets:

1) For each task in $\mathcal{R}$, its execution mode is unique and is not affected by the execution modes of other tasks in $\mathcal{N}$. In this case, the prices of computing resources and energy for this task can be optimized independently.
2) For each task in $\mathcal{L}$, whether it can be completed locally or not is determined by its device status, task information, as well as the price of energy set by SP, which is not related to the execution modes of other tasks in $\mathcal{N}$. In this case, the prices of computing resources and energy for this task can also be optimized independently.

3) For each task in $\mathcal{O}$, not all offloading requests may be accepted due to the computing limitation of the MEC server (i.e., $\mathcal{C}5$). Thus, its execution mode is affected by the execution modes of other tasks in $\mathcal{O}$. As a result, the prices of computing resources and energy for all tasks in $\mathcal{O}$ need to be optimized simultaneously.

Therefore, the prices of computing resources and energy for $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{O}$ can be optimized separately.

B. Price Optimization of Computing Resources and Energy for $\mathcal{R}$

Since U_i ($\forall i \in \mathcal{R}$) cannot be completed both in the local mode and in the MEC mode, the optimal mode is $m_i^* = 0$. In this case, DO does not need to purchase any computing resource or energy from SP for U_i. As a result, SP can set the optimal prices of computing resources and energy (denoted as $v_i^{c^*}$ and $v_i^{e^*}$) to any value within $[v_{i,min}^c, v_{i,max}^c]$ and $[v_{i,min}^e, v_{i,max}^e]$, respectively.

C. Price Optimization of Computing Resources and Energy for $\mathcal{L}$

In order to maximize f_i^{do}, DO always selects the more profitable one between the local mode and the non-execution mode for U_i ($\forall i \in \mathcal{L}$). Based on (12), we have

$$f_i^{do} = \begin{cases} \alpha D_i - v_i^e p_i^b, & \text{if } m_i = -1, \\ 0, & \text{if } m_i = 0. \end{cases} \tag{30}$$

Therefore, under the given v_i^e, if $\alpha D_i - v_i^e p_i^b \geq 0$ (i.e., $v_i^e \leq \frac{\alpha D_i}{p_i^b}$), then DO selects the local mode for U_i; otherwise, DO selects the non-execution mode for U_i. As a result, we have

$$m_i^* = \begin{cases} -1, & \text{if } v_i^e \leq \frac{\alpha D_i}{p_i^b}, \\ 0, & \text{if } v_i^e > \frac{\alpha D_i}{p_i^b}. \end{cases} \tag{31}$$

As for SP, since U_i does not require any computing resource of the MEC server, $v_i^{c^*}$ can be set to any value within $[v_{i,min}^c, v_{i,max}^c]$. In addition, regarding $v_i^{e^*}$, we have

$$v_i^{e^*} = \begin{cases} v_{i,max}^e, & \text{if } \frac{\alpha D_i}{p_i^b} \geq v_{i,max}^e, \\ v_{i,r_1}, & \text{if } \frac{\alpha D_i}{p_i^b} < v_{i,min}^e, \\ \frac{\alpha D_i}{p_i^b}, & \text{if } \max\{v_{i,min}^e, v_o T^h\} \leq \frac{\alpha D_i}{p_i^b} < v_{i,max}^e, \\ v_{i,r_2}, & \text{if } v_{i,min}^e \leq \frac{\alpha D_i}{p_i^b} < \min\{v_{i,max}^e, v_o T^h\} \end{cases} \tag{32}$$

where v_{i,r_1} represents a value randomly selected from $[v_{i,min}^e, v_{i,max}^e]$ and v_{i,r_2} represents a value randomly selected from $(\frac{\alpha D_i}{p_i^b}, v_{i,max}^e)$. (32) can be derived through the following process:

1) When $\frac{\alpha D_i}{p_i^b} \geq v_{i,max}^e$, $v_i^e \leq v_{i,max}^e \leq \frac{\alpha D_i}{p_i^b}$ always holds. Therefore, based on (31), $m_i^* = -1$. In this case, based on (14), $f_i^{sp} = v_i^e p_i^b - v_0 p_i^b T^h$. It can be found that f_i^{sp} is proportional to v_i^e; thus, in order to maximize f_i^{sp}, $v_i^{e^*} = v_{i,max}^e$;
2) When $\frac{\alpha D_i}{p_i^b} < v_{i,min}^e$, $v_i^e \geq v_{i,min}^e > \frac{\alpha D_i}{p_i^b}$ always holds. Thus, based on (31), $m_i^* = 0$. In this case, based on (14), $f_i^{sp} = 0$ and $v_i^{e^*}$ can be set to any value within $[v_{i,min}^e, v_{i,max}^e]$;
3) When $v_{i,min}^e \leq \frac{\alpha D_i}{p_i^b} < v_{i,max}^e$, we need to consider the following two cases:
   i) If $v_i^e \in [v_{i,min}^e, \frac{\alpha D_i}{p_i^b}]$, then $\alpha D_i - v_i^e p_i^b \geq 0$ always holds. Thus, based on (31), $m_i^* = -1$. In this case, based on (14), $f_i^{sp} = v_i^e p_i^b - v_0 p_i^b T^h$ (i.e., red lines in Fig. 4). Moreover, it can be found that f_i^{sp} is proportional to v_i^e. Therefore, in order to maximize f_i^{sp}, $v_i^{e^*} = \frac{\alpha D_i}{p_i^b}$ and $f_i^{sp}(v_i^{e^*}) = \alpha D_i - v_0 p_i^b T^h$.
   ii) If $v_i^e \in (\frac{\alpha D_i}{p_i^b}, v_{i,max}^e)$, then $\alpha D_i - v_i^e p_i^b < 0$ always holds. Thus, based on (31), $m_i^* = 0$. In this case, based on (14), f_i^{sp} is always equal to 0 (i.e., green lines in Fig. 4).

   Then, based on cases i) and ii), we can obtain $v_i^{e^*}$ over $[v_{i,min}^e, v_{i,max}^e]$. If $f_i^{sp}(\frac{\alpha D_i}{p_i^b}) = \alpha D_i - v_0 p_i^b T^h \geq 0$ and $v_{i,min}^e \leq \frac{\alpha D_i}{p_i^b} < v_{i,max}^e$ (i.e., $\max\{v_{i,min}^e, v_o T^h\} \leq \frac{\alpha D_i}{p_i^b} < v_{i,max}^e$), as shown in Fig. 4(a), it is clear that $v_i^{e^*} = \frac{\alpha D_i}{p_i^b}$. In addition, if $v_{i,min}^e \leq \frac{\alpha D_i}{p_i^b} < \min\{v_{i,max}^e, v_o T^h\}$, as shown in Fig. 4(b), $v_i^{e^*}$ can be set to any value within $(\frac{\alpha D_i}{p_i^b}, v_{i,max}^e)$.

D. Price Optimization of Computing Resources and Energy for $\mathcal{O}$

The price optimization of computing resources and energy for $\mathcal{O}$ can be reformulated as:

$$\begin{aligned} \mathcal{P}_2: \max\ & f^{sp}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}) \\ \text{s.t. } & \mathcal{C}1': v_{i,min}^e \leq v_i^e \leq v_{i,max}^e,\ i \in \mathcal{O} \\ & \mathcal{C}2': v_{i,min}^c \leq v_i^c \leq v_{i,max}^c,\ i \in \mathcal{O} \\ & \mathbf{m} = \arg\max\ f^{do}(\mathbf{v}^c, \mathbf{v}^e, \mathbf{m}) \\ & \text{s.t. } \mathcal{C}3': m_i \in \{-1, 1, 0\},\ i \in \mathcal{O} \\ & \quad \mathcal{C}4': r_i^c \geq 0, \forall i \in \mathcal{O} \\ & \quad \mathcal{C}5': \sum_{i \in \mathcal{O}} r_i^c \leq r_{max}^c \\ & \quad \mathcal{C}6': 0 \leq p_i^b \leq p_{i,max}^b, \forall i \in \mathcal{O} \\ & \quad \mathcal{C}10': T_i \leq T_{i,max}, \forall m_i = -1, i \in \mathcal{O}. \end{aligned} \tag{33}$$

Since $\mathcal{P}_2$ is a mixed-variable nonlinear bilevel optimization problem, neither traditional bilevel optimization algorithms nor single-level reduction-based bilevel optimization algorithms can be used. Therefore, as mentioned in Section II-B, we need to adopt nested-based bilevel optimization algorithms, which solve the upper-level and lower-level optimization problems in a nested manner. However, the lower-level optimization problem of $\mathcal{P}_2$ is NP-hard [48]. Although deterministic methods, such as branching, may find the global optimum of the lower-level optimization problem of $\mathcal{P}_2$, they may suffer from a high computational burden. Compared with deterministic methods, heuristic methods requires fewer computational burden. Note that, since the lower-level optimization needs to be performed for each upper-level solution in nested-based approaches, the computational cost of population-based heuristic methods (e.g., evolutionary algorithms) is still unbearable. This is

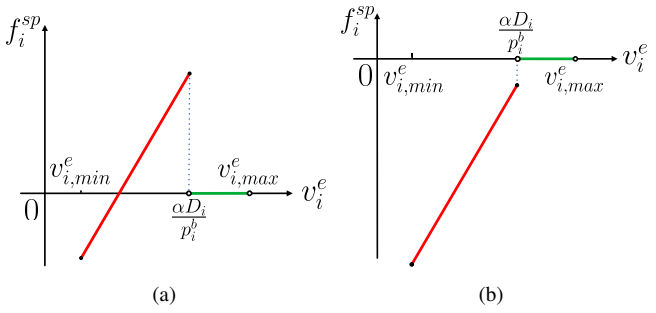

Fig. 4. Illustration of f_i^{sp} when $v^e_{i,\min} \leq \frac{\alpha D_i}{p_i^b} < v^e_{i,\max}$. (a) $\max\{v^e_{i,min}, v_o T^h\} \leq \frac{\alpha D_i}{p_i^b} < v^e_{i,max}$, (b) $v^e_{i,min} \leq \frac{\alpha D_i}{p_i^b} < \min\{v^e_{i,max}, v_o T^h\}$

because population-based heuristic methods search for the optimal solution by using multiple individuals at the same time. Therefore, a single-point based heuristic method, called variable neighborhood method (VNS), is developed to solve the lower-level optimization problem of $\mathcal{P}_2$.

As presented in **Algorithm** 2, VNS consists of two main parts: initial solution construction and task exchange.

1) Initial Solution Construction: In this part, we employ the greedy method to quickly obtain a good initial solution. The detailed process is give as follows. Under the given v_i^c and v_i^e $(i \in \mathcal{O})$, we first calculate the profit improvement of DO from the MEC mode against the more profitable one between the local mode and the non-execution mode [4]:

$$\Delta f_i^{do} = f_i^{do}|_{m_i=1} - \max\{f_i^{do}|_{m_i=-1}, f_i^{do}|_{m_i=0}\}, \forall i \in \mathcal{O} \tag{34}$$

where $f_i^{do}|_{m_i=1}$, $f_i^{do}|_{m_i=-1}$, and $f_i^{do}|_{m_i=0}$ represent the profits of DO from the MEC mode, the local mode, and the non-execution mode, respectively (line 2).

Note that, only if $\Delta f_i^{do} > 0$, DO can obtain a positive profit improvement from the MEC mode. In contrast, if $\Delta f_i^{do} \leq 0$, the task has no incentive to send an offloading request to the access point. Therefore, we define set $\mathcal{S} = \{i|\Delta f_i^{do} > 0, \forall i \in \mathcal{O}\}$ to record the tasks that send the offloading requests (line 3). Due to the computing limitation of the MEC server, not all offloading requests of the tasks can be accepted. Therefore, some tasks are selected from $\mathcal{S}$ to be completed in the MEC mode. In order to maximize the profit of DO under the computing limitation of the MEC server, we prefer to the tasks with higher profit improvement and lower computing resource requirement. To this end, for each task in $\mathcal{S}$, the normalized profit improvement of DO is calculated:

$$\Delta f_{i,norm}^{do} = \frac{\Delta f_i^{do}}{r_i^c}, \forall i \in \mathcal{S}. \tag{35}$$

The tasks in $\mathcal{S}$ are then sorted in descending order of $\Delta f_{i,norm}^{do}$ (lines 4-5). After that, we check the tasks in $\mathcal{S}$ one by one. If its computing resource requirement can be satisfied, the task is added to $\mathcal{S}_1$ and the amount of unused computing resources of the MEC server is updated by subtracting the computing resources allocated to this task; otherwise, the task is added to $\mathcal{S}_2$. After checking all tasks in $\mathcal{S}$, we further add tasks that belong to $\mathcal{O}$ but not belong to $\mathcal{S}$ into $\mathcal{S}_2$. Then, $\mathcal{S}_1$ and $\mathcal{S}_2$ are used to construct initial mode set $\mathbf{m}$. To be specific, DO selects the MEC mode for each task in $\mathcal{S}_1$, and selects the more profitable one between the local mode and the non-execution mode for each task in $\mathcal{S}_2$. Subsequently, the performance of $\mathbf{m}$ is evaluated at the lower level of $\mathcal{P}_2$ (lines 6-17).

[4]Some tasks in $\mathcal{O}$ may not be completed in the local mode. For these tasks, we consider that the profit of DO from the local mode is lower than that from the non-execution mode.

Algorithm 2 VNS

1: **/*Initial solution construction*/**
2: Calculate the profit improvement of DO for each task in $\mathcal{O}$: Δf_i^{do};
3: $\mathcal{S} = \{i|\Delta f_i^{do} > 0, i \in \mathcal{O}\}$;
4: Calculate the normalized profit improvement of DO for each task in $\mathcal{S}$: $\Delta f_{i,norm}^{do}$;
5: Sort the tasks in $\mathcal{S}$ in descending order of $\Delta f_{i,norm}^{do}$;
6: $\mathcal{S}_1 \leftarrow \emptyset$;
7: $\mathcal{S}_2 \leftarrow \emptyset$;
8: $r_{unused}^c = r_{max}^c$;
9: **for** each task U_i $(i \in \mathcal{S})$ **do**
10: **if** the computing resource requirement of U_i is satisfied **then**
11: Add U_i into $\mathcal{S}_1$ and update r_{unused}^c by subtracting the computing resources allocated to U_i;
12: **else**
13: Add U_i into $\mathcal{S}_2$;
14: **end if**
15: **end for**
16: $\mathcal{S}_2 \leftarrow \mathcal{S}_2 \cup \mathcal{O} \backslash \mathcal{S}$;
17: Construct $\mathbf{m}$ based on $\mathcal{S}_1$ and $\mathcal{S}_2$ and evaluate the performance of $\mathbf{m}$ at the lower level of $\mathcal{P}_2$;
18: **/*Task exchange*/**
19: **for** each task U_i $(i \in \mathcal{S}_2)$ **do**
20: **for** each task U_j $(j \in \mathcal{S}_1)$ **do**
21: **if** $\Delta f_i^{do} > \Delta f_j^{do}$ **then**
22: Swap U_i and U_j to form two new sets $\mathcal{S}_1'$ and $\mathcal{S}_2'$;
23: Construct $\mathbf{m}'$ based on $\mathcal{S}_1'$ and $\mathcal{S}_2'$ and evaluate the performance of $\mathbf{m}'$ at the lower level of $\mathcal{P}_2$;
24: **if** $\mathbf{m}'$ satisfies all constriants at the lower level of $\mathcal{P}_2$ and the profit of DO from $\mathbf{m}'$ is bigger than that from $\mathbf{m}$ **then**
25: $\mathcal{S}_1 \leftarrow \mathcal{S}_1'$, $\mathcal{S}_2 \leftarrow \mathcal{S}_2'$, and $\mathbf{m} \leftarrow \mathbf{m}'$;
26: **end if**
27: **end if**
28: **end for**
29: **end for**
30: **Output:** $\mathbf{m}$.

2) Task Exchange: In order to find a better solution, we exchange the tasks in $\mathcal{S}_1$ and $\mathcal{S}_2$. For U_i $(i \in \mathcal{S}_2)$ and U_j $(j \in \mathcal{S}_1)$, if $\Delta f_i^{do} > \Delta f_j^{do}$, we exchange U_i and U_j to form two new sets $\mathcal{S}_1'$ and $\mathcal{S}_2'$. Subsequently, $\mathcal{S}_1'$ and $\mathcal{S}_2'$ are used to construct new mode set $\mathbf{m}'$ and its performance is evaluated at the lower level of $\mathcal{P}_2$. If $\mathbf{m}'$ satisfies all constraints at the lower level of $\mathcal{P}_2$ and the profit of DO from $\mathbf{m}'$ is bigger than than that from $\mathbf{m}$, we update $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathbf{m}$ by using $\mathcal{S}_1'$, $\mathcal{S}_2'$, and $\mathbf{m}'$, respectively (lines 18-29). The above process is repeated until all tasks in $\mathcal{S}_1$ and $\mathcal{S}_2$ are checked.

Due to the discrete variables at the lower level, it is likely that the landscape for the upper level optimization problem of $\mathcal{P}_2$ is disconnected [36]. In this case, traditional methods may result in poor performance. To this end, DE, a simple and effective evolutionary algorithm proposed by Storn and Price [49], is adopted to address the upper-level optimization problem of $\mathcal{P}_2$. The overall process of the nested bilevel optimization algorithm (called DE-VNS) is presented in **Algorithm** 3. In the initialization, population $\mathbf{P}^0 = \{\mathbf{x}_1^0, \mathbf{x}_2^0, \cdots, \mathbf{x}_{NP}^0\}$ is

Algorithm 3 DE-VNS

1: $t = 0$; // t denotes the generation number
2: Randomly generate population $\mathbf{P}^0 = \{\mathbf{x}_1^0, \mathbf{x}_2^0, \cdots, \mathbf{x}_{NP}^0\}$, in which each individual represents the prices of computing resources and energy for the tasks in $\mathcal{O}$ and NP denotes the population size;
3: **for** i=1:NP **do**
4: Perform **Algorithm** 2 to obtain the corresponding optimal mode set for $\mathbf{x}_i^0$;
5: Evaluate the performance of $\mathbf{x}_i^0$ at the upper level of $\mathcal{P}_2$;
6: **end for**
7: $FEs^u = NP$;
8: **while** $FEs^u < MaxFEs^u$ **do**
9: Implement the mutation and crossover operators of DE on $\mathbf{P}^t$ to generate offspring population $\mathbf{Q}^t = \{\mathbf{q}_1^t, \mathbf{q}_2^t, \cdots, \mathbf{q}_{NP}^t\}$;
10: $\mathbf{P}^{t+1} \leftarrow \emptyset$;
11: **for** i=1:NP **do**
12: Perform **Algorithm** 2 to obtain the corresponding optimal mode set for $\mathbf{q}_i^t$;
13: Evaluate the performance of $\mathbf{q}_i^t$ at the upper level of $\mathcal{P}_2$;
14: Perform the select operator of DE to select the better one between $\mathbf{x}_i^t$ and $\mathbf{q}_i^t$, denoted as $\mathbf{x}_i^{t+1}$;
15: $\mathbf{P}^{t+1} \leftarrow \mathbf{P}^{t+1} \cup \mathbf{x}_i^{t+1}$;
16: **end for**
17: $FEs^u = FEs^u + NP$;
18: $t = t + 1$;
19: **end while**
20: **Output:** the best individual in $\mathbf{P}^{t+1}$.

first randomly initialized, in which each individual represents the prices of computing resources and energy for all tasks in $\mathcal{O}$ and NP denotes the population size. Then, VNS in **Algorithm** 2 is adopted to obtain the corresponding optimal mode set for each individual. After that, the performance of each individual is evaluated at the upper level of $\mathcal{P}_2$. During the evolution, offspring population $\mathbf{Q}^t = \{\mathbf{q}_1^t, \mathbf{q}_2^t, \cdots, \mathbf{q}_{NP}^t\}$ is generated from $\mathbf{P}^t = \{\mathbf{x}_1^t, \mathbf{x}_2^t, \cdots, \mathbf{x}_{NP}^t\}$ via the mutation and crossover operators of DE, where t denotes the generation number. Subsequently, the corresponding optimal mode set for each individual in $\mathbf{Q}^t$ is obtained via VNS and the performance of each individual in $\mathbf{Q}^t$ is evaluated at the upper level of $\mathcal{P}_2$. Finally, the selection operator is performed to select the better one between $\mathbf{x}_i^t$ and $\mathbf{q}_i^t$ ($i = 1, \ldots, NP$) into the next population $\mathbf{P}^{t+1}$. The above procedure repeats until the stopping criterion is met, i.e., the maximal number of the upper-level fitness evaluations (FEs) (denoted as $MaxFEs^u$) is reached. Afterward, the best individual in $\mathbf{P}^{t+1}$ is output.

E. Discussion

As shown in Fig. 5, we summarize the challenges and our techniques used in this paper. In order to solve challenge 1, we simplify $\mathcal{P}_0$ into $\mathcal{P}_1$. Subsequently, the task grouping is performed to tackle challenge 2. In addition, a nested bilevel optimization algorithm combining DE with VNS is proposed to solve challenges 3-5 derived from the price optimization of computing resources and energy for $\mathcal{O}$. Moreover, we develop the analytical methods for the price optimization of computing resources and energy for $\mathcal{L}$ and $\mathcal{R}$.

V. Experimental Studies

A. Competitors

Existing studies rarely design bilevel optimization methods for BOPs involving a continuous upper-level optimization problem and a discrete lower-level optimization problem. Therefore, it is difficult to directly use existing methods as competitors. For this reason, we fine-tuned two existing methods as competitors:

- Bilevel DE (BIDE) [44]: It uses the continuous version of DE at the upper level, while the discrete version of DE is adopted at the lower level. The main difference between the discrete and continuous versions of DE is that the discrete version of DE needs to implement the rounding operator after the crossover operator. In the rounding operator, if the value of a variable (ranging from 0 to 1) is not less than 0.5, DO will select the MEC mode for the corresponding task. It is worth noting that the mutation and crossover operators used in BIDE and DACBO are the same, i.e., "DE/rand/1" mutation operator and the binomial crossover operator [50]. In addition, a recently proposed strategy [51] is incorporated into DE at each level. This strategy stores the successful difference vectors and reuses them in subsequent generations, which can improve the search capability of DE.
- Bilevel genetic algorithm (BIGA) [45]: It adopts genetic algorithm (GA) as the search engine at each level. Note that, the simulated binary crossover and the polynomial mutation are employed at the upper level for the continuous optimization problem and the singe-point crossover and the bit-wise mutation are used at the lower level for the discrete optimization problem.

TABLE I
Parameter Settings of the studied WP-MEC system

Parameter	Value	Parameter	Value
$D_i, i \in \mathcal{N}$	[0.1,100] KB	$C_i, i \in \mathcal{N}$	[1,1000] MCycles
$T_{i,max}, i \in \mathcal{N}$	2 s	f_c	915 MHz
μ	0.8	T^h	0.5 s
$r_i^l, i \in \mathcal{N}$	0.5 GCycles/s	k_0	1e-28
B	1 MHz	N_0	1e-10 W
$p_i^t, i \in \mathcal{N}$	0.1 W	k_1	1e-10
v_0	0.1 per J	α	5e-4
$v_{i,min}^e, i \in \mathcal{N}$	1 per W	$v_{i,max}^e, i \in \mathcal{N}$	20 per W
$v_{i,min}^c, i \in \mathcal{N}$	1 per GCycles	$v_{i,max}^c, i \in \mathcal{N}$	20 per GCycles
r_{max}^c	10 GCycles	$p_{i,max}^b, i \in \mathcal{N}$	5 W

B. Parameter Setting

Table I summarizes the parameter settings of the studied WP-MEC system [18], [47]. Besides, ten instances with different numbers of IoTDs were used to evaluate the performance of DACBO, i.e., $n = 20, 40, \cdots, 200$. It is assumed that all IoTDs were randomly distributed in a circular area with a radius of 5 m and the access point was located in the center of the area.

The parameter settings of DACBO and the two competitors were given as follows. For DACBO and BIDE, the crossover control parameter and the scaling factor of DE were set to 0.9 and 0.9, respectively. For BIGA, the crossover probability of the simulated binary crossover, the mutation probability of the polynomial mutation, and the mutation probability of the bitwise mutation were set to 1.0, and $1/D$, and $1/D$, respectively, where D represents the length of individuals. In addition, the distributed indexes of the simulated binary crossover and the polynomial mutation were set to 20 and 20,

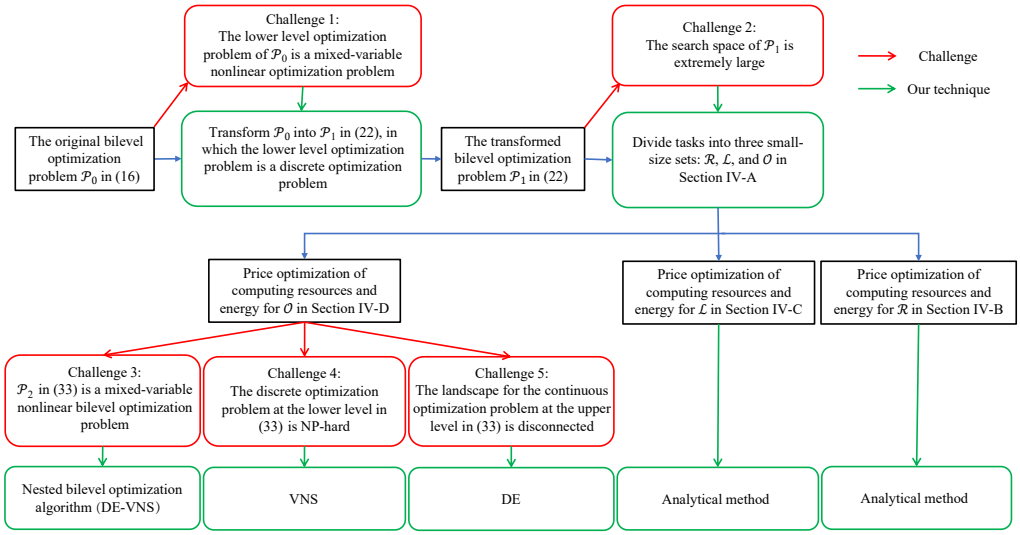

Fig. 5. Challenges and our techniques used in this paper.

respectively. NP was set to 30 at the upper level of DACBO, both levels of BIDE, and both levels of BIGA. The upper-level optimization of each algorithm was terminated at $30,000$ FEs (i.e., $MaxFEs^u = 30,000$). Additionally, in BIGA and BIDE, for a given upper-level solution, the maximum number of FEs for the lower-level optimization (denoted as $MaxFEs^l$) was set to $3,000$. Each algorithm was independently run 30 times on each instance. We implemented all the experiments in MATLAB and tested them on a personal computer with Intel Core i7-7500 CPU @2.70 GHz and 8 GB of RAM.

C. Performance Metric

In this paper, the average total profit of SP over 30 runs was selected as the performance metric to evaluate the performance of the three compared algorithms. However, for solution $\{\mathbf{v}^e, \mathbf{v}^c, \mathbf{m}\}$ obtained by an algorithm in one run, if $\mathbf{m}$ is not optimal corresponding to $\{\mathbf{v}^e, \mathbf{v}^c\}$, it may give rise to inaccurate evaluation about the average total profit of SP. Therefore, the following steps were implemented. For the given $\{\mathbf{v}^e, \mathbf{v}^c\}$, the analytical methods introduced in Sections IV-B and IV-C were used to obtain m_i^* ($i \in \mathcal{R} \cup \mathcal{L}$). As for $\mathcal{O}$, we run the lower-level optimization 30 times independently for $\{\mathbf{v}^e, \mathbf{v}^c\}$ and recorded the best mode m_i^* ($i \in \mathcal{O}$) [5]. Note that, in order to obtain m_i^* ($i \in \mathcal{O}$) that is as close as possible to the true optimal mode, we adopted GA used at the lower level of BIGA as the optimizer and set the population size and the maximum number of FEs for the lower-level optimization to $5 * NP$ and $10 * MaxFEs^l$, respectively.

D. Results and Discussions

In this section, we employed BIDE and BIGA to directly solve $\mathcal{P}_1$ and compared the performance of BIDE, BIGA, and DACBO. In Table II, we presented their results, where

[5]The above process was executed for performance evaluation after the algorithms had terminated, rather than during the running of the algorithms; thus, the running time of the above process was not counted in the running time of the algorithms.

TABLE II
RESULTS OF DACBO AND THE TWO COMPETITORS IN TERMS OF THE AVERAGE TOTAL PROFIT OF SP.

n	BIDE Mean (Std Dev)	BIGA Mean (Std Dev)	DACBO Mean (Std Dev)
20	3.4148e+2 (4.6544e+0) ↑ [1.35%]	3.3935e+2 (8.9212e+0) ↑ [1.99%]	3.4609e+2 (2.1235e+0)
40	4.8710e+2 (2.1031e+1) ↑ [10.37%]	4.7009e+2 (2.2985e+1) ↑ [14.37%]	5.3763e+2 (4.6501e+0)
60	5.7884e+2 (1.5758e+1) ↑ [22.51%]	5.6675e+2 (2.8098e+1) ↑ [25.12%]	7.0914e+2 (8.7725e+0)
80	6.0513e+2 (2.6902e+1) ↑ [26.84%]	5.5650e+2 (3.6681e+1) ↑ [37.92%]	7.6755e+2 (1.3840e+1)
100	6.8413e+2 (2.2255e+1) ↑ [27.94%]	6.0464e+2 (3.4952e+1) ↑ [44.76%]	8.7530e+2 (1.5404e+1)
120	6.9712e+2 (3.1566e+1) ↑ [34.71%]	6.2828e+2 (4.0854e+1) ↑ [49.48%]	9.3915e+2 (1.5264e+1)
140	7.7855e+2 (2.9308e+1) ↑ [36.43%]	7.1280e+2 (4.1079e+1) ↑ [49.02%]	1.0622e+3 (1.7471e+1)
160	7.9595e+2 (3.0746e+1) ↑ [45.54%]	7.0768e+2 (4.0829e+1) ↑ [63.71%]	1.1585e+3 (2.1564e+1)
180	8.5760e+2 (3.6570e+1) ↑ [50.05%]	7.6804e+2 (5.1921e+1) ↑ [67.55%]	1.2869e+3 (2.6290e+1)
200	8.7877e+2 (4.1630e+1) ↑ [55.00%]	8.1902e+2 (4.9973e+1) ↑ [66.31%]	1.3621e+3 (2.4448e+1)
↑ / ↓ / ≈	10/0/0	10/0/0	

"Mean" and "Std Dev" represent the average and standard deviation of the total profits of SP over 30 runs. In addition, percentages in the square brackets indicate the performance improvement of DACBO against each competitor. To test the statistical significance between DACBO and each competitor, the Wilcoxon's rank-sum test at a 0.05 significance level was performed. In Table II, "↑", "↓" and "≈" indicate that the performance of DACBO is better than, worse than, and similar to the competitor, respectively.

As shown in Table II, DACBO provides a better average total profit of SP than BIDE and BIGA on each instance. In terms of the performance improvement, DACBO shows obvious advantages on most instances compared with BIDE and BIGA. Specifically, the performance improvement of DACBO against BIDE can be greater than 20% and 30% when $60 \leq n \leq 100$ and $120 \leq n \leq 160$, respectively. When $n \geq 180$, DACBO achieves more than 50% performance improvement against BIDE. Compared with BIGA, the performance improvement of DACBO can achieve 14.37%,

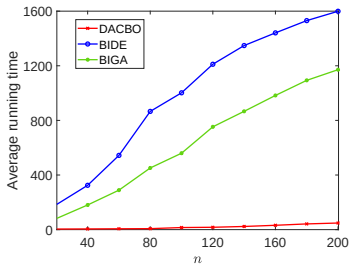

Fig. 6. Average running time of DACBO, BIDE, and BIGA.

TABLE III
RESULTS OF DACBO AND THE TWO COMPETITORS WITH THE DIVIDE-AND-CONQUER STRATEGY IN TERMS OF THE AVERAGE TOTAL PROFIT OF SP.

n	BIDE Mean (Std Dev)	BIGA Mean (Std Dev)	DACBO Mean (Std Dev)
20	3.4536e+2 (2.4383e+0) ≈ [0.21%]	3.4523e+2 (4.4660e+0) ↑ [0.25%]	3.4609e+2 (2.1235e+0)
40	5.3309e+2 (5.1402e+0) ↑ [0.85%]	5.2930e+2 (1.1924e+1) ↑ [1.57%]	5.3763e+2 (4.6501e+0)
60	7.0161e+2 (1.3835e+1) ↑ [1.07%]	6.9515e+2 (2.9238e+1) ↑ [2.01%]	7.0914e+2 (8.7725e+0)
80	7.2680e+2 (1.7214e+1) ↑ [6.84%]	6.8324e+2 (2.8833e+1) ↑ [12.34%]	7.6755e+2 (1.3840e+1)
100	7.7458e+2 (2.2761e+1) ↑ [7.29%]	7.1238e+2 (3.1876e+1) ↑ [22.87%]	8.7530e+2 (1.5404e+1)
120	8.0557e+2 (3.1519e+1) ↑ [16.58%]	7.4345e+2 (2.9547e+1) ↑ [26.32%]	9.3915e+2 (1.5264e+1)
140	8.9081e+2 (3.6438e+1) ↑ [19.23%]	8.3150e+2 (4.2365e+1) ↑ [27.74%]	1.0622e+3 (1.7471e+1)
160	8.9802e+2 (3.2629e+1) ↑ [29.01%]	8.4618e+2 (5.0934e+1) ↑ [36.91%]	1.1585e+3 (2.1564e+1)
180	9.2706e+2 (3.6570e+1) ↑ [38.81%]	9.1600e+2 (5.0754e+1) ↑ [40.49%]	1.2869e+3 (2.6290e+1)
200	9.7769e+2 (4.1630e+1) ↑ [39.31%]	9.5126e+2 (6.8667e+1) ↑ [43.19%]	1.3621e+3 (2.4448e+1)
↑ / ↓ / ≈	9/0/1	10/0/0	

25.12%, and 37.92% when $n = 40$, 60, and 80, respectively. In addition, when $100 \leq n < 160$ and $n \geq 160$, DACBO can improve more than 40% and 60% average total profit of SP, respectively. Moreover, DACBO performs statistically better than BIDE and BIGA on each instance. Fig. 6 summaries the average running time of BIDE, BIGA, and DACBO in one run on each instance. It is clear that DACBO is much faster than BIDE and BIGA. For instance, when $n = 200$, the average running time of BIDE, BIGA, and DACBO in one run is 1599.17 s, 1171.29 s, and 48.39 s, respectively.

The performance superiority of DACBO can be explained as follows. For both BIDE and BIGA, the dimensions of the search space (i.e., the numbers of variables) at the upper and lower levels are $2n$ and n, respectively. As stated in [46], BOPs with more than 25 variables at each level can be regarded as large-scale BOPs (note that we tested $20 \leq n \leq 200$ in this paper). Obviously, BIDE and BIGA face large-scale BOPs, thus resulting in poor performance. Instead, DACBO uses the divide-and-conquer strategy, which divides $\mathcal{P}_1$ into three sub-problems and solves them separately. Compared with $\mathcal{P}_1$, the smaller-scale optimization problems are more tractable; thus, DACBO achieves excellent performance.

One may be interested in the performance difference among BIDE, BIGA, and DACBO if the divide-and-conquer strategy is incorporated to BIDE and BIGA. Specifically, we only used BIDE and BIGA to address the price optimization of computing resources and energy for $\mathcal{O}$, and employed the analytical methods used in DACBO to address the joint price optimization of computing resources and energy for $\mathcal{R}$ and $\mathcal{L}$. Under this condition, the main difference among BIDE, BIGA, and DACBO is that the prices of computing resources and energy for $\mathcal{O}$ are optimized by BIDE, BIGA, and DE-VNS, respectively. Table III shows the results of BIDE, BIGA, and DACBO. We can observe that in terms of the average total profit of SP, DACBO is better than BIDE and BIGA on each instance. In addition, as the number of IoTDs increases, DACBO achieves increasing performance improvement compared with BIDE and BIGA. Specifically, when $n = 20$, DACBO provides 0.21% and 0.25% performance improvement against BIDE and BIGA, respectively. However, when $n = 200$, the performance improvement of DACBO is 39.31% and 43.19% against BIDE and BIGA, respectively. According to the Wilcoxon's rank-sum test at a 0.05 significance level, DACBO outperforms BIDE and BIGA on nine and ten instances, respectively, and performs similarly with BIDE on one instance.

The reasons why DACBO performs better than BIDE and BIGA can be explained in the following two aspects.

- Fig. 7 presents the average numbers of lower-level FEs required by DACBO, BIDE, and BIGA. It is clear that BIDE and BIGA need more FEs than DACBO. It is because population-based heuristic methods are adopted at the lower levels of BIDE and BIGA, while the single-point based heuristic method (i.e., VNS) is employed at the lower level of DACBO. Therefore, when only limited FEs can be provided for the lower-level optimization, the lower-level solution found by BIDE and BIGA may be far from the true optimum and give rise to inaccurate performance evaluation at the upper level.
- The studied BOP has a strong conflict between both levels. Specifically, compared with the true lower-level optimum, an inaccurate lower-level optimum may result in better upper-level performance [52]. Note, however, that, BIDE and BIGA may obtain different lower-level optimum for the same upper-level solution in different runs. As a result, some poor lower-level optimum may be retained due to the good performance at the upper level. In contrast, DACBO can obtain a stable lower-level optimum for the same upper-level solution in different runs; thus, the above issue can be alleviated.

Additionally, it is worth noting that the performance of BIDE and BIGA in Table III has an edge over that in Table II, which verifies the effectiveness of the divide-and-conquer strategy.

Remark 2: In the supplementary file, we also investigated the effectiveness of task grouping, the effectiveness of the greedy method, and the performance of different pricing schemes.

VI. CONCLUSION

In this paper, we studied a WP-MEC system. In this system, the price optimization of computing resources and energy was formulated as a BOP, in which the prices of computing resources and energy were optimized at the upper level for

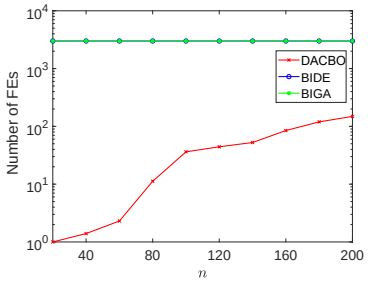

Fig. 7. Average number of lower-level FEs required by DACBO, BIDE, and DIGA.

SP, and the mode selection, broadcast power, and computing resource allocation were optimized at the lower level for DO. This BOP was then transformed into a tractable form, where only the mode selection was considered at the lower level by taking advantage of the relationships between the optimal broadcast power and the mode selection and between the optimal computing resource allocation and the mode selection. To solve the transformed BOP effectively, a divide-and-conquer bilevel optimization method, called DACBO, was proposed. DACBO first divided the transformed BOP into three sub-problems by grouping tasks into three independent sets. Afterward, for the first two sets, we devised analytical methods. In terms of the last one, DACBO employed DE and VNS as optimizers at the upper and lower levels, respectively. Moreover, a greedy method was designed to quickly construct a good initial solution for VNS. DACBO was applied to ten instances with differential scales and compared with two bilevel optimization methods. The results demonstrated the effectiveness of DACBO. In addition, we verified the superiority of the pricing scheme proposed in this paper.

REFERENCES

[1] Z. Yang, Y. Ding, Y. Jin, and K. Hao, "Immune-endocrine system inspired hierarchical coevolutionary multiobjective optimization algorithm for IoT service," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 164–177, Jan 2020.

[2] P. Huang, Y. Wang, K. Wang, and K. Yang, "Differential evolution with a variable population size for deployment optimization in a uav-assisted iot data collection system," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 3, pp. 324–335, 2020.

[3] M. S. Hossain and G. Muhammad, "Emotion recognition using secure edge and cloud computing," *Information Sciences*, vol. 504, pp. 589 – 601, 2019.

[4] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 468–476.

[5] P. Q. Huang, Y. Wang, K. Wang, and Z. Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 4228–4241, 2020.

[6] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[7] K. Wang, P. Huang, K. Yang, C. Pan, and J. Wang, "Unified offloading decision making and resource allocation in ME-RAN," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8159–8172, 2019.

[8] M. Zhang, J. Huang, and R. Zhang, "Wireless power transfer with information asymmetry: A public goods perspective," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 276–291, 2021.

[9] B. Clerckx, R. Zhang, R. Schober, D. W. K. Ng, D. I. Kim, and H. V. Poor, "Fundamentals of wireless information and power transfer: From RF energy harvester models to signal and system designs," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 1, pp. 4–33, Jan 2019.

[10] X. Hu, K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2375–2388, April 2018.

[11] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, Oct 2018.

[12] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Computing Surveys*, vol. 52, no. 5, Sep. 2019.

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

[14] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, Dec 2016.

[15] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, June 2019.

[16] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, April 2017.

[17] Y. Wang, Z. Y. Ru, K. Wang, and P. Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-uav-enabled mobile edge computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3984–3997, 2020.

[18] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, June 2018.

[19] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[20] Y. Du, K. Yang, K. Wang, G. Zhang, Y. Zhao, and D. Chen, "Joint resources and workflow scheduling in UAV-enabled wirelessly-powered mec for IoT systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 187–10 200, Oct 2019.

[21] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, March 2018.

[22] P. Liu, G. Xu, K. Yang, K. Wang, and Y. Li, "Joint optimization for residual energy maximization in wireless powered mobile-edge computing systems," *KSII Transactions on Internet & Information Systems*, vol. 12, no. 12, pp. 5614–5613, 2018.

[23] S. Kim, S. Park, M. Chen, and C. Youn, "An optimal pricing scheme for the energy-efficient mobile edge computation offloading with ofdma," *IEEE Communications Letters*, vol. 22, no. 9, pp. 1922–1925, Sep. 2018.

[24] K. Ma, C. Wang, J. Yang, C. Hua, and X. Guan, "Pricing mechanism with noncooperative game and revenue sharing contract in electricity market," *IEEE Transactions on Cybernetics*, vol. 49, no. 1, pp. 97–106, Jan 2019.

[25] J. Nicolaisen, V. Petrov, and L. Tesfatsion, "Market power and efficiency in a computational electricity market with discriminatory double-auction pricing," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 5, pp. 504–523, Oct 2001.

[26] S. Son and K. M. Sim, "A price- and-time-slot-negotiation mechanism for cloud service reservations," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 713–728, June 2012.

[27] N. C. Luong, P. Wang, D. Niyato, Y. Liang, Z. Han, and F. Hou, "Applications of economic and pricing models for resource management in 5g wireless networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3298–3339, 2019.

[28] X. Wang and L. Duan, "Dynamic pricing and capacity allocation of UAV-provided mobile services," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, April 2019, pp. 1855–1863.

[29] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain

networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4585–4600, June 2019.
[30] Z. Wang, K. Huang, X. Yang, X. Wan, Z. Fan, and Y. Xu, "Price-based resource allocation in wireless power transfer-enabled massive mimo networks," *Sensors*, vol. 19, no. 15, pp. 1–17, 2019.
[31] D. Han, W. Chen, and Y. Fang, "A dynamic pricing strategy for vehicle assisted mobile edge computing systems," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 420–423, April 2019.
[32] T. Ding, C. Li, C. Yan, F. Li, and Z. Bie, "A bilevel optimization model for risk assessment and contingency ranking in transmission system reliability evaluation," *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3803–3813, Sep. 2017.
[33] J. Xie, Y. Mei, A. T. Ernst, X. Li, and A. Song, "A bi-level optimization model for grouping constrained storage location assignment problems," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 385–398, Jan 2018.
[34] M. Razmara, G. R. Bharati, M. Shahbakhti, S. Paudyal, and R. D. Robinett, "Bilevel optimization framework for smart building-to-grid systems," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 582–593, March 2018.
[35] R. M. Kovacevic and G. C. Pflug, "Electricity swing option pricing by stochastic bilevel optimization: A survey and new approaches," *European Journal of Operational Research*, vol. 237, no. 2, pp. 389 – 403, May 2014.
[36] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, April 2018.
[37] J. Bard and J. Moore., "A branch and bound algorithm for the bilevel programming problem," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 2, pp. 281–292, 1990.
[38] H. Önal, "A modified simplex approach for solving bilevel linear programming problems," *European Journal of Operational Research*, vol. 67, no. 1, pp. 126 – 135, 1993.
[39] E. Aiyoshi and K. Shimizu, "A solution method for the static constrained stackelberg problem via penalty method," *IEEE Transactions on Automatic Control*, vol. 29, no. 12, pp. 1111–1114, December 1984.
[40] Yuping Wang, Yong-Chang Jiao, and Hong Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 2, pp. 221–232, 2005.
[41] A. Sinha, T. Soun, and K. Deb, "Using karush-kuhn-tucker proximity measure for solving bilevel optimization problems," *Swarm and Evolutionary Computation*, vol. 44, pp. 496 – 510, 2019.
[42] X. He, Y. Zhou, and Z. Chen, "Evolutionary bilevel optimization based on covariance matrix adaptation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 258–272, 2019.
[43] A. Sinha, P. Malo, and K. Deb, "Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping," *European Journal of Operational Research*, vol. 257, no. 2, pp. 395–411, 2017.
[44] J. S. Angelo, E. Krempser, and H. J. C. Barbosa, "Differential evolution for bilevel programming," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 470–477.
[45] Y. Huang, K. Wang, K. Gao, T. Qu, and H. Liu, "Jointly optimizing microgrid configuration and energy consumption scheduling of smart homes," *Swarm and Evolutionary Computation*, vol. 48, pp. 251 – 261, 2019.
[46] P. Q. Huang and Y. Wang, "A framework for scalable bilevel optimization: Identifying and utilizing the interactions between upper-level and lower-level variables," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 6, pp. 1150–1163, 2020.
[47] J. Feng, Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1320–1323, Oct 2019.
[48] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2603–2616, 2018.
[49] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
[50] X. Wang, Z. Dong, and L. Tang, "Multiobjective differential evolution with personal archive and biased self-adaptive mutation selection," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 12, pp. 5338–5350, 2020.
[51] A. Ghosh, S. Das, A. K. Das, and L. Gao, "Reusing the past difference vectors in differential evolution—a simple but significant improvement," *IEEE Transactions on Cybernetics*, vol. 50, no. 11, pp. 4821–4834, 2020.
[52] M. M. Islam, H. K. Singh, and T. Ray, "A surrogate assisted approach for single-objective bilevel optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 681–696, 2017.

Pei-Qiu Huang received the B.S. degree in automation and the M.S. degree in control theory and control engineering both from the Northeastern University, Shenyang, China, in 2014 and 2017, respectively, and the Ph.D. degree in control science and engineering, Central South University, Changsha, China, in 2021. His current research interests include evolutionary computation, bilevel optimization, and mobile edge computing.

Yong Wang (M'08–SM'17) received thePh.D. degree in control science and engineering from the Central South University, Changsha, China, in 2011.

He is a Professor with the School of Automation, Central South University, Changsha, China. His current research interests include intelligent learning and optimization and their interdisciplinary applications.

Dr. Wang is an Associate Editor of the IEEE Transactions on Evolutionary Computation and the Swarm and Evolutionary Computation. He was a recipient of Cheung Kong Young Scholar by the Ministry of Education, China, in 2018, and a Web of Science highly cited researcher in Computer Science in 2017 and 2018.

Kezhi Wang (M'15–SM'20) received the B.E. and M.E. degrees from the School of Automation, Chongqing University, China, in 2008 and 2011, respectively, and the Ph.D. degree in engineering from the University of Warwick, U.K., in 2015. He was a Senior Research Officer with the University of Essex, U.K., from 2015 to 2017. He is currently a Senior Lecturer with the Department of Computer and Information Sciences, Northumbria University, U.K. His research interests include mobile edge computing, intelligent reflection surface, and machine learning.

Supplementary File for "A Divide-and-Conquer Bilevel Optimization Algorithm for Jointly Pricing Computing Resources and Energy in Wireless Powered MEC"

NOMENCLATURE

α	The reward coefficient
$\mathbf{m}$	The mode set
$\mathbf{p}^b$	The broadcast power allocated to all tasks
$\mathbf{r}^c$	The computing resources allocated to all tasks
$\mathbf{v}^c$	The price of computing resources for all tasks
$\mathbf{v}^e$	The price of energy for all tasks
$\mathcal{N}$	The set of IoTDs
μ	The power conversion efficiency
B	The channel bandwidth
C_i	The computing resources required to complete U_i
D_i	The size of input data of U_i
d_i	The distance between the access point and the ith IoTD
E_i^c	The computing energy consumption of U_i on the MEC server
E_i^t	The transmission energy consumption of U_i
E_i^h	The energy harvested by the ith IoTD
E_i^l	The computing energy consumption of U_i on the ith IoTD
f^{do}	The total profit of DO
f^{sp}	The total profit of SP
f_c	The carrier frequency
f_i^{do}	The profit of DO from U_i
f_i^{sp}	The profit of SP from U_i
h_i	The channel gain between the access point and the ith IoTD
k_0	The effective cacitance coefficient of IoTDs
k_1	The effective capacitance coefficient of the MEC server
m_i	The execution mode of U_i
n	The number of IoTDs
N_0	The variance of complex Gaussian channel noise
p_i^b	The broadcast power allocated to U_i
p_i^t	The transmission power of the ith IoTD
$p_i^{b^*}$	The optimal broadcast power allocated to U_i
$p_{i,max}^b$	The upper bound of the broadcast power allocated to U_i
R_i	The transmission rate of the ith IoTD
r_i^c	The computing resources allocated to U_i
r_i^l	The computing capability of the ith IoTD
$r_i^{c^*}$	the optimal computing resource allocated to U_i
r_{max}^c	The computing capacity of the MEC server
T_i^c	The computing time of U_i on the MEC server
T^h	The energy harvesting time
T_i^l	The computing time of U_i on the ith IoTD
T_i^t	The transmission time of U_i
T_i	The completion time of U_i
$T_{i,max}$	The maximum time allowed to complete U_i
U_i	The task of the ith IoTD
v_0	The unit price of energy consumed by the access point
v_i^c	The unit price of computing resources for U_i
v_i^e	The unit price of energy for U_i

S-I. Additional Experiments and Discussions

A. Effectiveness of Task Grouping

TABLE S-I
RESULTS OF DACBO-WOT AND DACBO IN TERMS OF THE AVERAGE TOTAL PROFIT OF SP.

n	DACBO-WoT Mean (Std Dev)	DACBO Mean (Std Dev)
20	3.4563e+2 (1.9305e+0) $\approx$	3.4609e+2 (2.1235e+0)
40	5.3723e+2 (4.9855e+0) $\approx$	5.3763e+2 (4.6501e+0)
60	7.0398e+2 (1.4306e+1) $\approx$	7.0914e+2 (8.7725e+0)
80	7.6446e+2 (1.2579e+1) $\approx$	7.6755e+2 (1.3840e+1)
100	8.6672e+2 (1.5339e+1) $\uparrow$	8.7530e+2 (1.5404e+1)
120	9.2947e+2 (1.8687e+1) $\uparrow$	9.3915e+2 (1.5264e+1)
140	1.0475e+2 (1.8386e+1) $\uparrow$	1.0622e+3 (1.7471e+1)
160	1.1470e+3 (1.8176e+1) $\uparrow$	1.1585e+3 (2.1564e+1)
180	1.2686e+3 (2.4280e+1) $\uparrow$	1.2869e+3 (2.6290e+1)
200	1.3420e+3 (2.1437e+1) $\uparrow$	1.3621e+3 (2.4448e+1)
$\uparrow / \downarrow / \approx$	6/0/4	

In order to verify the effectiveness of task grouping, we designed a variant of DACBO, called DACBO-WoT. DACBO-WoT does not group tasks into three sets and directly adopts DE-VNS to jointly price computing resources and energy for all tasks. Table S-I presents the results of DACBO and DACBO-WoT. From Table S-I, in terms of the average total profit of SP, DACBO obtains the better result than DACBO-WoT on each instance. In addition, according to the Wilcoxon's rank-sum test at a 0.05 significance level, DACBO beats DACBO-WoG on six instances. Overall, the task grouping can improve the performance of DACBO, especially on instances with a larger number of IoTDs.

B. Effectiveness of the Greedy Method

TABLE S-II
RESULTS OF DACBO-WOG AND DACBO IN TERMS OF THE AVERAGE TOTAL PROFIT OF SP.

n	DACBO-WoG Mean (Std Dev)	DACBO Mean (Std Dev)
20	3.4661e+2 (8.7389e–1) ≈	3.4609e+2 (2.1235e+0)
40	5.3780e+2 (4.5728e+0) ≈	5.3763e+2 (4.6501e+0)
60	7.1531e+2 (1.1548e+1) ↓	7.0914e+2 (8.7725e+0)
80	7.4152e+2 (1.6904e+1) ↑	7.6755e+2 (1.3840e+1)
100	7.7291e+2 (2.0276e+1) ↑	8.7530e+2 (1.5404e+1)
120	7.9194e+2 (2.2444e+1) ↑	9.3915e+2 (1.5264e+1)
140	9.1949e+2 (2.8146e+1) ↑	1.0622e+3 (1.7471e+1)
160	1.0126e+3 (2.2288e+1) ↑	1.1585e+3 (2.1564e+1)
180	1.1179e+3 (3.2397e+1) ↑	1.2869e+3 (2.6290e+1)
200	1.2009e+3 (3.5696e+1) ↑	1.3621e+3 (2.4448e+1)
↑ / ↓ / ≈	7/1/2	

In this paper, a greedy method was used to generate an initial solution for VNS. To investigate the effect of the greedy method on the performance of DACBO, a variant of DACBO, called DACBO-WoG, was designed, in which the order of tasks was generated randomly instead of using the greedy method. The results of DACBO and DACBO-WoG are presented in Table S-II. From Table S-II, in terms of the average total profit of SP, DACBO outperforms DACBO-WoG on seven out of ten instances according to the Wilcoxon's rank-sum test at a 0.05 significance level. The comparison reveals that the greedy method does improve the performance of DACBO.

C. Performance of Different Pricing Schemes

In this subsection, we are interested in studying the performance of different pricing schemes. We compared our pricing scheme with the following three pricing schemes:

- Minimal pricing scheme: For each task, v_i^c and v_i^e are set to $v_{i,min}^c$ and $v_{i,min}^e$, respectively.
- Maximal pricing scheme: For each task, v_i^c and v_i^e are set to $v_{i,max}^c$ and $v_{i,max}^e$, respectively.
- Random pricing scheme: For each task, v_i^c and v_i^e are set to any value within $[v_{i,min}^c, v_{i,max}^c]$ and $[v_{i,min}^e, v_{i,max}^e]$, respectively.

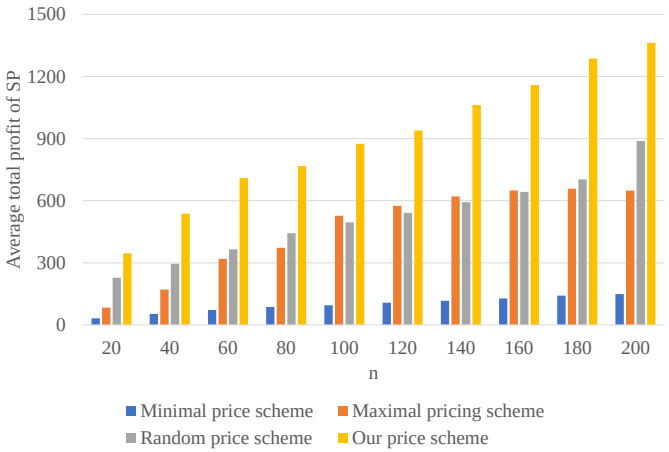

Fig. S-1. Results of the four pricing schemes in terms of the average total profit of SP.

Fig. S-1 presents the average total profits of SP resulting from the four pricing schemes on ten instances over 30 runs. As shown in Fig. S-1, our pricing scheme consistently performs the best among the four pricing schemes, followed by the random pricing scheme. Thus, the superiority of our pricing scheme has been verified.

D. Sensitivity in Relation to F and CR in DACBO

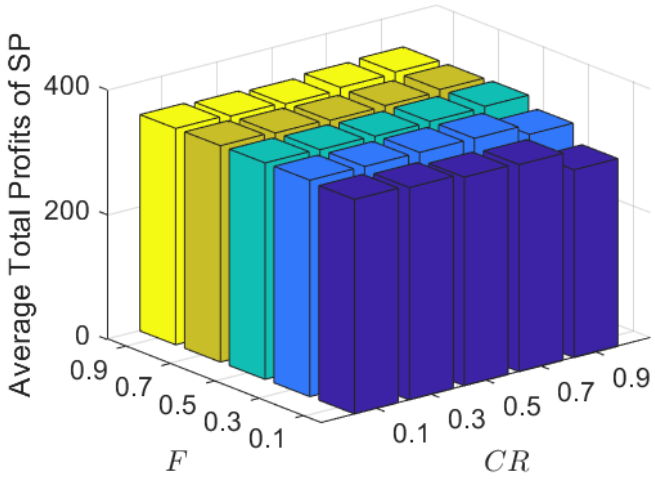

Fig. S-2. Results of DACBO with 25 different combinations of F and CR in the case of n = 20.

To study the sensitivities of F and CR in DACBO, we tested DACBO with 25 combinations of F and CR in the case of n = 20, in which F and CR are selected from two sets $F = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $CR = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, respectively. Fig. S-2 presents the average total profits of SP resulting from DACBO with 25 combinations of F and CR. It can be seen from Fig. S-2 that the performance of DACBO is not sensitive to the combinations of these two parameters, in addition to the combinations of $CR \geq 0.3$ and $F = 0.1$ as well as $CR = 0.9$ and $F = 0.3$.