Energy-Efficient Heuristic Computation Offloading With Delay Constraints in Mobile Edge Computing

Jing Mei, Zhao Tong, *Member, IEEE*, Kenli Li, *Member, IEEE*, Lianming Zhang, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—By offloading computation-intensive tasks to the edge cloud, mobile edge computing (MEC) has been regarded as an effective technology for enhancing computational capacity and extending the battery lifetime of mobile devices (MDs). However, due to the limitation of bandwidth and computing resources in MEC, unreasonable task offloading might lead to intensive resource competition, which recedes the performance gains benefit from offloading. When the tasks are latency-sensitive, a proper task offloading strategy is more important. Considering the heterogeneous delay constraints and resource competition comprehensively, we aim at minimizing the energy consumption of MDs subject to the individual delay constraints of tasks by jointly optimizing the task offloading and resource allocation in terms of wireless channel and remote computation capacity in a multi-MD MEC system in this paper. Due to the complexity of the primal optimization problem, a heuristic algorithm is devised. In the algorithm, a subset of tasks to be offloaded is incrementally constructed, and the corresponding offloading sub-problem is then repeatedly solved for this task subset using a two-stage algorithm until the total energy consumption can no longer be further reduced. The first stage of solving the sub-problem is to find the optimal full offloading scheme for the to-offload tasks, which is proved to be a convex optimization problem. For the task subset without a full offloading solution, an effective iterative algorithm is employed in the second stage where the channel allocation and computing resource allocation are optimized alternately. A great number of experiments are given to verify the performance of the proposed algorithm. We observe that the heuristic algorithm shows different performance when adopting different task ordering schemes. The proposed heuristic algorithm is evaluated against three reference schemes, and the results show that it can save up to 14.20% of energy consumption while guaranteeing the delay requirements of all tasks.**

***Index Terms*—Computation offloading, delay constraint, edge computing, energy optimization, resource competition.**

Manuscript received 25 May 2022; revised 11 October 2023; accepted 11 October 2023. Date of publication 16 October 2023; date of current version 13 December 2023. This work was supported in part by the Program of National Natural Science Foundation of China under Grants 62072174, 61502165, and 62372172, in part by National Natural Science Foundation of Hunan Province, China under Grants 2022JJ40278, 2022JJ30398, 2020JJ5370, and 2021JJ30455, in part by Scientific Research Fund of Hunan Provincial Education Department, China under Grant 22A0026. *(Corresponding authors: Zhao Tong; Lianming Zhang.)*

Jing Mei, Zhao Tong, and Lianming Zhang are with the College of Information Science and Engineering, Hunan Normal University, Changsha, Hunan 410081, China (e-mail: JingMei1988@163.com; tongzhao@hunnu.edu.cn; zlm@hunnu.edu.cn).

Kenli Li is with the College of Information Science and Engineering, Hunan University and National Supercomputing Center, Changsha, Hunan 410082, China (e-mail: lkl@hnu.edu.cn).

Keqin Li is with the College of Information Science and Engineering, Hunan University and National Supercomputing Center, Changsha, Hunan 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSC.2023.3324604, provided by the authors.

Digital Object Identifier 10.1109/TSC.2023.3324604

I. Introduction

ALONG with the fast development of 5G and Internet of Things (IoT) technology, billions of mobiles capable of computation and communication, e.g., mobile devices, sensors and wearable devices, are connected to the Internet via cellular networks. Due to the device size limitation, an IoT device often carries a capacity-limited battery and an energy-saving processor which is always of low performance. The low computation capacity cannot satisfy the high-performance computation requirement of many applications, e.g., augmented reality and image processing [1], [2], Additionally, the battery life of these devices is notably short and unsatisfactory. To tackle the two intrinsic shortages of IoT devices, one promising solution is to leverage mobile edge computing (MEC). MEC enables devices to offload computation-intensive tasks to nearby clouds located at the edge of a radio access network, such as a Wi-Fi access point (AP) or cellular base station. [3]. In this way, a mobile device (MD) can deal with tasks with high computation capacity requirements but consume less energy. Hence, computation offloading is an effective method to enhance computing power and lengthen the battery durability of MDs.

Although computation offloading can break the bottlenecks encountered by MDs effectively, not all tasks can benefit from task offloading, especially when numerous tasks are offloading to an MEC server simultaneously. In an MEC system, both the computation and communication resources are limited. When several devices request to offload tasks simultaneously, they compete for the channel resource and MEC computing resource. On the one hand, channel competition could result in network congestion, which affects the communication quality inevitably. As network congestion intensifies, the delay of offloading tasks to MEC increases, and could even exceed local execution delay. On the other hand, the devices tend to consume significantly higher amounts of energy for communication under a poor communication environment, potentially surpassing the energy consumption of local execution. Moreover, when the tasks have strict requirements in terms of execution delay, the problem caused by resource competition becomes more serious. Due to such reasons, a proper computation offloading strategy is crucial to exploit limited resources to handle more tasks while satisfying their specific requirements for delay and energy consumption [4].

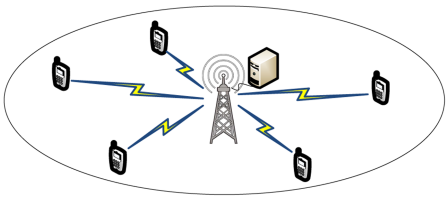


Fig. 1. Application scenario.

Computation offloading optimization has been studied from different perspectives in recent years. In this paper, we study the problem for a different application scenario, as depicted in Fig. 1. In the scenario, multiple mobile devices (MDs) request to offload their tasks to the nearby MEC server via the access point (AP). The MDs are heterogeneous in terms of computing and communication capabilities, power consumption, etc. Each MD owns a task to be processed which is of strict delay requirement and should be completed before its predefined deadline. The tasks considered in this paper are divisible, which can be offloaded to MEC and executed locally in parallel. The MDs compete for the limited channel bandwidth and MEC computing capacity, and the time-division multiple access (TDMA) model is considered in this paper. Our problem is to determine the partial offloading strategy by jointly optimizing the offloading decision and the resource allocation scheme for each MD such that all tasks can be completed before their respective deadlines while the sum of energy consumption is minimized.

Compared with the existing works, our novelty mainly lies in the following two aspects. First, we consider the delay-sensitive tasks which are characterized by their specific delay requirements. To guarantee the strict delay constraints, we adopt partial offloading in our framework, which makes the problem more challenging than that with binary offloading. Second, a TDMA-based MEC system with limited channel and computing resources is considered in this paper, hence, we optimize the channel allocation and MEC computing resource allocation jointly. Finding the optimal resource allocation scheme is also challenging since it depends on the task offloading ratio which needs to be determined as well.

We formulate our partial offloading problem as an energy minimization problem with delay and resource constraints. The solution to the offloading problem consists of two parts: task offloading decision which determines the ratio of tasks offloaded to MEC, and resource allocation decision which determines how much resources are allocated to each task. To achieve global optimization, the task offloading decision and the resource allocation decision should be optimized simultaneously. Since the optimal task offloading decision is unknown in advance, the objective function and the delay constraint involve a divide-by-zero error, which makes the problem much more complicated to tackle. To solve this problem, we propose an iterative heuristic algorithm. In the outer loop, the to-offload task set is determined and updated by adding new tasks according to a proposed discipline. In the inner loop, the offloading sub-problem is formulated to determine the optimal offloading strategy for the given to-offload task set such that the energy consumption is minimized. With the update of the to-offload task set, the whole process is repeated until the energy consumption cannot be reduced. A two-stage algorithm is also proposed to solve the offloading sub-problem. The first stage is to find the full offloading scheme for the to-offload tasks, which is proved to be convex optimization. For the task subset without a full offloading solution, an effective iterative binary search algorithm is employed in the second stage. This algorithm optimizes the allocation of channels and computing resources in an alternating manner.

The contributions of the paper is summarized as follows:

- The partial offloading problem is studied for a TDMA-based MEC system to minimize the total energy consumption of MDs while guaranteeing the delay constraints of all tasks. The task offloading ratio, the channel allocation, and the MEC computing resource allocation are optimized jointly. The improvement in the model makes the problem more realistic compared with the existing studies.
- An iterative heuristic approach is proposed to solve the primal optimization problem, which is transformed into a series of offloading sub-problems where the offload tasks are fixed and updated incrementally until the optimal offloading task set is found. Further, a two-stage algorithm is designed to solve the offloading sub-problem repeatedly for the expanding to-offload task set until the sum energy consumption cannot be optimized anymore.
- In the experiments, the performance of our algorithm under seven different task updating methods is compared first, which verifies the superiority of the adopted strategy. Besides, we compare our scheme with three offloading schemes, and the results show that the proposed offloading strategy can significantly reduce energy consumption.

The rest of the paper is organized as follows. In Section II, we provide a comprehensive description of the models used in this paper, including the system model, local computing model, and remote offloading model. Furthermore, we present a rigorous definition of our energy minimization problem. In Section III, we give a detailed analysis of our method and introduce our heuristic algorithm. In Section IV, we present the experimental data to evaluate the performance of the proposed algorithm and give some analysis and explanations. Section V concludes the work finally.

II. Models

In the following section, we will introduce the models adopted in this paper. Based on the models, the computation offloading optimization problem can be specified and studied rigorously.

A. System Model

In this paper, we consider a MEC system as depicted in Fig. 2. In the system, a cloudlet-enabled AP is considered. Within the radio access network (RAN), n MDs indexed by $\mathcal{N}=\{1,2,\ldots,n\}$ request to offload tasks to MEC. Assume that each MD only generates one task within a time slot, the task set is denoted as $\mathcal{J}=\{J_1,J_2,\ldots,J_n\}$ where J_i is submitted by MD_i. J_i is described by a tuple $(D_i,C_i,\mathcal{T}_i)$, where $\mathcal{T}_i$ is the delay requirement, D_i is the input data size, and C_i is the load-input

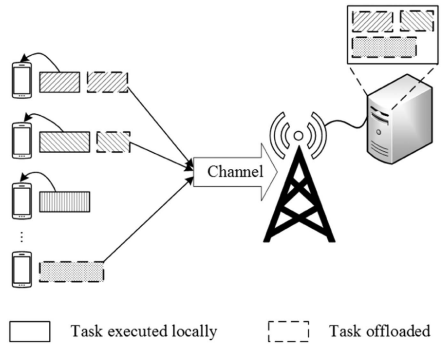


Fig. 2. System model.

data ratio (LDR) which is a constant and describes the number of CPU cycles required to process one bit of input data. C_i varies from different applications and can be obtained through offline measurement [5]. Hence, the execution requirement of J_i is D_iC_i. In the existing research, it is usually assumed that the execution requirements of tasks are known in advance [6].

The tasks considered in this paper are divisible, and each task can be executed on its host MD or offloaded to MEC fully or partially. Let α_i be the ratio of J_i executed locally and then $1-\alpha_i$ be the ratio of J_i offloaded to MEC. Specifically, when J_i is executed locally and not offloaded to MEC, the value of α_i is 1. Conversely, when J_i is fully offloaded to MEC, α_i is 0. If the value of α_i falls between 0 and 1, it indicates that J_i is partially offloaded to MEC and partially executed locally. When multiple tasks are concurrently offloaded to MEC, they compete for the limited wireless channel resources. We employ a TDMA technique for the channel access. Let β_i be the ratio of the time slot allocated to MD_i, it has

$$0 \leq \beta_i \leq 1, \sum_{i=1}^{n} \beta_i \leq 1.$$

In addition, due to the finite computing capacity of MEC, the tasks also compete for computing resources. Let γ_i be the ratio of computing resources of MEC allocated to J_i, it has

$$0 \leq \gamma_i \leq 1, \sum_{i=1}^{n} \gamma_i \leq 1.$$

The decision regarding task offloading and resource allocation is made by the edge cloud manager which is capable of gathering the necessary information such as the parameters of MDs, task characteristics, and resource state [7].

B. Local Computing Model

The computation power consumption P^i_{comp} (in Watts) of MD_i consists of two components, which are dynamic power consumption P^i_d and static power consumption P^i_s. The dynamic power consumption is represented as $P^i_d = \kappa_i S_i^{\lambda_i}$, where S_i is the computation speed (in GHz) of MD_i, and κ_i and λ_i are technology-dependent constants. The static power consumption P^i_s is also a constant. Therefore, we have $P^i_{comp} = \kappa_i S_i^{\lambda_i} + P^i_s$. The power model is adopted widely in many studies [3], [8], [9].

The local execution time (in seconds) of J_i on its host MD is

$$T^i_{local} = \frac{\alpha_i D_i C_i}{S_i},$$

and the energy consumption for the local computation (in Joules) of J_i is

$$E^i_{local} = P^i_{comp} T^i_{local} = \frac{(\kappa_i S_i^{\lambda_i} + P^i_s)\alpha_i D_i C_i}{S_i}. \quad (1)$$

C. Remote Offloading Model

The process of offloading tasks to MEC consists of two stages: data transmission and remote computation.

Data Transmission: Let P^i_{tra} be the transmission power (in Watts) of MD_i. When MD_i occupies the wireless channel exclusively, the communication rate S^i_{tra} is

$$S^i_{tra} = B \log_2 \left(1 + \frac{P^i_{tra} g_i}{\omega_i}\right),$$

where B presents the channel bandwidth, g_i denotes the channel gain between MD_i and MEC, and $\omega_i = g_i/(I_i + \sigma_i^2)$ where I_i is the interference on the communication channel caused by other devices that transmit data to the same MEC, and σ_i^2 denotes the power spectrum density of additive white Gaussian noise [10], [11], [12].

As mentioned before, when multiple MDs offload tasks to MEC simultaneously, they occupy the wireless channel in turn, which affects the communication rate greatly. Considering the TDMA channel system for an arbitrary time slot and the ratio β_i of the time slot allocated to MD_i, the achievable communication rate between MD_i and MEC is

$$r^i_{tra} = \beta_i S^i_{tra}.$$

Then, the communication time of delivering data from MD_i to the MEC server is

$$T^i_{tra} = \frac{(1-\alpha_i)D_i}{r^i_{tra}} = \frac{(1-\alpha_i)D_i}{\beta_i S^i_{tra}},$$

and the transmission energy consumption is

$$E^i_{tra} = T^i_{tra} P^i_{tra} = \frac{(1-\alpha_i)D_i P^i_{tra}}{\beta_i S^i_{tra}}.$$

Remote Computation: Similarly, when multiple tasks are offloaded to MEC for execution simultaneously, computing resources in MEC are also being competed for. Since γ_i is the ratio of computing resource of MEC allocated to execute task J_i, the remote computation time of J_i on MEC is

$$T^i_{MEC} = \frac{(1-\alpha_i)D_i C_i}{\gamma_i S_{MEC}}.$$

Our objective is to minimize energy consumption of MDs, so it is unnecessary to consider the computation energy consumption of J_i on MEC.

According to the above analysis, the total execution time of J_i for remote offloading is

$$T^i_{remote} = T^i_{tra} + T^i_{MEC} = \frac{(1-\alpha_i)D_i}{\beta_i S^i_{tra}} + \frac{(1-\alpha_i)D_i C_i}{\gamma_i S_{MEC}},$$

and the total energy consumption of J_i for remote offloading is

$$E^i_{remote} = E^i_{tra} = \frac{(1-\alpha_i)D_iP^i_{tra}}{\beta_i S^i_{tra}}.$$

Because the size of the output data of a task is much smaller than the input data, we ignore the transmission time and energy consumption of the computation results returned from MEC to MDs in this paper. The communication and computation models are adopted widely by many researchers [13], [14].

D. Problem Formulation

Given that the local execution and remote offloading of a task occur simultaneously, the execution delay of J_i is calculated as

$$T_i = \max\{T^i_{local}, T^i_{remote}\},$$

and it cannot exceed the delay constraint $\mathcal{T}_i$, that is,

$$\max\left\{\frac{\alpha_i D_i C_i}{S_i}, (1-\alpha_i)\left(\frac{D_i}{\beta_i S^i_{tra}} + \frac{D_i C_i}{\gamma_i S_{MEC}}\right)\right\} \le \mathcal{T}_i, \forall i \in \mathcal{N}. \tag{2}$$

The total energy consumption of all MDs is

$$\begin{aligned} E &= \sum_{i=1}^{n} E_i = \sum_{i=1}^{n}(E^i_{local} + E^i_{remote}) \\ &= \sum_{i=1}^{n}\left(\frac{\alpha_i D_i C_i(\kappa_i S_i^{\lambda_i} + P^i_s)}{S_i} + \frac{(1-\alpha_i)D_iP^i_{tra}}{\beta_i S^i_{tra}}\right). \end{aligned} \tag{3}$$

Problem Formulation: Given a MEC system denoted by MEC $\triangleq (S_{MEC}, B)$ and n MDs indexed by $\mathcal{N} = \{1, 2, \ldots, n\}$ where MD$_i \triangleq (S_i, \kappa_i, \lambda_i, P^i_s, P^i_{tra})$. Each MD generates a divisible task J_i per time slot and $J_i \triangleq (D_i, C_i, \mathcal{T}_i)$. All MDs request to offload tasks to MEC. Our *computation offloading problem* is to find an offloading decision $\{(\alpha_i, \beta_i, \gamma_i) \mid 1 \le i \le n\}$ such that the total energy consumption of all MDs is minimized while the delay constraints of all tasks are satisfied, which can be formulated as an energy optimization problem with delay and resource constraints as follows:

$$(\mathbf{P1}) \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} \sum_{i=1}^{n}\left(\frac{\alpha_i D_i C_i(\kappa_i S_i^{\lambda_i} + P^i_s)}{S_i} + \frac{(1-\alpha_i)D_iP^i_{tra}}{\beta_i S^i_{tra}}\right)$$

$$\text{s.t.} \quad \frac{\alpha_i D_i C_i}{S_i} \le \mathcal{T}_i, \forall i \in \mathcal{N}, \tag{4a}$$

$$(1-\alpha_i)\left(\frac{D_i}{\beta_i S^i_{tra}} + \frac{D_i C_i}{\gamma_i S_{MEC}}\right) \le \mathcal{T}_i, \forall i \in \mathcal{N}, \tag{4b}$$

$$\sum_{i=1}^{n}\beta_i \le 1, 0 \le \beta_i \le 1, \forall i \in \mathcal{N}, \tag{4c}$$

$$\sum_{i=1}^{n}\gamma_i \le 1, 0 \le \gamma_i \le 1, \forall i \in \mathcal{N}, \tag{4d}$$

$$0 \le \alpha_i \le 1, \forall i \in \mathcal{N}, \tag{4e}$$

where $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, $\boldsymbol{\beta} = \{\beta_1, \beta_2, \ldots, \beta_n\}$, and $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$.

In problem $(\mathbf{P1})$, constraints (4a) and (4b) denote the delay requirement of task J_i $(1 \le i \le n)$. Constraints (4c) and (4d) denote the resource constraints. Constraint (4e) is the offloading indicator constraint.

This problem has $3n$ variables, which are $\alpha_i, \beta_i, \gamma_i$ $(1 \le i \le n)$. It is a continuous optimization problem and there is absolutely no closed-form solution. Hence, we design a heuristic method to solve it in the next section.

III. Heuristic Algorithms

In this section, a heuristic method is introduced to solve the energy minimization problem. To make the algorithm more understandable, we first give an algorithm framework, and then introduce the details step by step.

A. The Algorithm Framework

Analyzing problem $(\mathbf{P1})$, its solution consists of three parts: the task division scheme $\boldsymbol{\alpha}$, the channel resource allocation scheme $\boldsymbol{\beta}$, and the MEC computing resource allocation scheme $\boldsymbol{\gamma}$. The three parts are affected mutually, which makes the problem complicated to solve. To find the optimal solution of $(\mathbf{P1})$, we transform it into a series of optimization sub-problems.

In the primal problem, the range of α_i is [0,1] for all tasks. If $\alpha_i = 1$, the entire task of J_i is executed on its host MD, and it is unnecessary to allocate resources for it, that is, β_i and γ_i are equal to 0, which triggers the *divide-by-zero* error. To avoid that, we first determine the tasks which are bound to be offloaded to MEC, and then find the offloading decision for these tasks.

Let $\mathcal{S}$ be the to-offload task set. Once $\mathcal{S}$ is fixed, the solution space of $\boldsymbol{\alpha}$ becomes $\{\alpha_i \mid 0 \le \alpha_i < 1, i \in \mathcal{S}; \alpha_i = 1, i \notin \mathcal{S}\}$, and the resource allocation vectors become $\boldsymbol{\beta} = \{\beta_i \mid 0 < \beta_i \le 1, i \in \mathcal{S}; \beta_i = 0, i \notin \mathcal{S}\}$ and $\boldsymbol{\gamma} = \{\gamma_i \mid 0 < \gamma_i \le 1, i \in \mathcal{S}; \gamma_i = 0, i \notin \mathcal{S}\}$. That means, we just need to consider the offloading strategy for $\mathcal{S}$, and the sub-problem $(\mathbf{P2})$ is constructed as

$$(\mathbf{P2}) \min_{\boldsymbol{\alpha}_\mathcal{S},\boldsymbol{\beta}_\mathcal{S},\boldsymbol{\gamma}_\mathcal{S}} \sum_{i\in\mathcal{S}}\left(\frac{\alpha_i D_i C_i(\kappa_i S_i^{\lambda_i} + P^i_s)}{S_i} + \frac{(1-\alpha_i)D_iP^i_{tra}}{\beta_i S^i_{tra}}\right)$$

$$\text{s.t.} \quad \frac{\alpha_i D_i C_i}{S_i} \le \mathcal{T}_i, \forall i \in \mathcal{S}, \tag{5a}$$

$$(1-\alpha_i)\left(\frac{D_i}{\beta_i S^i_{tra}} + \frac{D_i C_i}{\gamma_i S_{MEC}}\right) \le \mathcal{T}_i, \forall i \in \mathcal{S}, \tag{5b}$$

$$\sum_{i\in\mathcal{S}}\beta_i \le 1, \sum_{i\in\mathcal{S}}\gamma_i \le 1, \tag{5c}$$

$$0 < \beta_i \le 1, 0 < \gamma_i \le 1, \forall i \in \mathcal{S}, \tag{5d}$$

$$0 \le \alpha_i < 1, \forall i \in \mathcal{S}, \tag{5e}$$

where $\boldsymbol{\alpha}_\mathcal{S} = \{\alpha_i \mid i \in \mathcal{S}\}$, $\boldsymbol{\beta}_\mathcal{S} = \{\beta_i \mid i \in \mathcal{S}\}$ and $\boldsymbol{\gamma}_\mathcal{S} = \{\gamma_i \mid i \in \mathcal{S}\}$.

Solving problem $(\mathbf{P2})$ is significantly less challenging compared to problem $(\mathbf{P1})$. Whereas, how to determine the optimal

$\mathcal{S}$ becomes a key issue. To determine it, we first construct an initial $\mathcal{S}$ which contains all tasks that cannot be processed locally. Then, $\mathcal{S}$ is updated by adding new tasks based on specific principles. After each update, the sub-problem (P2) is solved to calculate the sum of energy consumption under the updated $\mathcal{S}$. This process is repeated until the total energy consumption is not reduced anymore. Next, we will discuss the process of initializing and updating $\mathcal{S}$ as outlined below.

- *Initialization of $\mathcal{S}$:* If the computing capacity of a MD is sufficient to guarantee the latency constraint of a task, i.e., $D_iC_i > \mathcal{T}_iS_i$, then the task can be assigned to be executed locally, otherwise, it must be offloaded to MEC partially or fully. Therefore, the set $\mathcal{S}$ is initialized as the tasks which cannot be executed locally.
- *Update of $\mathcal{S}$:* In order to determine the method of updating $\mathcal{S}$, it is necessary to establish a priority for each task. Let

$$X_i = \frac{D_iC_i(\kappa_i S_i^{\lambda_i} + P_s^i)}{S_i}, Y_i = \frac{D_iP_{tra}^i}{S_{tra}^i},$$

the energy consumption of J_i can be simplified as

$$E_i = \alpha_i X_i + \frac{Y_i}{\beta_i}(1-\alpha_i) = \left(X_i - \frac{Y_i}{\beta_i}\right)\alpha_i + \frac{Y_i}{\beta_i}. \quad (6)$$

Offloading tasks with a higher $X_i - Y_i/\beta_i$ value can result in a greater reduction in energy consumption compared to offloading tasks with a lower $X_i - Y_i/\beta_i$ value. Since the β_i value is not predetermined, the priority of task J_i is defined as the value of X_i/Y_i. In each iteration, the task with the highest priority, denoted as J_k, is selected and added to the set $\mathcal{S}$, that is, $\mathcal{S}$ is updated as $\mathcal{S} \cup J_k$.

The algorithm frame is described as Algorithm 1.

Algorithm 1: Main Algorithm.

Input: Task set $\mathcal{J} = (J_1, J_2, \ldots, J_n)$;
Output: $\boldsymbol{\alpha}_{opt}$, $\boldsymbol{\beta}_{opt}$, $\boldsymbol{\gamma}_{opt}$, and E_{opt};
1: Calculate the priority for all tasks;
2: Initial the to-offload task set $\mathcal{S}$ and let $L \leftarrow \mathcal{J} - \mathcal{S}$;
3: $E_{opt} \leftarrow \infty$;
4: **while** L is not null **do**
5: $\boldsymbol{\alpha}_L \leftarrow \{\alpha_i = 1 \mid J_i \in L\}$, $\boldsymbol{\beta}_L \leftarrow \{\beta_i = 0 \mid J_i \in L\}$, $\boldsymbol{\gamma}_L \leftarrow \{\gamma_i = 0 \mid J_i \in L\}$;
6: $E_L \leftarrow$ the total energy consumption of tasks in L calculated by (1);
7: Solve the problem $(\mathbf{P2})$, and obtain the optimal offloading scheme $(\boldsymbol{\alpha}_\mathcal{S}, \boldsymbol{\beta}_\mathcal{S}, \boldsymbol{\gamma}_\mathcal{S})$ and the corresponding optimal energy consumption E_S for $\mathcal{S}$;
8: **if** $E_S + E_L < E_{opt}$ **then**
9: $E_{opt} \leftarrow E_S + E_L$;
10: $(\boldsymbol{\alpha}_{opt}, \boldsymbol{\beta}_{opt}, \boldsymbol{\gamma}_{opt}) \leftarrow (\boldsymbol{\alpha}_L \cup \boldsymbol{\alpha}_\mathcal{S}, \boldsymbol{\beta}_L \cup \boldsymbol{\beta}_\mathcal{S}, \boldsymbol{\gamma}_L \cup \boldsymbol{\gamma}_\mathcal{S})$;
11: **else**
12: **break**;
13: **end if**
14: $J_{cur} \leftarrow$ the task with the highest priority in L;
15: $\mathcal{S} \leftarrow \mathcal{S} \cup J_{cur}$, $L \leftarrow \mathcal{J} - \mathcal{S}$;
16: **end while**

B. Solution for the Offloading Sub-Problem

In Algorithm 1, the key issue is to solve the problem $(\mathbf{P2})$ to find the optimal offloading for $\mathcal{S}$ (see line 7). However, the complexity of this problem renders it impractical to derive a closed-form solution. Hence, we try to find the numerical solutions through two distinct stages:

- First, the feasibility of completely offloading the tasks in $\mathcal{S}$ to MEC is assessed. If the statement is true, it is necessary to determine the most efficient resource allocation scheme and its corresponding energy consumption. *(Full offloading stage)*
- Second, if the tasks in $\mathcal{S}$ cannot be completely offloaded to MEC, an iterative binary search approach is employed to determine a partial offloading scheme for these tasks. *(Partial offloading stage)*

The details of the two stages are introduced in Sections III-B1 and III-B2, respectively.

1) Full Offloading Stage: To judge if $\mathcal{S}$ can be fully offloaded to MEC, it is crucial to ascertain the existence of a resource allocation scheme when $\boldsymbol{\alpha} = \{\alpha_i \mid \alpha_i = 0, i \in \mathcal{S}; \alpha_i = 1, i \notin \mathcal{S}\}$. In this scenario, the problem $(\mathbf{P2})$ can be reformulated as

$$(\mathbf{P3}) \quad \min_{\boldsymbol{\beta}_\mathcal{S}, \boldsymbol{\gamma}_\mathcal{S}} \quad \sum_{i \in \mathcal{S}} \frac{D_iP_{tra}^i}{\beta_i S_{tra}^i}$$

$$\text{s.t.} \quad \frac{D_i}{\beta_i S_{tra}^i} + \frac{D_iC_i}{\gamma_i S_{MEC}} \le T_i, \forall i \in \mathcal{S}, \quad (7a)$$

$$\sum_{i \in \mathcal{S}} \beta_i \le 1, \sum_{i \in \mathcal{S}} \gamma_i \le 1, \quad (7b)$$

$$0 < \beta_i \le 1, 0 < \gamma_i \le 1, \forall i \in \mathcal{S}. \quad (7c)$$

Theorem 3.1: Problem $(\mathbf{P3})$ is a convex optimization problem.

Proof: The proof is given in the supplementary material. □

Given that problem $(\mathbf{P3})$ is a convex optimization problem, we can use CVX, the convex optimization tool in Matlab, to solve it. CVX is a Matlab-based modeling system for convex optimization [15], which can find the optimal solution for problem $(\mathbf{P3})$ if it is solvable, which includes determining the resource allocation scheme $(\boldsymbol{\beta}_\mathcal{S}, \boldsymbol{\gamma}_\mathcal{S})$ and the corresponding energy consumption E_{opt}. If it is unsolvable, the process enters the partial offloading stage.

2) Partial Offloading Stage: If the computing capacity of MEC is insufficient, a feasible solution for problem $(\mathbf{P3})$ cannot be found. Hence, some tasks in $\mathcal{S}$ can only be partially offloaded to MEC. Excepting $(\boldsymbol{\beta}_\mathcal{S}, \boldsymbol{\gamma}_\mathcal{S})$, the task partial offloading scheme $\boldsymbol{\alpha}_\mathcal{S}$ should be determined for $\mathcal{S}$ as well. Because $\boldsymbol{\alpha}_\mathcal{S}$, $\boldsymbol{\beta}_\mathcal{S}$, and $\boldsymbol{\gamma}_\mathcal{S}$ are affected mutually, finding the optimal combination of $(\boldsymbol{\alpha}_\mathcal{S}, \boldsymbol{\beta}_\mathcal{S}, \boldsymbol{\gamma}_\mathcal{S})$ is complicated. In this stage, we design an iterative binary search method to determine the sub-optimal solution step by step, and the algorithm framework is given in Algorithm 2. The detailed analysis and the solution process for each step is introduced later.

Algorithm 2: Main Algorithm for Partial Offloading Solution.

Input: The to-offload task set $\mathcal{S}$;
Output: $\boldsymbol{\alpha}_{\mathcal{S}}^{opt}, \boldsymbol{\beta}_{\mathcal{S}}^{opt}, \boldsymbol{\gamma}_{\mathcal{S}}^{opt}$ and $E_{\mathcal{S}}^{opt}$ of $\mathcal{S}$;
1: Initialize $\boldsymbol{\gamma}_{\mathcal{S}}$;
2: $E_{\mathcal{S}}^{opt} \leftarrow \infty$;
3: **while** true **do**
4: Find the optimal $\boldsymbol{\beta}_{\mathcal{S}}$ under the given $\boldsymbol{\gamma}_{\mathcal{S}}$ by Algorithm 3; (**Section III-B2-b**))
5: Calculate $\boldsymbol{\alpha}_{\mathcal{S}}$ under $\boldsymbol{\beta}_{\mathcal{S}}$ and $\boldsymbol{\gamma}_{\mathcal{S}}$; (**Section III-B2-a**))
6: Calculate the energy consumption $E_{\mathcal{S}}$ under $\boldsymbol{\alpha}_{\mathcal{S}}$, $\boldsymbol{\beta}_{\mathcal{S}}$ and $\boldsymbol{\gamma}_{\mathcal{S}}$ using (3);
7: **if** $E_{\mathcal{S}} < E_{\mathcal{S}}^{opt}$ **then**
8: $\boldsymbol{\alpha}_{\mathcal{S}}^{opt} \leftarrow \boldsymbol{\alpha}_{\mathcal{S}}, \boldsymbol{\beta}_{\mathcal{S}}^{opt} \leftarrow \boldsymbol{\beta}_{\mathcal{S}}, \boldsymbol{\gamma}_{\mathcal{S}}^{opt} \leftarrow \boldsymbol{\gamma}_{\mathcal{S}}$, $E_{\mathcal{S}}^{opt} \leftarrow E_{\mathcal{S}}$;
9: Update $\boldsymbol{\gamma}_{\mathcal{S}}$ based on $\boldsymbol{\beta}_{\mathcal{S}}$ by Algorithm 6; (**Section III-B2-b**))
10: **else**
11: **break;**
12: **end if**
13: **end while**

a) Determination of optimal $\boldsymbol{\alpha}_{\mathcal{S}}$: First, we analyze how α_i is affected by β_i and γ_i, and give the calculation of optimal α_i under a given combination of β_i and γ_i.

According to (6), it can be easily observed that the energy consumption of MD_i is a monotonic function in terms of α_i once β_i is fixed. Assume that the value range of α_i is expressed as

$$\underset{\sim}{\alpha_i} \le \alpha_i \le \widetilde{\alpha}_i,$$

where $\underset{\sim}{\alpha_i}$ and $\widetilde{\alpha}_i$ are the lower and upper boundaries of α_i, respectively. Then, the optimal value of α_i can be discussed as follows:

- If $X_i < Y_i/\beta_i$, that is, $\beta_i < Y_i/X_i$, then E_i is a decreasing function of α_i, the optimal α_i is $\widetilde{\alpha}_i$;
- If $X_i > Y_i/\beta_i$, that is, $\beta_i > Y_i/X_i$, then E_i is an increasing function of α_i, the optimal α_i is $\underset{\sim}{\alpha_i}$;
- If $X_i = Y_i/\beta_i$, that is, $\beta_i = Y_i/X_i$, then E_i is a constant. The task J_i is executed locally in priority, so the optimal α_i is $\widetilde{\alpha}_i$.

Combining constraints (5a), (5b), and (5e) of problem ($\mathbf{P2}$), the solution space of α_i is updated as

$$\max\left\{0, 1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}\right\} \le \alpha_i \le \min\left\{\frac{\mathcal{T}_i S_i}{D_i C_i}, 1\right\}.$$

Since the value range of α_i is varying with different β_i and γ_i, the optimal value of α_i can be discussed in more detail as follows:

Case 1: $\mathcal{T}_i S_i/(D_i C_i) > 1$.

- When $\beta_i \le Y_i/X_i$, the optimal value of α_i is equal to 1, denoted as $\alpha_i^* = 1$, since
$$\max\left\{0, 1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}\right\} \le 1$$
holds always.
- When $\beta_i > Y_i/X_i$ and
$$1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1} \le 0,$$
that is,
$$\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i}{D_i},$$
the optimal α_i^* is $\alpha_i^* = 0$.
- When $\beta_i > Y_i/X_i$ and
$$0 < 1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1} \le 1,$$
that is,
$$\frac{\mathcal{T}_i}{D_i} < \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}},$$
the optimal value α_i^* is
$$\alpha_i^* = 1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}.$$

Case 2: $\mathcal{T}_i S_i/(D_i C_i) \le 1$.

- When $\beta_i \le Y_i/X_i$ and
$$\max\left\{0, 1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}\right\} \le \frac{\mathcal{T}_i S_i}{D_i C_i},$$
that is,
$$\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i C_i}{D_i C_i - \mathcal{T}_i S_i},$$
the optimal value α_i^* is
$$\alpha_i^* = \frac{\mathcal{T}_i S_i}{D_i C_i}.$$
- When $\beta_i > Y_i/X_i$ and
$$1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1} \le 0,$$
that is,
$$\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i}{D_i},$$
the optimal value α_i^* is $\alpha_i^* = 0$.
- When $\beta_i > Y_i/X_i$ and
$$0 < 1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1} \le \frac{\mathcal{T}_i S_i}{D_i C_i},$$
that is,
$$\frac{\mathcal{T}_i}{D_i} < \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i C_i}{D_i C_i - \mathcal{T}_i S_i},$$

the optimal value α_i^* is

$$\alpha_i^* = 1 - \frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}.$$

The calculation of the optimal α_i^* under different cases is summarized as the following equations.

Case 1: When $\mathcal{T}_i S_i/(D_i C_i) > 1$,

$$\alpha_i^* = \begin{cases} 1, & \text{if } \beta_i \le \frac{Y_i}{X_i}; \\ 1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}, & \\ \quad \text{if } \beta_i > \frac{Y_i}{X_i}, \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} > \frac{\mathcal{T}_i}{D_i}; \\ 0, & \text{if } \beta_i > \frac{Y_i}{X_i}, \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i}{D_i}; \\ \text{no solution}, & \text{otherwise}, \end{cases} \tag{8}$$

and the corresponding energy consumption of MD_i is

$$E_i(\beta_i,\gamma_i) = \begin{cases} X_i, & \text{if } \beta_i \le \frac{Y_i}{X_i}; \\ (X_i - \frac{Y_i}{\beta_i})\left[1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}\right]+\frac{Y_i}{\beta_i}, & \\ \quad \text{if } \beta_i > \frac{Y_i}{X_i}, \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} > \frac{\mathcal{T}_i}{D_i}; \\ \frac{Y_i}{\beta_i}, & \text{if } \beta_i > \frac{Y_i}{X_i}, \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i}{D_i}; \\ \infty, & \text{otherwise}. \end{cases} \tag{9}$$

Case 2: When $\mathcal{T}_i S_i/(D_i C_i) \le 1$,

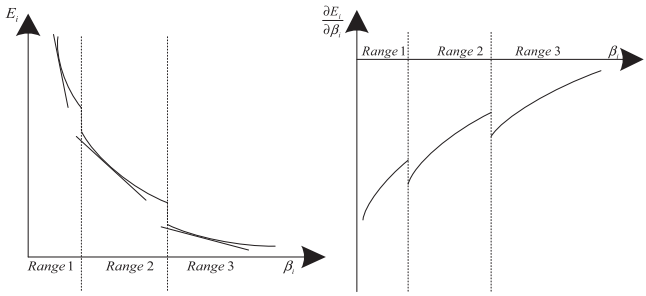
$$\alpha_i^* = \begin{cases} \frac{\mathcal{T}_i S_i}{D_i C_i}, & \text{if } \beta_i \le \frac{Y_i}{X_i}, \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i C_i}{D_i C_i - \mathcal{T}_i S_i}; \\ 1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}, & \\ \quad \text{if } \beta_i > \frac{Y_i}{X_i}, \frac{\mathcal{T}_i}{D_i} < \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i C_i}{D_i C_i - \mathcal{T}_i S_i}; \\ 0, & \text{if } \beta_i > \frac{Y_i}{X_i}, \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i}{D_i}; \\ \text{no solution}, & \text{otherwise}, \end{cases} \tag{10}$$

and the corresponding energy consumption of MD_i is

$$E_i(\beta_i,\gamma_i) = \begin{cases} (X_i - \frac{Y_i}{\beta_i})\frac{\mathcal{T}_i S_i}{D_i C_i}+\frac{Y_i}{\beta_i}, & \\ \quad \text{if } \beta_i \le \frac{Y_i}{X_i}, \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i C_i}{D_i C_i - \mathcal{T}_i S_i}; \\ (X_i - \frac{Y_i}{\beta_i})\left[1-\frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}\right]+\frac{Y_i}{\beta_i}, & \\ \quad \text{if } \beta_i > \frac{Y_i}{X_i}, \frac{\mathcal{T}_i}{D_i} < \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i C_i}{D_i C_i - \mathcal{T}_i S_i}; \\ \frac{Y_i}{\beta_i}, & \text{if } \beta_i > \frac{Y_i}{X_i}, \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i}{D_i}; \\ \infty, & \text{otherwise}. \end{cases} \tag{11}$$

Based on (8) and (10), the optimal α_i can be calculated directly under a given β_i and γ_i for all $i \in \mathcal{S}$. Till now, the key to solving the optimization problem becomes to find the optimal β_i and γ_i for all tasks, and problem $(\mathbf{P2})$ can be simplified as

$$(\mathbf{P4}) \min_{\boldsymbol{\beta}_\mathcal{S}, \boldsymbol{\gamma}_\mathcal{S}} \quad E = \sum_{i\in\mathcal{S}} E_i(\beta_i,\gamma_i)$$
$$\text{s.t.} \quad \sum_{i\in\mathcal{S}} \beta_i \le 1, \sum_{i\in\mathcal{S}} \gamma_i \le 1, \tag{12a}$$



(a) The function curve of E_i . (b) The function curve of $\partial E_i/\partial\beta_i$.

Fig. 3. The function curve of E_i in term of β_i.
Range 1: $\beta_i \le Y_i/X_i$;
Range 2: $\beta_i > Y_i/X_i$ and $\frac{\mathcal{T}_i}{D_i} < \frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i S_i}{D_i C_i}$;
Range 3: $\beta_i > Y_i/X_i$ and $\frac{1}{\beta_i S_{tra}^i}+\frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i}{D_i}$.

$$0 < \beta_i \le 1, 0 < \gamma_i \le 1, \forall i \in \mathcal{S}. \tag{12b}$$

The subsequent section presents an iterative binary search approach for addressing problem $(\mathbf{P4})$, wherein the $\boldsymbol{\beta}_\mathcal{S}$ and $\boldsymbol{\gamma}_\mathcal{S}$ parameters are progressively optimized until no further improvement in energy consumption can be achieved.

b) Determination of optimal $\boldsymbol{\beta}_\mathcal{S}$ under a given $\boldsymbol{\gamma}_\mathcal{S}$: In the section, we introduce how to determine the optimal $\boldsymbol{\beta}_\mathcal{S}$ under a given $\boldsymbol{\gamma}_\mathcal{S}$. Under a given $\boldsymbol{\gamma}_\mathcal{S}$, problem $(\mathbf{P4})$ is simplified as:

$$2(\mathbf{P5}) \min_{\boldsymbol{\beta}_\mathcal{S}} \quad E = \sum_{i\in\mathcal{S}} E_i(\beta_i)$$
$$\text{s.t.} \quad \sum_{i\in\mathcal{S}} \beta_i \le 1, 0 < \beta_i \le 1, \forall i \in \mathcal{S}.$$

Before solving this problem, some analyses are given.

Theorem 3.2: E_i is a piece-wise function and decreases monotonically with respect to β_i. In each continuous interval, $\partial E_i/\partial\beta_i \le 0$ and $\partial^2 E_i/\partial\beta_i^2 \ge 0$.

Proof: The proof is given in the supplementary material, and the function curve of $E_i(\beta_i)$ is shown as Fig. 3. □

Theorem 3.3: Given an optimization problem

$$\min_{\boldsymbol{x}} \quad F = \sum_{i=1}^{m} f_i(x_i),$$
$$\text{s.t.} \quad \sum_{i=1}^{m} x_i \le L, x_i > 0, \forall i,$$

where $f_i(x_i)$ is a continuous function which $\partial f_i/\partial x_i \le 0$ and $\partial^2 f_i/\partial x_i^2 \ge 0$, for $1 \le i \le m$, then the optimal solution must satisfy

$$\sum_{i=1}^{m} x_i = L,$$

and

$$\frac{\partial f_i}{\partial x_i} = \frac{\partial f_j}{\partial x_j}, \text{ for any } 1 \le i < j \le m.$$

Proof: The proof is given in the supplementary material. □

Algorithm 3: Finding $\boldsymbol{\beta}_{\mathcal{S}}^{*}$ such that $\sum_{i\in\mathcal{S}}\beta_i = 1$.

Input: $\boldsymbol{\gamma}_{\mathcal{S}}$, $B_{\min}$ and $B_{\max}$;
Output: The optimal $\boldsymbol{\beta}_{\mathcal{S}}^{*}$;
1: $\xi_{left} \leftarrow B_{\min}, \xi_{right} \leftarrow B_{\max}$,
2: Calculate $\boldsymbol{\beta}_{left}$ and $\boldsymbol{\beta}_{right}$ by Algorithm 4 with inputs ξ_{left} and ξ_{right}, respectively;
3: **if** sum$(\boldsymbol{\beta}_{right}) < 1$ **then**
4: $\quad\boldsymbol{\beta}_{opt} \leftarrow \boldsymbol{\beta}_{right}$;
5: **else**
6: $\quad$**if** sum$(\boldsymbol{\beta}_{left}) > 1$ **then**
7: $\quad\quad\boldsymbol{\beta}_{opt} \leftarrow \oslash$;
8: $\quad$**else**
9: $\quad\quad$**while** $\xi_{right} - \xi_{left} > error$ **do**
10: $\quad\quad\quad\xi_{middle} \leftarrow \xi_{left} + \xi_{right}$;
11: $\quad\quad\quad$Calculate $\boldsymbol{\beta}_{middle}$ by Algorithm 4 with input ξ_{middle};
12: $\quad\quad\quad$**if** sum$(\boldsymbol{\beta}_{middle}) < 1$ **then**
13: $\quad\quad\quad\quad\xi_{left} \leftarrow \xi_{middle}$;
14: $\quad\quad\quad$**else**
15: $\quad\quad\quad\quad\xi_{right} \leftarrow \xi_{middle}$;
16: $\quad\quad\quad$**end if**
17: $\quad\quad$**end while**
18: $\quad\quad\boldsymbol{\beta}_{opt} \leftarrow \boldsymbol{\beta}_{right}$;
19: $\quad$**end if**
20: **end if**
21: $\boldsymbol{\beta}_{\mathcal{S}}^{*} \leftarrow \boldsymbol{\beta}_{opt}$;

Algorithm 4: Finding $\boldsymbol{\beta}_{\mathcal{S}}$ Such That $\partial E_i/\partial\beta_i = \xi(i \in \mathcal{S})$.

Input: The given ξ; $[\beta_{i,l}^1, \beta_{i,r}^1]$, $[\beta_{i,l}^2, \beta_{i,r}^2]$, $[\beta_{i,l}^3, \beta_{i,r}^3]$; $[BL_i^1, BU_i^1]$, $[BL_i^2, BU_i^2]$, $[BL_i^3, BU_i^3]$ for all $i \in \mathcal{S}$;
Output: $\boldsymbol{\beta}_{\mathcal{S}} = \{\beta_i \mid i \in \mathcal{S}|\}$ where $\partial E_i(\beta_i)/\partial\beta_i = \xi(i \in \mathcal{S})$;
1: **for** each $i \in \mathcal{S}$ **do**
2: $\quad$**switch** (ξ)
3: $\quad\quad$**case** $\xi > BU_i^3$**:**
4: $\quad\quad\quad\beta_i \leftarrow \beta_{i,r}^3$;
5: $\quad\quad$**case** $BL_i^3 < \xi \le BU_i^3$**:**
6: $\quad\quad\quad$Set $E_i = Y_i/\beta_i$;
7: $\quad\quad\quad$Calculate β_i by Algorithm 5 with input $(\xi, \beta_{i,l}^3, \beta_{i,r}^3)$;
8: $\quad\quad$**case** $BU_i^2 < \xi \le BL_i^3$**:**
9: $\quad\quad\quad\beta_i \leftarrow \beta_{i,l}^3$;
10: $\quad\quad$**case** $BL_i^2 \le \xi \le BU_i^2$**:**
11: $\quad\quad\quad$Set $E_i = (X_i - \frac{Y_i}{\beta_i})[1 - \frac{\mathcal{T}_i}{D_i}(\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}})^{-1}] + \frac{Y_i}{\beta_i}$;
12: $\quad\quad\quad$Calculate β_i by Algorithm 5 with input $(\xi, \beta_{i,l}^2, \beta_{i,r}^2)$;
13: $\quad\quad$**case** $BU_i^1 < \xi < BL_i^2$ and $\mathcal{T}_i S_i/(D_i C_i) \le 1$**:**
14: $\quad\quad\quad\beta_i \leftarrow \beta_{i,l}^2$;
15: $\quad\quad$**case** $BL_i^1 < \xi < BU_i^1$ and $\mathcal{T}_i S_i/(D_i C_i) \le 1$**:**
16: $\quad\quad\quad$Set $E_i = (X_i - \frac{Y_i}{\beta_i})\frac{\mathcal{T}_i S_i}{D_i C_i} + \frac{Y_i}{\beta_i}$;
17: $\quad\quad\quad$Calculate β_i by Algorithm 5 with input $(\xi, \beta_{i,l}^1, \beta_{i,r}^1)$;
18: $\quad\quad$**case** $\xi < BL_i^1$ and $\mathcal{T}_i S_i/(D_i C_i) \le 1$**:**
19: $\quad\quad\quad\beta_i \leftarrow \beta_{i,l}^1$;
20: $\quad\quad$**default:**
21: $\quad\quad\quad\beta_i \leftarrow 0$;
22: $\quad\quad$**end switch**
23: **end for**

Combining Theorems 3.2 and 3.3, it can be concluded that, to find the optimal $\boldsymbol{\beta}_{\mathcal{S}}$ for problem $(\mathbf{P5})$, the key is to find the optimal ξ such that $\partial E_i/\partial\beta_i = \xi$ for all $i \in \mathcal{S}$ and $\sum_{i\in\mathcal{S}}\beta_i = 1$. The details are introduced as follows.

Calculation Of Search Range: Let $[\beta_{i,l}^1, \beta_{i,r}^1]$, $[\beta_{i,l}^2, \beta_{i,r}^2]$ and $[\beta_{i,l}^3, \beta_{i,r}^3]$ be three sub-intervals of β_i in the ascending order, which can be calculated based on the segmentation condition in (9) or (11), and let $[BL_i^1, BU_i^1]$, $[BL_i^2, BU_i^2]$, and $[BL_i^3, BU_i^3]$ be the corresponding sub-intervals of $\partial E_i/\partial\beta_i$ shown in Fig. 3, and the calculation of $\partial E_i/\partial\beta_i$ is given in the supplementary material. Let $B_{\min} = \min\{BL_1^1, \ldots, BL_{|\mathcal{S}|}^1\}$ and $B_{\max} = \max\{BU_1^3, \ldots, BU_{|\mathcal{S}|}^3\}$, then $[B_{\min}, B_{\max}]$ is the effective search range. The search process is decomposed into three sub-algorithms Algorithms 3, 4 and 5:

Algorithms 4 and 5 are to find $\boldsymbol{\beta}_{\mathcal{S}}$ for a given ξ such that $\partial E_i/\partial\beta_i = \xi$ for all $i \in \mathcal{S}$ as much as possible;

Algorithm 3 is to search ξ during $[B_{\min}, B_{\max}]$ using a binary search method such that $\sum_{i\in\mathcal{S}}\beta_i = 1$ where $\beta_i(i \in \mathcal{S})$ is determined by Algorithm 4.

In Algorithm 3, ξ_{left} and ξ_{right} are initialized as the lower boundary and upper boundary of $\partial E_i/\partial\beta_i$ $(i \in \mathcal{S})$ first. The $\boldsymbol{\beta}$ value is found by Algorithm 4 for ξ_{left} and ξ_{right}, denoted as $\boldsymbol{\beta}_{left}$ and $\boldsymbol{\beta}_{right}$, respectively (line 2). If the sum of $\boldsymbol{\beta}_{right}$ is less than 1, that means the channel resource is enough to achieve the maximal marginal benefit, so $\boldsymbol{\beta}_{right}$ is the optimal $\boldsymbol{\beta}_{opt}$ (lines 3 to 4). If the sum of $\boldsymbol{\beta}_{left}$ is greater than 1, this means no available offloading scheme can be found to satisfy the predefined constraints under the given $\boldsymbol{\lambda}$ (lines 6 to 7). Otherwise, there ought to be a partial derivative value ξ making the sum of $\boldsymbol{\beta}_{\mathcal{S}}$ equal to 1, and the binary search method is adopted to find the right ξ and the corresponding $\boldsymbol{\beta}_{\mathcal{S}}$ is the optimal solution to problem $(\mathbf{P5})$ (lines 9 to 17).

Algorithms 4 and 5 are to find the $\boldsymbol{\beta}_{\mathcal{S}}$ value under a given ξ which satisfies $\partial E_i/\partial\beta_i = \xi$ for all $i \in \mathcal{S}$. Since E_i and $\partial E_i/\partial\beta_i$ are piece-wise functions of β_i, the search range of β_i should be determined first in Algorithm 4, by doing so we can determine the right E_i function. After function E_i is determined, Algorithm 5 is called to find the β_i such that $\partial E_i/\partial\beta_i = \xi$ (lines 7, 12 and 17). The method adopted in Algorithm 5 is also a binary search method since $\partial E_i/\partial\beta_i$ is a monotonic function in each sub-interval of β_i.

c) Update of $\boldsymbol{\gamma}_{\mathcal{S}}$ under a given $\boldsymbol{\beta}_{\mathcal{S}}$: The optimal value of $\boldsymbol{\beta}_{\mathcal{S}}$ can be determined for a given $\boldsymbol{\gamma}_{\mathcal{S}}$. While the optimal $\boldsymbol{\beta}_{\mathcal{S}}$ depends on the initial value of $\boldsymbol{\gamma}_{\mathcal{S}}$, it is necessary to iteratively improve $\boldsymbol{\gamma}_{\mathcal{S}}$ based on the solved $\boldsymbol{\beta}_{\mathcal{S}}$.

Algorithm 5: Finding β_i Under a Given $E_i(\beta_i)$ Function.

Input: ξ, the search range $[\beta_{left}, \beta_{right}]$;
Output: β_i that makes $\partial E_i/\partial\beta_i = \xi$;
1: **while** $\beta_{right} - \beta_{left} > error$ **do**
2: $\beta_{middle} \leftarrow (\beta_{left} + \beta_{right})/2$;
3: $Der_{middle} \leftarrow \partial E_i(\beta_{middle})/\partial\beta_i$;
4: **if** $Der_{middle} < \xi$ **then**
5: $\beta_{left} \leftarrow \beta_{middle}$;
6: **else**
7: **if** $Der_{middle} > \xi$ **then**
8: $\beta_{right} \leftarrow \beta_{middle}$;
9: **else**
10: $\beta_i \leftarrow \beta_{middle}$;
11: **end if**
12: **end if**
13: **end while**
14: $\beta_i \leftarrow (\beta_{left} + \beta_{right})/2$;

Under a given $\boldsymbol{\beta}_{\mathcal{S}}$, the optimal $\boldsymbol{\gamma}_{\mathcal{S}}$ can also be determined using a similar method of finding optimal $\boldsymbol{\beta}_{\mathcal{S}}$ under a given $\boldsymbol{\gamma}_{\mathcal{S}}$. The detailed analyses are given as follows.

Case 1: If $\beta_i \le Y_i/X_i$ and $\mathcal{T}_i S_i/(D_i C_i) \ge 1$, we have $\alpha_i^* = 1$. Hence,

$$\gamma_i = 0. \tag{14}$$

Case 2: If $\beta_i \le Y_i/X_i$ and $\mathcal{T}_i S_i/(D_i C_i) < 1$, we have $\alpha_i = \mathcal{T}_i S_i/(D_i C_i)$ and E_i is not dependent on γ_i. Hence, the value of γ_i is determined in order to satisfy constraint (4b), and it is obtained by calculating

$$1 - \frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}}\right)^{-1} = \frac{\mathcal{T}_i S_i}{D_i C_i}. \tag{15}$$

Case 3: If $\beta_i > Y_i/X_i$ and

$$\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}} \le \frac{\mathcal{T}_i}{D_i},$$

it has $\alpha_i^* = 0$ and $E_i = Y_i/\beta_i$ which is independent of γ_i. Hence, γ_i is set as small as possible in the case, that is

$$\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}} = \frac{\mathcal{T}_i}{D_i}. \tag{16}$$

Case 4: If $\beta_i > Y_i/X_i$ and

$$\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}} > \frac{\mathcal{T}_i}{D_i},$$

it has

$$\alpha_i^* = 1 - \frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}}\right)^{-1},$$

and

$$E_i = \left(X_i - \frac{Y_i}{\beta_i}\right)\left[1 - \frac{\mathcal{T}_i}{D_i}\left(\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}}\right)^{-1}\right] + \frac{Y_i}{\beta_i},$$

which is a function of γ_i, and it is easy to derive the formulas $\partial E_i/\partial\gamma_i < 0$ and $\partial^2 E_i/\partial\gamma_i^2 > 0$.

Algorithm 6: Updating $\boldsymbol{\gamma}_{\mathcal{S}}$ Based on a Fixed $\boldsymbol{\beta}_{\mathcal{S}}$.

Input: The given $\boldsymbol{\beta}_{\mathcal{S}}$ and $\boldsymbol{\gamma}_{\mathcal{S}}$;
Output: $\boldsymbol{\gamma}_{\mathcal{S}}^*$ after update;
1: $\boldsymbol{\gamma}_{\mathcal{S}}^* \leftarrow \mathbf{0}$;
2: Define a task set $\mathcal{Q}$ and $\mathcal{Q} \leftarrow \varnothing$;
3: **for** $i \in \mathcal{S}$ **do**
4: **if** β_i and γ_i satisfy case 1, case 2, or case 3 **then**
5: Calculate γ_i^* by solving (14), (15) or (16).
6: **else**
7: Insert task index into $\mathcal{Q}$;
8: **end if**
9: **end for**
10: $\gamma_{rest} \leftarrow 1 - \sum_{i\in\mathcal{S}} \gamma_i^*$;
11: Set $E_i(\gamma_i)$ as $(X_i - \frac{Y_i}{\beta_i})[1 - \frac{\mathcal{T}_i}{D_i}(\frac{1}{\beta_i S_{tra}^i} + \frac{C_i}{\gamma_i S_{MEC}})^{-1}] + \frac{Y_i}{\beta_i}$;
12: Calculate the lower boundary ξ_{left} and upper boundary ξ_{right} of $\partial E_i/\partial\gamma_i$ $(i \in \mathcal{Q})$;
13: Calculate $\boldsymbol{\gamma}_{left}$ and $\boldsymbol{\gamma}_{right}$ for $\mathcal{Q}$ by a binary search algorithm with inputs ξ_{left} and ξ_{right}, respectively (same with Algorithm 5);
14: **if** $\text{sum}(\boldsymbol{\gamma}_{right}) < \gamma_{rest}$ **then**
15: $\boldsymbol{\gamma}_{\mathcal{Q}}^* \leftarrow \boldsymbol{\gamma}_{right}$;
16: **else**
17: **if** $\text{sum}(\boldsymbol{\gamma}_{left}) > \gamma_{rest}$ **then**
18: $\boldsymbol{\gamma}_{\mathcal{Q}}^* \leftarrow \varnothing$;
19: **else**
20: **while** $\xi_{right} - \xi_{left} > error$ **do**
21: $\xi_{middle} \leftarrow \xi_{left} + \xi_{right}$;
22: Calculate $\boldsymbol{\gamma}_{middle}$ for $\mathcal{Q}$ by a binary search algorithm with input ξ_{middle} (same with Algorithm 5);
23: **if** $\text{sum}(\boldsymbol{\gamma}_{middle}) < \gamma_{rest}$ **then**
24: $\xi_{left} \leftarrow \xi_{middle}$;
25: **else**
26: $\xi_{right} \leftarrow \xi_{middle}$;
27: **end if**
28: **end while**
29: $\boldsymbol{\gamma}_{\mathcal{Q}}^* \leftarrow \boldsymbol{\gamma}_{right}$;
30: **end if**
31: **end if**

In this case, the optimal γ_i can be found using the binary search method which is similar to Algorithm 5. The overall algorithm to update $\boldsymbol{\gamma}_{\mathcal{S}}$ is given in Algorithm 6. In the algorithm, we first calculate the new γ_i value directly for those tasks whose β_i and γ_i satisfy case 1, case 2, or case 3 (lines 3 to 9). Then, the rest tasks share the remaining computing resource γ_{rest} which is calculated in line 10. To make the remaining resource achieve the optimal marginal benefit, a similar method is adopted with Algorithm 3 to find the optimal $\boldsymbol{\gamma}_{\mathcal{Q}}$ such that the sum of $\boldsymbol{\gamma}_{\mathcal{Q}}$ is equal to γ_{rest}.

C. Summary of Solution Process

The heuristic algorithm for solving the prime energy optimization problem is complex since the prime problem is divided

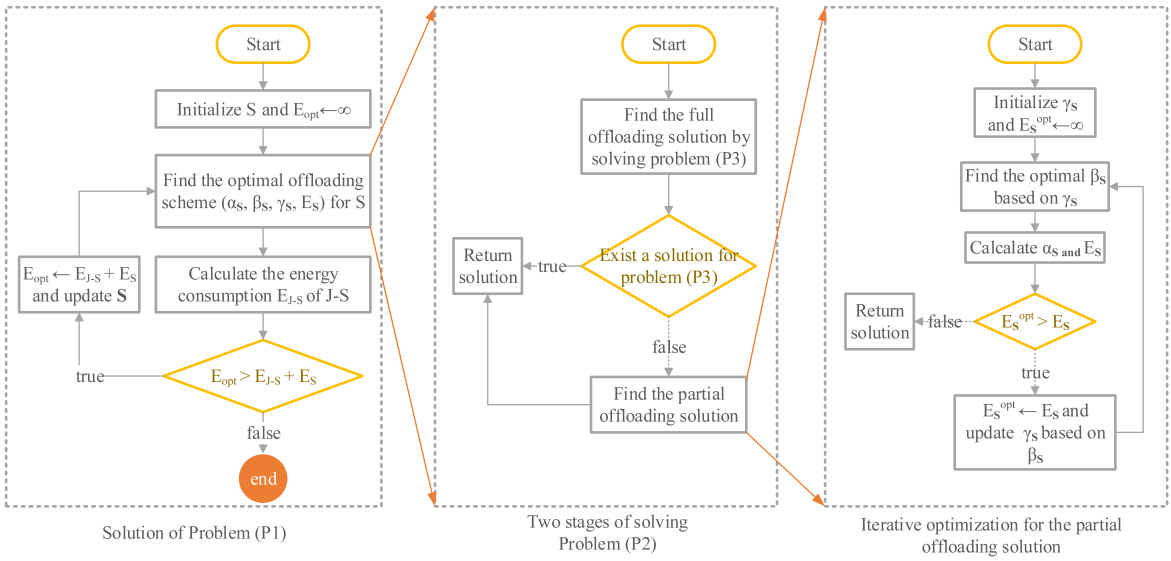


Fig. 4. Algorithm flow chart.

into multiple sub-problems, and the sub-algorithms for solving the sub-problems are nested layer by layer. To enhance the comprehensibility of the algorithm, we provide a flow chart of the algorithm as Fig. 4.

IV. Performance Evaluation

In this section, a series of comparison results are presented to evaluate the performance of the proposed algorithms.

A. Parameter Setting

The configuration of the simulation environment is listed as follows:

- Operating System: 64-bit, Windows 10
- CPU: Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz
- RAM: 8.00 GB
- Programming platform: R2016 Matlab.

In the simulations, the computing capacity of the MEC server is set from 5 to 30 GHz which is a relatively reasonable range, since there is a lot of technical support, for example, multi-core/multi-CPU technology, cluster technology, etc. The bandwidth is set between 10 and 30 MHz by referring to the parameters of 5G network. The computation speed of MDs is randomly generated during the range of [0.7, 1.1], which is the average of the computing capacity of general IoT chips and mobile intelligent terminals. The values of κ_i and λ_i are set as 10^{-27} and 3, respectively [3], [9]. The static power consumption P_s^i is negligible compared with the dynamic power consumption, so we set it to a very small value between 0.02 and 0.05. The task size D_i is randomly selected during [100, 500] KB, and the computation density C_i is randomly set during [500, 1000] cycles/bit. ω_i is randomly selected during the range of [1.5,2.5] Watts^{-1} [10], [12]. The transmitting power of MDs is set during the range of $[20, 29]$ dbm [16]. The parameters are summarized in Table I.

TABLE I
Parameter Setting

parameter	value	unit
S_{MEC}	$5 \sim 30$	GHz
B	$10 \sim 30$	MHz
n	$9 \sim 24$	-
S_i	$0.7 \sim 1.1$	GHz
κ_i	10^{-27}	-
λ_i	3	-
P_s^i	$0.02 \sim 0.05$	Watts
D_i	$100 \sim 500$	KB
C_i	$500 \sim 1000$	cycles/bit
$\mathcal{T}_i$	$1 \sim 4$	Seconds
ω_i	$1.5 \sim 2.5$	Watts^{-1}
P_{tra}^i	$20 \sim 29$	dbm

B. Task Ordering Schemes

As introduced in Section III, the to-offload task set $\mathcal{S}$ is updated according to the predefined priorities. To verify the superiority of our method, we compare the algorithm performance when adopting seven different task ordering schemes. Let L be the order of the tasks. In this paper, we consider the following schemes for the order of $L = \{i_1, i_2, \ldots, i_n\}$.

- Original Order (ORG) – Tasks are arranged in the original order.
- Smallest Data First (SDF) – Tasks are arranged such that $d_{i_1} \leq d_{i_2} \leq \cdots \leq d_{i_n}$.
- Largest Data First (LDF) – Tasks are arranged such that $d_{i_1} \geq d_{i_2} \geq \cdots \geq d_{i_n}$.
- Smallest Computation Requirement First (SCRF) – Tasks are arranged such that $d_{i_1}c_{i_1} \leq d_{i_2}c_{i_2} \leq \cdots \leq d_{i_n}c_{i_n}$.
- Largest Computation Requirement First (LCRF) – Tasks are arranged such that $d_{i_1}c_{i_1} \geq r_{i_2}c_{i_2} \geq \cdots \geq r_{i_n}c_{i_n}$.
- Largest Ratio First (LRF) – Tasks are arranged such that $R_{i_1} \geq R_{i_2} \geq \cdots \geq R_{i_n}$ where $R_i = Y_i/X_i$.

TABLE II
EXPERIMENTAL DATA FOR OPTIMAL COMPUTATION OFFLOADING UNDER DIFFERENT MEC SPEEDS

S_{MEC}	ORG		SDF		LDF		SCRF		LCRF		LRF		SRF	
	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}
5	14.8082	8	15.0603	4	14.3434	17	15.2410	3	14.0728	33	15.4115	1	13.9293	55
10	12.2954	3	12.6737	2	11.8889	9	12.8854	1	11.6241	31	12.9294	1	11.4953	64
15	10.5445	7	10.8253	0	10.2913	12	10.9847	0	10.1311	29	10.9885	0	9.97220	59
20	10.2665	4	10.4666	0	9.96160	9	10.4985	0	9.83340	26	10.5080	0	9.66710	66
25	10.2788	2	10.3811	1	10.0010	21	10.4337	1	9.86940	34	10.4636	1	9.80660	55
30	9.97480	5	10.1544	1	9.75450	13	10.2267	1	9.62850	29	10.2405	1	9.51560	65

TABLE III
EXPERIMENTAL DATA FOR OPTIMAL COMPUTATION OFFLOADING UNDER DIFFERENT MD SIZES

Number of MDs	ORG		SDF		LDF		SCRF		LCRF		LRF		SRF	
	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}
9	3.4911	9	3.5206	9	3.4418	20	3.5273	10	3.4160	31	3.5298	9	3.3914	35
12	6.6814	9	6.7884	4	6.5571	21	6.8264	5	6.4767	34	6.8492	4	6.3750	58
15	10.3018	3	10.4086	2	9.9903	13	10.5302	1	9.8837	30	10.5241	0	9.7722	58
18	14.1299	7	14.3764	2	13.7617	10	14.4804	1	13.5851	27	14.4756	1	13.435	63
21	17.8822	2	18.1963	0	17.4915	13	18.3079	0	17.3071	27	18.3477	1	17.1475	60
24	22.0991	3	22.4384	0	21.697	15	22.5895	0	21.436	33	22.5432	0	21.3221	56

TABLE IV
EXPERIMENTAL DATA FOR OPTIMAL COMPUTATION OFFLOADING UNDER DIFFERENT DELAY REQUIREMENTS

Delay Requirement	ORG		SDF		LDF		SCRF		LCRF		LRF		SRF	
	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}	E_{opt}	C_{opt}
$1.0 \sim 2.5$	12.1007	10	12.3103	5	11.8524	29	12.3660	9	11.7797	38	12.3764	10	11.7263	45
$1.5 \sim 3.0$	10.0154	5	10.1306	1	9.6999	15	10.2344	1	9.5786	28	10.2700	0	9.4460	61
$2.0 \sim 3.5$	9.4536	5	9.5729	2	9.1840	13	9.6426	1	9.0722	23	9.6621	1	8.8949	64
$2.5 \sim 4.0$	8.9961	2	9.1181	1	8.7463	17	9.1649	1	8.6758	30	9.1666	1	8.5472	59

- Smallest Ratio First (SRF) – Tasks are arranged such that $R_{i_1} \leq R_{i_2} \leq \cdots \leq R_{i_n}$ where $R_i = Y_i/X_i$.

In above task ordering schemes, SRF is the scheme we designed in this paper.

C. Impact of Different Task Ordering Schemes

In Tables II–IV, the optimal total energy consumption is listed concerning different MEC speeds, different MD sizes, and different task delay requirements when adopting diverse ordering schemes introduced above.

- In Table II, the bandwidth is $B = 20$, the number of MDs is $n = 15$, the task delay requirement is randomly generated in [1.5, 3], and the MEC speed S_{MEC} is set from 5 to 30 in step 5.
- In Table III, the bandwidth is $B = 20$, the MEC speed is $S_{MEC} = 20$, the task delay requirement is randomly generated in [1.5, 3], and the number of MDs is set as 9 to 24 in step 3.
- In Table IV, the number of MDs is $n = 15$, the MEC speed is $S_{MEC} = 20$, the bandwidth is $B = 20$, and the task delay requirement is randomly generated in [1,2.5], [1.5, 3], [2,3.5], and [2.5,4], respectively.

To avoid errors caused by randomness, we generate 100 sets of simulation data for each case. For each set of simulation data, our algorithm is applied using seven different task ordering methods: ORG, SDF, LDF, SCRF, LCRF, LRF and SRF. The average energy consumption of the 100 results is calculated for each method, denoted as E_{opt} in the tables. Additionally, we present the number of times that the algorithm performs best under each task ordering method, denoted as C_{opt}.

We have the following observations from the experimental results.

- LCRF performs better than SCRF, and SFR performs better than LRF. Besides, LDF mainly performs better than SDF, while their performance might be affected by the delay requirement.
- SCRF, SDF and LRF perform even worse than ORG.
- On the whole, SRF performs best among the seven methods, LCRF and LDF follow.

In addition, we can conclude the changing trend of energy consumption with different parameters from the tables. In Table II, the total energy consumption of MDs shows an obvious downward trend when the computing capacity of MEC increases from 5 to 15, and then keeps stable during 20 and 30. The reasons are given as follows. When the MEC computing capacity is small, increasing the computing capacity of MEC can reduce the execution latency of tasks effectively, so more tasks can be offloaded to MEC and energy consumption is reduced correspondingly. Whereas, when the computing capacity increases further, its affection on the execution delay becomes smaller and channel resource becomes the key that restricts the performance improvement. From Table III, it is easy to see that the energy consumption of MDs increases when the number of MDs in the

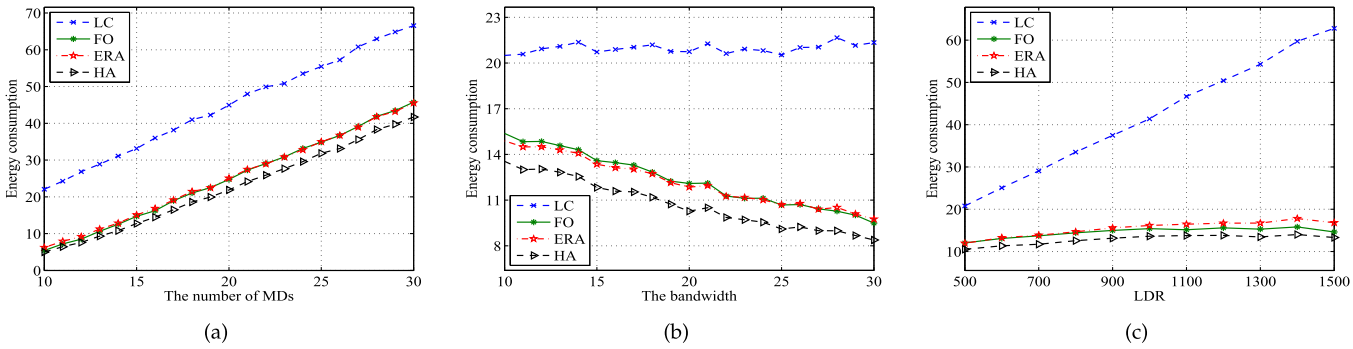


Fig. 5. Energy consumption vesus different (a) number of MDs; (b) bandwidth; (c) LDR.

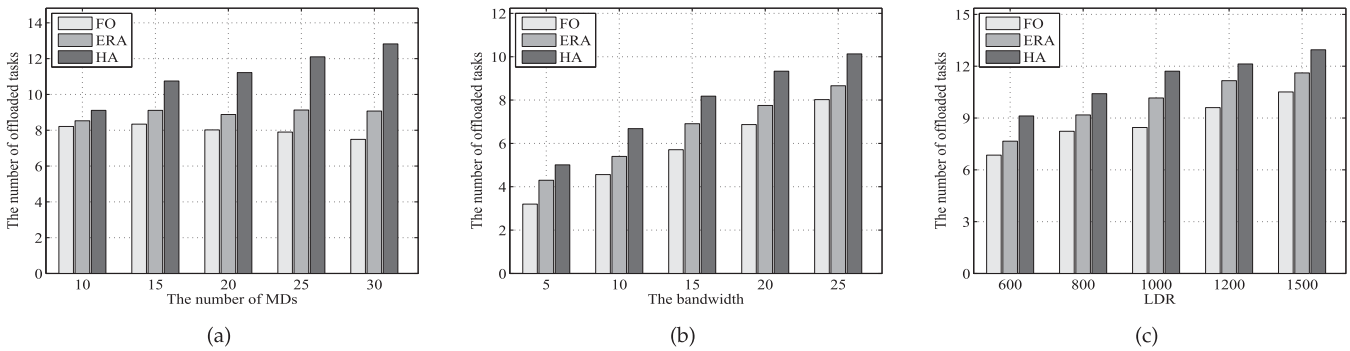


Fig. 6. Number of offloaded tasks vesus different (a) number of MDs; (b) bandwidth; (c) LDR.

coverage of an MEC server increases. The results are apparent since more tasks are generated when the number of MDs increases. Table IV shows that when the delay requirements of tasks get looser, more energy consumption can be saved. That is explained as follows. Fewer resources are required per MD to guarantee the looser delay requirement. For the MEC server with the same computing capacity, it can process much more offloaded tasks on the premise of ensuring performance requirements. In addition, offloading tasks for remote execution consumes less energy than executing them locally. Synthesizing the above reasons, the energy consumption declines when the delay requirements of tasks get looser. Analyzing the performance indicator C_{opt}, we also can find that our method consistently outperforms others in the majority of the 100 repeated experiments.

D. Comparison of Different Strategies

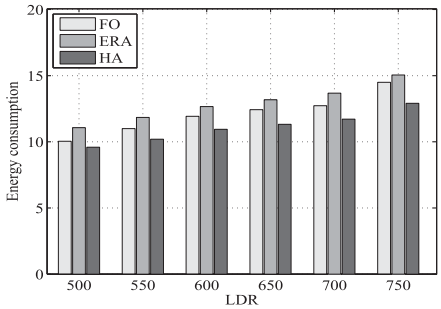
In this section, we compare our Heuristic Algorithm (HA) with three reference schemes,[1] named as

- *Local computation scheme (LC)*, where all MDs process their tasks locally.
- *Full offloading scheme (FO)*, where the tasks are either processed locally or offloaded to MEC. The offloaded tasks are determined by the proposed method in this paper.
- *Equal resource allocation scheme (ERA)*, where wireless channel resource and MEC computing resource are equivalently assigned to its accessed MDs.

To adapt the LC scheme, the delay requirements of tasks should be satisfied when they are processed locally. Hence, the generated tasks should satisfy $D_i C_i / S_i < \mathcal{T}_i$. The other parameters are set Table I. Three groups of simulations are conducted to compare the performance of the four schemes under different numbers of MDs, different bandwidths, and different LDRs, respectively. Similarly, to avoid errors caused by randomness, we generate 100 sets of simulation data for each case.

In Figs. 5 and 6, the energy consumption and the number of offloaded tasks of the four schemes are compared for different parameters, respectively. Fig. 5(a) shows that the energy consumption is increasing with the increase in the number of tasks for all algorithms. Among them, LC consumes much more energy than the other three algorithms, which proves that task offloading can reduce energy consumption effectively. Moreover, our proposed algorithm outperforms both of FO and ERA in terms of energy saving, and it can reduce up to 15% of energy consumption compared with them. Fig. 6(a) shows that the number of offloaded tasks does not change obviously with the increasing number of tasks. That is because the communication and MEC computation resources are fixed, which determines the offloading capacity. Fig. 5(b) depicts the changes of energy consumption under different bandwidths. It is obvious that the energy consumption of LC is not affected by the change in bandwidth, while the energy consumption of FO, ERA, and

[1] Since there is no literature which investigated the same problem in the similar scenario, we propose three reference schemes for comparison herein.



LDR	500	600	700	800	900	1000
Failure Ratio of FO(%)	4	16	32	56	81	95

Fig. 7. Performance comparison under different LDR.

the proposed algorithm declines when the bandwidth becomes greater. The reason can be given by analyzing the results of Fig. 6(b) which shows that more tasks can be offloaded with the increase of bandwidth. In Fig. 5(c), the energy consumption under different LDRs is compared for the four algorithms. From the figure we can see that the energy consumption of LC increases greatly with the increase of LDR, that is because a higher LDR leads to a heavier computation requirement. In contrast, the energy consumption of FO, ERA, and the proposed algorithm does not show obvious changes. The reason is analyzed as follows. Under a greater LDR value, it takes more energy to process a task of the same size locally, while the energy consumption of offloading the task to MEC remains the same. Hence, to achieve optimal energy consumption when LDR rises, offloading more tasks to MEC is inevitable, which can slow down the growth of energy consumption effectively. Fig. 6(c) verifies the change in the number of offloaded tasks.

In the above comparison, the generated tasks are forced to satisfy the condition of $D_iC_i/S_i < \mathcal{T}_i$ to make them executable locally. However, in practical applications, there are many computation-intensive delay-sensitive tasks that cannot be processed by MDs. To verify the advantage of our proposed algorithm in dealing with such tasks, we give another group of simulation results in Fig. 7 which presents the energy consumption and successful ratio of different computation offloading strategies with different LDR.

The figure shows that the energy consumption of the three schemes is increasing with the increase of LDR. Among the three schemes, our proposed algorithm performs best in terms of energy saving, followed by FO, and finally ERA. With the increase of LDR, our algorithm can save more energy compared with FO and ERA, and the energy saving is up to 10.95% and 14.20% when LDR is 750. In addition, although FO performs better than ERA in terms fo energy saving, it has an obvious flaw it cannot find an offloading solution sometimes and the probability of failure becomes greater with the increase of LDR. The reasons are explained as follows. When LDR becomes greater, the ratio of *forced-offloading tasks* gets higher. Since FO is to offload tasks fully to MEC, it is more likely to find no solution when adopting the FO scheme. The table lists the number of times that FO fails to find an offloading solution. From the table, we can see that the failure ratio of FO increases to 95% when the LDR value increases to 1000.

V. Conclusion

We have considered a computation partial offloading problem for a multi-MD MEC system based on TDMA. An energy optimization problem with delay constraint is formulated where the resource competition for wireless channel and MEC computing resources are considered. Due to the complexity of the primal optimization problem, a heuristic algorithm is proposed to solve the problem by dividing it into a series of offloading sub-problems. The offloading sub-problem is to find the optimal offloading scheme for a fixed offloaded task set which is updated iteratively based on the proposed discipline. We first try to find the full offloading solution for the sub-problem, which is proved to be a convex problem. Then, an iterative binary searching is proposed to find the optimal partial offloading solution in which the resource allocation concerning channel resources and MEC computing frequency are optimized cyclically. The experimental results show that the proposed algorithm outperforms the comparing schemes in terms of energy savings under strict delay requirements.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

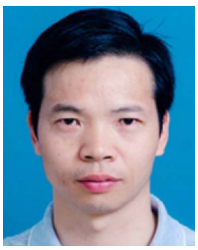
References

[1] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2443–2459, Apr. 2020.
[2] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.
[3] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.
[4] H. Yu, Q. Wang, and S. Guo, "Energy-efficient task offloading and resource scheduling for mobile edge computing," in *Proc. IEEE Int. Conf. Netw. Architecture Storage*, 2018, pp. 1–4.
[5] A. Miettinen and J. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 22–25.
[6] G. Xie, G. Zeng, R. Li, and K. Li, *Scheduling Parallel Applications on Heterogeneous Distributed Systems*. Tokyo, Japan: Singapore Pte Ltd., 2019.
[7] 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.
[8] Z. Kuang, Z. Ma, Z. Li, and X. Deng, "Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing," *J. Syst. Archit.*, vol. 118, no. 99, 2021, Art. no. 102167.
[9] V. Joshi and K. Patil, "Delay-energy aware task offloading and vm migration policy for mobile edge computing," *Wireless Pers. Commun. Int. J.*, vol. 4, 2022, Art. no. 123.
[10] K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.
[11] C. Singhal and S. De, *Resource Allocation in Next-Generation Broadband Wireless Access Networks*. Hershey, US: IGI Global, 2017.
[12] K. Li, "Heuristic computation offloading algorithms for mobile users in fog computing," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 2, pp. 1–28, 2021.

[13] S. Li, S. Lin, L. Cai, W. Li, and G. Zhu, "Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3384–3398, Mar. 2020.
[14] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
[15] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," 2014. [Online]. Available: http://cvxr.com/cvx
[16] M. Electronics, "Mouser-sub-ghz modules," *Website*, 2023. [Online]. Available: https://www.mouser.com/c/embedded-solutions/wireless-rf-modules/sub-ghz-modules/

Jing Mei received the PhD degree in computer science from Hunan University, China, in 2015. She is currently an associate professor in the College of Information Science and Engineering in Hunan Normal University. Her research interests include cloud computing, fog computing and mobile edge computing, high performance computing, task scheduling and resource management, etc. She has published more than 30 research articles in international conference and journals, such as *IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed System, IEEE Transactions on Service Computing, Cluster Computing, Journal of Grid Computing, Journal of Supercomputing.*

Zhao Tong (Member, IEEE) received the PhD degree in computer science from Hunan University, Changsha, China, in 2014. He was a visiting scholar with the Georgia State University from 2017 to 2018. He is currently an associate professor with the College of Information Science and Engineering of Hunan Normal University, the young backbone teacher of Hunan Province, China. His research interests include parallel and distributed computing systems, resource management, Big Data and machine learning algorithm. He has published more than 25 research papers in international conferences and journals, such as *IEEE Transactions on Parallel and Distributed Systems*, *Information Sciences*, FGCS, NCA, and JPDC, PDCAT, etc. He is a senior member of the China Computer Federation (CCF).



Kenli Li (Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar with the University of Illinois, Urbana Champaign from 2004 to 2005. He is currently a full professor of computer science and technology with Hunan University and deputy director of National Supercomputing Center, in Changsha. His major research includes parallel computing, cloud computing, and DNA computing. He has published more than 110 papers in international conferences and journals, such as *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Signal Processing*, JPDC, ICPP, CCGrid, FGCS. He is currently or has served on the editorial boards of *IEEE Transactions on Computers*, *International Journal of Pattern Recognition and Artificial Intelligence*, He is an outstanding member of CCF.

Lianming Zhang (Member, IEEE) received the BSc and MSc degrees from the Department of Physics of Hunan Normal University, and the PhD degree from the School of Information Science and Engineering of Central South University, China. He is currently a professor in the College of Information Science and Engineering of Hunan Normal University, China, and leads the Key Laboratory of Internet of Things Technology and Application there. His main research interests include network intelligence, software-defined networking, edge computing, and complex network. He completed a two-year postdoctoral fellowship in complex networks at the School of Computer Science and Engineering of the South China University of Technology. He manages research projects funded by various sources such as the National Natural Science Foundation of China and other companies. He has published more than 120 papers.

Keqin Li (Fellow, IEEE) received the BS degree in computer science from Tsinghua University, in 1985 and the PhD degree in computer science from the University of Houston, in 1990. He is currently a SUNY distinguished professor with the State University of New York and a National Distinguished Professor with Hunan University (China). He has authored or co-authored more than 950 journal articles, book chapters, and refereed conference papers. He received several best paper awards from international conferences including PDPTA-1996, NAECON-1997, IPDPS-2000, ISPA-2016, NPC-2019, ISPA-2019, and CPSCom-2022. He holds nearly 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top five most influential scientists in parallel and distributed computing in terms of single-year and career-long impacts based on a composite indicator of the Scopus citation database. He was a 2017 recipient of the Albert Nelson Marquis Lifetime Achievement Award for being listed in Marquis Who's Who in Science and Engineering, Who's Who in America, Who's Who in the World, and Who's Who in American Education for more than twenty consecutive years. He received the Distinguished Alumnus Award from the Computer Science Department at the University of Houston, in 2018. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing, in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing, in 2023. He is a member of the SUNY Distinguished Academy. He is an AAAS Fellow, and an AAIA Fellow. He is a member of academia europaea (Academician of the Academy of Europe)