

# High Efficiency Inference Accelerating Algorithm for NOMA-Based Edge Intelligence

Xin Yuan, Ning Li<sup>ID</sup>, *Member, IEEE*, Tuo Zhang, Muqing Li, Yuwen Chen<sup>ID</sup>, *Member, IEEE*,  
Jose Fernan Martinez Ortega<sup>ID</sup>, and Song Guo<sup>ID</sup>, *Fellow, IEEE*

**Abstract**—Even the artificial intelligence (AI) has been widely used and significantly changed our life, deploying the large AI models on resource limited edge devices directly is not appropriate. Thus, the model split inference is proposed to improve the performance of edge intelligence (EI), in which the AI model is divided into different sub-models and the resource-intensive sub-model is offloaded to edge server wirelessly for reducing resource requirements and inference latency. Unfortunately, with the sharp increasing of edge devices, the shortage of spectrum resource in edge network becomes seriously in recent years, which limits the performance improvement of EI. Refer to the NOMA-based edge computing (EC), integrating non-orthogonal multiple access (NOMA) technology with split inference in EI is attractive. However, the NOMA-based communication aspect and the influence of intermediate data transmission fail to be considered properly in model split inference of EI in previous works, and the sophistication in resource allocation caused by NOMA scheme makes it further complicated. Thus, the Effective Communication and Computing resource allocation algorithm is proposed in this paper for accelerating the split inference in NOMA-based EI, shorted as ECC. Specifically, the ECC takes the energy consumption and the inference latency into account to find the optimal model split strategy and resource allocation strategy (subchannel, transmission power, computing resource). Since the minimum inference delay and energy consumption cannot be satisfied simultaneously, the gradient descent (GD) based algorithm is adopted to find the optimal tradeoff between them. Moreover, the loop iteration GD approach (Li-GD) is developed to reduce the complexity of the GD algorithm caused by parameter

discretization. The key idea of Li-GD is that: the initial value of the  $i$ th layer's GD procedure is selected from the optimal results of the former  $(i - 1)$  layers' GD procedure whose intermediate data size is the closest to  $i$ th layer. Additionally, the properties of the proposed algorithms are investigated, including convergence, complexity, and approximation error. The experimental results demonstrate that the performance of ECC is much better than that of the previous studies.

**Index Terms**—Edge intelligence, model split, inference accelerating, NOMA.

## I. INTRODUCTION

ARTIFICIAL intelligence (AI) has been widely used and has significantly changed our lifestyle, such as metaverse [1], [2], automatic driving [2], [3], [4], image generation [5], etc. However, because the sizes of AI models are always large to provide high-performance services, the quantity of computing resources required for these models is also large. Therefore, it is inappropriate to deploy these AI models on edge devices, such as mobile phones, vehicles, unmanned aerial vehicles (UAVs), etc., in which computing resources are quite limited. To address this issue, one possible solution is to split the large AI model into different sub-models and offload the resource-intensive part to the edge server wirelessly (e.g., WiFi, 5G, etc.) to reduce resource requirement and inference latency [6], [7], [8], [52], [53], [54]. The model split strategies between device, edge server and cloud has been investigated by previous works, and some excellent works have been proposed, such as [9], [10], [11], [12], [13], [14], [15], and [16]. These studies find the optimal model segmentation points or early-existing points to minimize inference delay and resource requirements while maintaining high inference accuracy by reinforcement learning [9], convex optimization [10], [11], [12], [13], heuristic algorithm [14], [15], [16], etc.

However, since the number of edge devices has increased sharply in recent years, spectrum resources in edge networks have become increasingly scarce [32]. This seriously limits the performance improvement of the EI. To facilitate massive connectivity over wireless networks and improve the spectrum utilization and data transmission rate, the non-orthogonal multiple access (NOMA) technique has been proposed as an alternative to traditional OMA [17], [18], [19], [20], [21], [22]. The NOMA technology has been integrated into multi-access mobile edge computing (MEC) to improve the its performance [23], [24], [25], [26], [27], [28], [29], [30], [31], [32]. In NOMA-based MEC systems, the task offloading,

Received 14 December 2023; revised 23 June 2024; accepted 24 August 2024. Date of publication 9 September 2024; date of current version 13 November 2024. This work was supported in part by NSFC under Grant 62101159, in part by NSF of Shandong under Grant ZR2021MF055, and in part by the Research Grants Council of Hong Kong through the Areas of Excellence Scheme under Grant AoE/E-601/22-R. The associate editor coordinating the review of this article and approving it for publication was F. Hou. (*Corresponding author: Ning Li.*)

Xin Yuan is with the School of Ocean Engineering, Harbin Institute of Technology, Harbin 150001, China (e-mail: xin.yuan@hit.edu.cn).

Ning Li, Tuo Zhang, and Muqing Li are with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: li.ning@upm.es; 23S130405@stu.hit.edu.cn; 23S130418@stu.hit.edu.cn).

Yuwen Chen is with the College of Computer Science, Beijing University of Technology and Science, Beijing 100124, China (e-mail: yuwen.chen@bjut.edu.cn).

Jose Fernan Martinez Ortega is with the Departamento de Ingeniería y Arquitecturas Telemáticas, Universidad Politécnica de Madrid, 28040 Madrid, Spain (e-mail: jf.martinez@upm.es).

Song Guo is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: songguo@cse.ust.hk).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TWC.2024.3454086>.

Digital Object Identifier 10.1109/TWC.2024.3454086

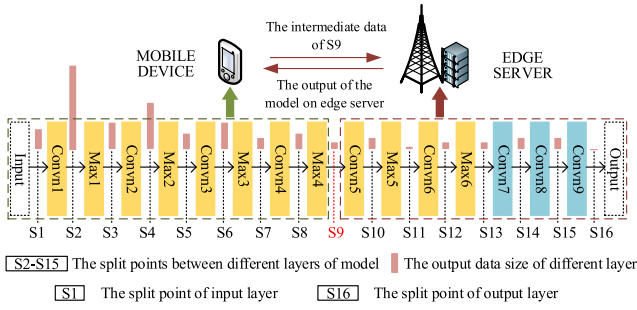


Fig. 1. An example of YOLOv2.

the channel allocation, and the transmission power allocation strategies are jointly optimized to achieve high spectrum efficiency, high transmission rate, and low transmission delay.

Motivated by the benefits of NOMA-based MEC, integrating NOMA into model split inference in EI to reduce inference latency will be attractive. However, the NOMA-based communication aspect has not been properly considered in model split inference of EI in previous works, and the sophistication in communication resource allocation caused by the NOMA scheme makes it further complicated. *First*, although the task offloading in NOMA-based MEC has been investigated intensively, these approaches cannot be applied directly to model split inference in NOMA-based EI. This is because the model split inference in NOMA-based EI is much more fine-grained than task offloading in MEC [9], [10], [11], [12], [13], [14], [15], [16]. In model split inference, the sub-models on end devices and edge servers need to exchange model parameters (e.g., weights and activation values) frequently during inference. However, the effect of intermediate data transmission between edge devices and edge servers on task calculation delay and energy consumption is ignored in NOMA-based MEC. This means that for the task offloading in NOMA-based MEC, the more tasks that are calculated on devices, the higher the calculation delay and energy consumption, and vice versa. However, this is inapplicable to the model split inference in NOMA-based EI, in which the output data sizes of different layers are quite uneven and have a significant effect on the model split strategy. For instance, as shown in Fig.1, the output data size of layer “Conv1” in YOLOv2 is about 50 times larger than that of layer “Max5”. Therefore, considering the intermediate data transmission delay, the inference delay when the model split point is  $s_9$  may be lower than that when it is  $s_1$ . Therefore, the model split strategies seriously affect the inference delay, especially when NOMA technology is applied, in which the transmission power of the device and edge server can be adjusted to achieve different transmission rates [23]. Moreover, due to the uneven and discrete nature of the intermediate data size, calculating the optimal model split strategy is considerably more difficult. *Second*, for mobile devices, one of the most important factors to consider is the energy consumption, because it is always powered by a battery that is limited to the energy store and determines the lifetime of the devices. In most MEC scenarios, the lifetime of mobile devices is more important than inference delay and accuracy to some extent, as a long lifetime is the foundation of high quality

of services (QoS) [33], [34], [35]. Once the mobile devices are out of energy, the QoS is meaningless. Therefore, the energy consumption of mobile devices is critical for achieving high-performance EI. Except for the end devices, the edge servers running 24/7 consume a large amount of energy and contribute a significant proportion of global carbon emissions, thus, the energy consumption on the edge server should also be reduced to achieve a green data center [42], [44]. Even energy consumption has been considered in model split inference in previous works [31], considering the challenges that arise by integrating NOMA into model split inference in EI, the transmission power adjustment, the model split point selection, and the energy consumption are interactional and tightly coupled, which means that the algorithms proposed in previous studies cannot work effectively in NOMA-based EI.

However, finding the optimal model split strategy and resource allocation (transmission power, computing resource, etc.) strategy to achieve the minimum inference latency and energy consumption for NOMA-based EI is challenging. This is because the optimal objectives of these two parameters are opposite. For instance, when minimizing the energy consumption of the edge server, the size of the model offloaded to the edge server should be as small as possible. However, this means most parts of the model should be calculated on the device, which causes a high inference delay and energy consumption of the end device. Therefore, determining the optimal tradeoff between these two parameters and optimizing the performance of the entire system effectively is difficult.

Based on the above issues, in this paper, an Effective Communication and Computing resource allocation algorithm is proposed for accelerating split inference in NOMA-based EI, which is abbreviated as ECC. Specifically, when the end device has a model inference task that needs to be calculated in the NOMA-based EI, it takes the energy consumption of both device and edge server and the inference latency into account to find the optimal model split strategy, subchannel allocation strategy (uplink and downlink), transmission power allocation strategy (uplink and downlink), and computing resource allocation strategy. Since the minimum inference delay and energy consumption cannot be satisfied simultaneously, and the variables of subchannel allocation and model split are discrete, the gradient descent (GD) algorithm is adopted to determine the optimal tradeoff between them. Moreover, the loop iteration GD approach (Li-GD) is proposed to reduce the complexity of the GD algorithm caused by discrete parameters. The key idea of the Li-GD algorithm is that: the initial value of the  $i$ th layer’s GD procedure is selected from the optimal results of the former  $(i - 1)$  layers’ GD procedure whose intermediate data size is the closest to the  $i$ th layer. Additionally, the properties of the proposed algorithms are investigated, including convergence, complexity, and approximation error.

The main contributions of this paper can be summarized as follows.

- 1) In this study, we integrate NOMA into the model split inference in EI. To the best of our knowledge, this is the first work to investigate the possibility and approach of using NOMA technology to improve

the performance of model split inference in EI under multi-user scenario. Moreover, the challenges and issues caused by integration are also discussed in this paper.

- 2) Since NOMA has a significant effect on model split point selection and energy consumption, we take both the energy consumption and inference delay into account to find the optimal model split strategy and resource allocation strategy (computing resource, channel resource, transmission power) for NOMA-based split inference in EI. Moreover, because the minimum energy consumption and minimum inference latency cannot be achieved simultaneously, the GD-based algorithm is adopted in this study to effectively achieve an optimal tradeoff between them.
- 3) Moreover, considering the complexity of this issue caused by uneven and discrete intermediate data size, we propose a Li-GD algorithm to improve the efficiency of the GD procedure. The key idea of the Li-GD algorithm is that: the initial value of the  $i$ th layer's GD procedure is selected from the optimal results of the former  $(i-1)$  layers' GD procedure whose intermediate data size is the closest to the  $i$ th layer.
- 4) The properties of the proposed Li-GD algorithm are investigated. The Li-GD algorithm is convergent, and the convergence time is  $K = \frac{\|x^0 - x^*\|_2^2}{2\eta\epsilon}$ , the complexity of the Li-GD is  $O(X\bar{K}\mathcal{F}Mx^3\ln^2(x))$ , the approximate error is smaller than  $\frac{\epsilon}{\rho_{min}(1-B_{max})\log_2\left(1+\frac{P_{min}}{\Delta^*+\frac{\alpha P_{max}}{2}}\right)}$ .

Additionally, it can reduce the complexity and convergence time compared with the traditional GD approach.

The remainder of this paper is organized as follows. Section II introduces the related works. The network models and the problems to be solved in this paper are presented in Section III. The Li-GD algorithm is proposed in Section IV, and its properties, e.g., the convergence, the complexity, etc., are also investigated in this section. In Section V, the effectiveness of the proposed ECC algorithm is demonstrated through simulation. Finally, Section VI summarizes the conclusions of this work.

## II. RELATED WORKS

In this section, the details of the most related works of ECC are introduced. These works can be divided into two major categories: the NOMA-based MEC [23], [24], [25], [26], [27], [28], [29], [30], [31], [32] and the model split and resource allocation in EI [9], [10], [11], [12], [13], [14], [15], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47]. Additionally, the disadvantages of these works are also discussed in this section.

### A. NOMA-Based Mobile Edge Computing

The NOMA has been used in uplink or downlink channel during task offloading and results transmission in MEC. Then, multi-objectives, such as the task offloading strategy, and channel allocation strategy, etc., are optimized jointly to achieve high spectrum efficiency, high transmission rate, and low transmission delay of the MEC.

Specifically, the authors in [23] proposed heuristic algorithms for MEC-aware NOMA single cell, in which the user clustering, computation resource and transmission powers allocation were jointly optimized to minimize the energy consumption. In [24], the authors proposed a joint optimization scheme of SIC ordering and computation resource in a MEC-aware NOMA Narrow Band Internet of Things (NB-IoT) network, to minimize the total execution delay for each bit. The authors in [25] considered two kinds of users' scenarios, including the single user and the multiple users, in which the offloading workloads and transmission time were optimized to minimize the delay, while satisfying the computation requirement of each user. In [26], the authors proposed a scheduling-based algorithm for the device-to-device (D2D)-assisted and NOMA-based MEC system, in which D2D was applied to enhance user collaboration and relieve the computing burden on edge server, and the computation resource, power, and channel allocations were jointly optimized to minimize the system cost. The aforementioned studies focus on the single-cell networks. The NOMA-assisted MEC in multi-cell networks has been learned in [27], [28], [29], [30], [31], and [32]. Specifically, the authors in [27] investigated the joint power and computation resource allocation scheme for the NOMA-assisted MEC heterogeneous network (HetNet), to minimize the energy consumption of users. In [28], the authors proposed an energy-efficient joint offloading decision, user scheduling and resource allocation scheme for an integrated fog-cloud network. Moreover, [29] investigate the online user allocation problem in MEC with NOMA. The authors minimize the allocation delay and transmit power costs by Lyapunov to convert the long-term optimization problem into a series of subproblems in every time slot. The authors in [30] propose an online algorithm to solve the dynamic edge user allocation problem in NOMA-based MEC system. The authors in [31] tackle the edge demand response problem in NOMA-based MEC system. A two-phase game-theoretical approach is proposed in [32].

However, the effect of intermediate data transmission on task calculation delay and energy consumption, which is the main motivation of this study, is ignored in the above related works. Therefore, the algorithms that proposed in these works cannot be directly applied to the split inference of NOMA-based EI.

### B. Model Split and Resource Allocation in EI

For the model split and resource allocation, the research efforts focus on offloading computation from the resource-constrained mobile to the powerful cloud to reduce inference time [36], [37]. Neurosurgeon [38] explores a computation offloading method for DNNs between the mobile device and the cloud server at layer granularity. In [39], the authors propose MAUI, which is an offloading framework that can determine to execute the functions of a program on edge or cloud. But the MAUI is not explicitly designed for DNN segmentation since the size of communication data between devices and edge/cloud is small. The DDNN is proposed in [40], which is a distributed deep neural network architecture that is distributed across different computing devices. The

purpose of DDN is to reduce the communication data size among devices for the given DNN. The authors in [9] propose multi-exit DNN inference acceleration framework based on multi-dimensional optimization. This is the first work that investigates the bottlenecks of executing multi-exit DNNs in edge computing and builds a novel model for inference acceleration with exit selection, model partition, and resource allocation. The authors of [11] and [41] propose feature-based slicing method, which divided a layer into multiple slices for parallel inference. The [41] established a mobile devices' cluster and published the slices to appropriate nodes. In [11], the authors propose a technique to divide a DNN in multiple partitions that can be processed locally by end devices or offloaded to one or multiple powerful nodes, such as in fog networks. Unlike slicing feature map, the authors of [43] sliced the input data into multiple tiles for parallelism. However, both feature-based and input-based partition require frequent synchronous data transmission. Other studies adopt layer wise partition. In [15], the authors investigate the optimization problem of DNN partitioning in a realistic multiuser resource-constrained condition that rarely considered in previous works. And they propose iterative alternating optimization (IAO) algorithm to achieve the optimal solution in polynomial time. However, in most cases, model partition degenerates into binary offloading in this approach. Thus, the authors of [45] and [46] compressed the intermediate data to reduce the transmission latency with a certain loss of accuracy. In [47], by modeling and solving a batch task scheduling problem, the feasibility of partitioning on a three-exit DNN is also verified under a three-tier framework, i.e., devices, edge, and cloud. In [10], the authors devise a collaborative edge computing system CoopAI to distribute DNN inference over several edge devices with a novel model partition technique to allow the edge devices to prefetch the required data in advance to compute the inference cooperatively in parallel without exchanging data. In [12], the authors propose JointDNN, for collaborative computation between a mobile device and cloud for DNNs in both inference and training phase. In [13], considering online exit prediction and model execution optimization for multi-exit DNN, the authors propose a dynamic path based DNN synergistic inference acceleration framework (DPDS), in which the exit designators are de-signed to avoid iterative entry for exits. Moreover, the multi-exit DNN is dynamically partitioned according to network environment to achieve fine-grained computing offloading. In [14], the authors design the DNN surgery. The DNN surgery allows partitioned DNN to be processed at both the edge and cloud while limiting the data transmission. Moreover, they design a dynamic adaptive DNN surgery (DADS) scheme to optimally partitions the DNN under different network conditions.

However, the above model split inference and resource allocation algorithms do not consider the effect of NOMA on the inference latency and energy consumption. Therefore, for the multi-user scenario, owing to the shortage of spectrum resources, the performance of model split inference in EI will be affected seriously.

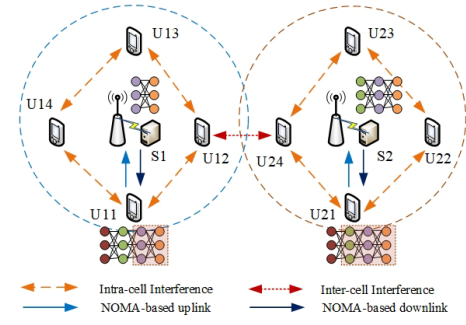


Fig. 2. Network model.

### III. NETWORK MODEL AND PROBLEM STATEMENT

In this section, the split inference, the inference delay and the energy consumption models that used in this paper are introduced. The network model is illustrated in Fig.3. Subsequently, based on the proposed models, the problems that will be solved in this study are described in detail.

As shown in Fig.2, we consider mobile edge computing in a multi-cell network with  $N$  single-antenna APs and  $U$  single-antenna end devices, indexed by  $N = \{1, 2, \dots, N\}$  and  $U = \{1, 2, \dots, U\}$ , respectively. The total system bandwidth  $B$  is equally divided into  $M$  orthogonal subchannels, indexed by  $M = \{1, 2, \dots, M\}$ . We consider the use of the nearest AP association policy [48]. As such, each end device associates with the nearest AP that can provide the maximum average channel gain. We denote the set of devices served by AP  $n$  as  $U_n$ . Therefore, each device  $u \in U$  offloads its split inference model to its associated AP  $n \in N$  via subchannel  $m \in M$ . We assume that  $M < N$  and denote the set of devices served by AP  $n$  on subchannel  $m$  as  $U_n^m$ . Therefore, the devices associated with different APs may access to the same subchannel and interfere with each other. The uplink and downlink between mobile user and edge server are all NOMA-based channels. Thus, not only can the users in the coverage of the same AP interfere with each other, but also with the users who select the same subchannel in adjacent APs. For each mobile user, there is a large model with  $\mathcal{F}$  layers needs to be split and offloaded from device to edge server.

#### A. Split Inference Model

In this section, we take YOLOv2 as an example to demonstrate the model split policy and the effect of different model split strategies on communication overhead. As shown in Fig.1, the model has 16 available split points, which are  $s_1, s_2, \dots, s_{15}, s_{16}$ , respectively. Note that  $s_1$  means the entire model is offloaded to and calculated on the edge server,  $s_{16}$  means that the entire model is calculated on the device. The number of layers of YOLOv2 is larger than that of available split points. This means that not all the layers in the model can be selected as split points. Therefore, for model  $\mathcal{M}$ , its model split policies can be expressed as:  $S^{\mathcal{M}} = s_i^{\mathcal{M}} (\forall i \in \{1, 2, \dots, \mathcal{F}_{\mathcal{M}}\})$ , where  $\mathcal{F}_{\mathcal{M}}$  is the number of available split points in model  $\mathcal{M}$ , in which  $s_1$  means the whole model is offloaded to and calculated on edge server,  $s_{\mathcal{F}_{\mathcal{M}}}$  means the whole model is calculated on device. For instance, in Fig.1,

$\mathcal{F}_M = 16$  and  $s_3$  means the model split point is between layer Max1 and Conv2. Therefore, when the model split policy is  $s_i$ , it means that the first to  $s_i$ -th layers are calculated on mobile device, and the  $(s_i+1)$ -th to  $\mathcal{F}_M$ -th layers are offloaded to the edge server for deep inference. The intermediate data between  $s_i$ -th layer and  $(s_i+1)$ -th is transmitted through wireless channel.

Therefore, the intermediate data size can affect the transmission delay and inference delay seriously. For instance, as shown in Fig. 1, the intermediate data size between Conv1 and Max1 is about 50 times larger than that between Max5 and Conv6. Thus, if the split point is selected as  $s_1$  rather than  $s_{10}$ , the data transmission delay will be serious. However, this does not mean that the smaller intermediate data size, the lower inference delay is. Because the inference delay is also affected by the calculation time of the sub-models on edge server and devices, which is decided by the computing capability of edge server and devices, respectively. The details of the inference delay model and transmission delay model are presented below.

### B. Inference Delay Model

The inference delay includes three parts: 1) the delay that caused by model inference on mobile device, 2) the delay that caused by model inference on edge server, and 3) the delay that caused by intermediate data transmission between the device and edge server.

1) *Inference Delay on Device*: As presented in Section III-A, the number of model layers is  $\mathcal{F}$  and the model split decision is  $s_i$ , which means that the first to  $s_i$ -th layers are calculated on mobile device  $i$ , and the  $(s_i+1)$ -th to  $\mathcal{F}$ -th layers are offloaded to the edge server for deep inference.  $c_i$  is defined as the floating-point operation capability of device  $i$ . Then, the inference latency on mobile devices after model segmentation can be calculated as:

$$T_i^{device} = \sum_{\delta=1}^{s_i} \frac{f_{l_\delta}}{c_i} \quad (1)$$

where  $f_{l_\delta}$  is the computation task of each layer in the main branch, containing convolutional layer  $f_{conv}$ , pooling layer  $f_{pool}$ , and ReLU layer  $f_{relu}$  [9]. Thus,  $f_{l_\delta}$  can be computed as:

$$f_{l_\delta} = m_{\delta 1} f_{conv} + m_{\delta 2} f_{pool} + m_{\delta 3} f_{relu} \quad (2)$$

where  $m_{\delta 1}$ ,  $m_{\delta 2}$ , and  $m_{\delta 3}$  denote the number of convolutional layers, pooling layers, and ReLU layers, respectively, and  $m_{\delta 1} + m_{\delta 2} + m_{\delta 3} = s_i$ .

2) *Inference Delay on Edge Server*: The execution time is not proportional to the amount of allocated computational resources for the inference tasks, such as DNN, under the scenario that the edge server is multicore CPU. As demonstrated in [15], up to 44% error in execution time between theory and experiment. Thus, in this paper, let  $r_i$  represent the number of minimum computational resource units that allocated to user  $i$ ;  $c_{min}$  implies the capability of minimum computational resource unit. Since in a multicore CPU scenario, the execution time is not linear with respect to the amount of allocated computational

resource, a compensation function  $\lambda(r_i)$  is introduced to fit the execution time in the multicore CPU scenario. For the single core scenario, the  $\lambda(r_i)$  is degenerated to  $r_i$ , and for the multicore scenario,  $\lambda(r_i) > r_i$ . The  $\lambda(r_i)$  can be estimated based on the approach that proposed in [15]. Therefore, in this paper, we only assume that  $\lambda(r_i)$  increases with  $r_i$ , but not linearly. To model the nonlinearity in the execution time, the execution time on edge can be expressed as:

$$T_i^{server} = \sum_{\delta=s_i+1}^{\mathcal{F}} \frac{f_{e_\delta}}{\lambda(r_i) c_{min}} \quad (3)$$

where  $f_{e_\delta}$  is the computation task of each layer in the main branch, containing convolutional layer  $f_{conv}$ , pooling layer  $f_{pool}$ , and ReLU layer  $f_{relu}$ . Thus,  $f_{e_\delta}$  can be calculated as:

$$f_{e_\delta} = m_{\delta 4} f_{conv} + m_{\delta 5} f_{pool} + m_{\delta 6} f_{relu} \quad (4)$$

where  $m_{\delta 4}$ ,  $m_{\delta 5}$ , and  $m_{\delta 6}$  denote the number of convolutional layers, pooling layers, and ReLU layers, respectively, and  $m_{\delta 4} + m_{\delta 5} + m_{\delta 6} = \mathcal{F} - s_i$ .

3) *Network Transmission Delay*: There are two different types of network transmission delay: 1) the intermediate output transmission delay from the device to the edge server in the uplink and 2) the final result transmission delay from the edge server to the device in the downlink.

Firstly, for the intermediate data transmission delay, when the model is split at  $s_i$ -th layer, the intermediate data generated by the  $s_i$ -th layer will be transmitted to edge server to complete the inference. As shown in Fig. 1, the end devices served by the same AP  $n$  on the same subchannel  $m$  form a NOMA cluster  $U_n^m$ , where NOMA protocol is adopted to perform the model offloading. Let  $N_n^m$  and  $h_{n,i}^m$  denote the number of devices in  $U_n^m$  and the channel coefficient of the uplink from device  $i$  to AP  $n$  on subchannel  $m$ , respectively. Without loss of generality, we assume that  $|h_{n,1}^m|^2 > |h_{n,2}^m|^2 > \dots > |h_{n,N}^m|^2 > \dots > |h_{n,N}^m|^2$ . According to the rules of NOMA protocol, the APs apply SIC for multi-user detection. Specifically, each AP  $n$  sequentially decodes the signal from devices with higher channel gains and regards all the other signals as the interference. As such, the received signal-to-interference-plus-noise ratio (SINR) of AP  $n$  for device  $i \in U_n^m$  is given by:

$$\begin{aligned} \Upsilon_{n,i}^m &= \frac{p_{n,i}^m |h_{n,i}^m|^2}{\sum_{v=i+1}^U \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \sum_{l=1, l \neq n}^N \sum_{t=1}^U \beta_{l,i}^m p_{l,t}^m |g_{l,t}^m|^2 + \sigma^2} \end{aligned} \quad (5)$$

where  $p_{n,i}^m$  is the transmission power of device  $i \in U_n^m$ ,  $g_{l,t}^m$  is the channel coefficient of the interference link from device  $t$  served by AP  $l$  except AP  $n$  on the same subchannel  $m$ ,  $\sigma^2$  is the additive white Gaussian noise. Additionally, for successfully channel decoding with SIC through subchannel  $m$  of AP  $n$ , let  $I_n^m$  be the threshold of signal strength, only the users that satisfy the constraint  $p_{n,i}^m |h_{n,i}^m|^2 > I_n^m$ ,  $\forall i \in U_n^m$  can offload the partial model to edge server; otherwise, the entire model will be calculated on devices. Moreover,  $\sum_{v=i+1}^U \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2$  is the intra-cell interference and

$\sum_{l=1, l \neq n}^N \sum_{t=1}^U \beta_{l,t}^m p_{l,t}^m |g_{l,t}^m|^2$  is the inter-cell interference. Then, based on (5), the transmission rate of device  $i \in U_n^m$  can be expressed as:

$$R_{n,i}^m = \beta_{n,i}^m \cdot \frac{B_{up}}{M} \log_2 (1 + \Upsilon_{n,i}^m) \quad (6)$$

where  $\beta_{n,i}^m$  represents the subchannel allocation variable. Specifically,  $\beta_{n,i}^m = 1$  indicates that subchannel  $m$  is allocated to device  $i$ ; otherwise,  $\beta_{n,i}^m = 0$ . Let  $w_{s_i}$  represent the data size at the  $s_i$ -th layer. Then, the intermediate output transmission delay can be calculated as:

$$T_i^{tran-i} = \frac{w_{s_i}}{R_{n,i}^m} = \frac{w_{s_i}}{\beta_{n,i}^m \cdot \frac{B_{up}}{M} \log_2 (1 + \Upsilon_{n,i}^m)} \quad (7)$$

For the final data transmission from edge server to user with downlink NOMA scheme, the AP transmits a superposition-coded signal to each user on a subchannel [49]. In downlink transmissions, users employ SIC to decode the received superposed signal. Without loss of generality, suppose that the devices served by the same AP  $j$  on the same subchannel  $k$  are represented by  $U_j^k$ . Let  $N_j^k$  and  $H_{j,i}^k$  denote the number of devices in  $U_j^k$  and the channel coefficient of the downlink from edge server  $j$  to device  $i$ . Similar to the uplink, we assume that  $|H_{j,1}^k|^2 < |H_{j,2}^k|^2 < \dots < |H_{j,i}^k|^2 < \dots < |H_{j,N}^k|^2$ . For user 1, since it has the weakest channel coefficient, it decodes the superposed signal from edge server  $j$  without performing SIC. Then the user 1's decoded component is subtracted from the superposed signal. The subsequent user in  $U_j^k$ , i.e., user 2, can decode the received signal without interference from user 1. Following this principle, the signal received by user  $i \in U_j^k$  on subchannel  $k$  in BS  $j$  has a SINR  $\Psi_{j,i}^k$  of (8), shown at the bottom of the page, where  $|H_{j,i}^k|^2$  is channel gain of user  $i$  on subchannel  $k$ ,  $\sum_{q=i+1}^U \beta_{j,q}^k P_{j,q}^k |H_{j,q}^k|^2$  is the intra-cell interference experienced by user  $i$ ,  $\sum_{x=1, x \neq j}^N \sum_{y=1}^U \beta_{x,y}^k P_{x,y}^k |G_{x,y}^k|^2$  is the inter-cell interference experienced by user  $i$  (caused by the neighbor AP of users  $i$ ), and  $\sigma^2$  is the additive white Gaussian noise. Additionally, in (8), let  $|H_{j,i}^k|^2 P_{j,i}^k > I_j^k$ ,  $\forall i \in U_j^k$ , where  $I_j^k$  is the threshold of signal strength for successful channel decoding with SIC by subchannel  $k$  of AP  $j$ . In the downlink NOMA, considering the factors that affect the condition of subchannel, the SIC decoding order of users on subchannel  $j$  must be the weaker users (high inter-cell interference and low channel gain) decode before stronger users. According to [50], the achievable data rate of user  $i$  by SIC can be expressed as:

$$\Phi_{j,i}^k = \beta_{j,i}^k \frac{B_{down}}{M} \log_2 (1 + \Psi_{j,i}^k) \quad (9)$$

Let  $m_i$  denotes the data size of the final inference result at edge server, then the final result transmission delay can be

expressed by:

$$T_i^{tran-f} = \frac{m_i}{\Phi_{j,i}^k} = \frac{m_i}{\beta_{j,i}^k \frac{B_{down}}{M} \log_2 (1 + \Psi_{j,i}^k)} \quad (10)$$

Thus, the network transmission delay can be expressed as:

$$T_i^{trans} = T_i^{tran-i} + T_i^{tran-f} = \frac{w_{s_i}}{R_{n,i}^m} + \frac{m_i}{\Phi_{j,i}^k} \quad (11)$$

Intuitively, the overall execution latency of the task in mobile user  $i$  can be expressed as:

$$\begin{aligned} T_i &= T_i^{device} + T_i^{server} + T_i^{trans} \\ &= \sum_{\delta=1}^{s_i} \frac{f_{l_\delta}}{c_i} + \sum_{\delta=s_i+1}^{\mathcal{F}} \frac{f_{e_\delta}}{\lambda(r_i) c_{min}} + \frac{w_{s_i}}{R_{n,i}^m} + \frac{m_i}{\Phi_{j,i}^k} \end{aligned} \quad (12)$$

### C. Energy Consumption Model

The energy consumption that considered in this paper includes: 1) the energy that consumed for model inference at end device and edge server, and 2) the energy that consumed for intermediate data transmission between end device and edge server.

1) *Energy Consumption on Device*: Let  $\xi_i$  represent the effective switched capacitance of CPU, which is determined by the chip structure of mobile device  $i$ , then the energy consumption of mobile devices that caused for model inference can be calculated as:

$$E_i^l = \sum_{\delta=1}^{s_i} \xi_i c_i^2 \varphi_i f_{l_\delta} \quad (13)$$

where  $\varphi_i$  is the required CPU cycles to compute 1-bit data on end device.

Let  $p_{n,i}^m$  denotes the transmission power of mobile device  $i$  on subchannel  $m$ , the energy consumption of intermediate data transmission can be computed as:

$$E_i^t = p_{n,i}^m \cdot \frac{w_{s_i}}{R_{n,i}^m} \quad (14)$$

2) *Energy Consumption on Edge Server*: For edge server, its energy consumption also comes from two aspects: the model inference and final result transmission. Based on (10), the energy consumption of intermediate data transmission can be computed as:

$$E_e^t = P_{j,i}^k \cdot \frac{m_i}{\Phi_{j,i}^k} \quad (15)$$

Let  $\xi_e$  represent the effective switched capacitance of CPU, which is determined by the chip structure of edge server  $e \in N$ , then the energy consumption of edge server that used for model inference can be calculated as:

$$E_e^l = \sum_{\delta=s_i+1}^{\mathcal{F}} \xi_e (\lambda(r_i) c_{min})^2 \varphi_e f_{l_\delta} \quad (16)$$

where  $\varphi_e$  is the required CPU cycles to compute 1-bit data on edge server. Thus, the energy consumption for task execution

$$\Psi_{j,i}^k = \frac{|H_{j,i}^k|^2 P_{j,i}^k}{\sum_{q=i+1}^U \beta_{j,q}^k P_{j,q}^k |H_{j,q}^k|^2 + \sum_{x=1, x \neq j}^N \sum_{y=1}^U \beta_{x,y}^k P_{x,y}^k |G_{x,y}^k|^2 + \sigma^2} \quad (8)$$

and intermediate data transmission between mobile device and edge server can be derived by:

$$\begin{aligned} E_i &= E_i^l + E_i^t + E_e^l + E_e^t \\ &= \sum_{\delta=1}^{s_i} \xi_i c_i^2 \varphi_i f_{l\delta} + \sum_{\delta=s_i+1}^{\mathcal{F}} \xi_e (\lambda(r_i) c_{min})^2 \varphi_e f_{l\delta} \\ &\quad + p_{n,i}^m \cdot \frac{w_{s_i}}{R_{n,i}^m} + P_{j,i}^k \cdot \frac{m_i}{\Phi_{j,i}^k} \end{aligned} \quad (17)$$

#### D. Problem Statement

As demonstrated in Section III-A, the split strategy of model  $\mathcal{M}$  is  $S^{\mathcal{M}} = s_i^{\mathcal{M}} (\forall i \in \{1, 2, \dots, \mathcal{F}_{\mathcal{M}}\})$ ; the intermediate data size that relates to different split strategies is  $D^{\mathcal{M}} = \{d_1^{\mathcal{M}}, d_2^{\mathcal{M}}, \dots, d_{\mathcal{F}_{\mathcal{M}}}^{\mathcal{M}}\}$ . Therefore, the split inference delay includes four parts: the inference delay on device  $T_i^{device}(s_i)$ , the inference delay on edge server  $T_i^{server}(s_i, r_i)$ , the delay for intermediate data transmission  $T_i^{tran-i}(s_i, \beta_{n,i}^m, p_{n,i}^m)$ , and the delay for final result transmission  $T_i^{tran-f}(\beta_{j,i}^k, P_{j,i}^k)$ . Similarly, the energy consumption in model split inference also includes four parts: the energy consumption on device  $E_i^l(s_i)$ , the energy consumption for intermediate data transmission  $E_i^t(s_i, \beta_{n,i}^m, p_{n,i}^m)$ , the energy consumption for final result transmission  $E_e^t(\beta_{j,i}^k, P_{j,i}^k)$ , and the energy consumption on edge server  $E_e^l(s_i, r_i)$ .

Then, based on (1), (3), (7) and (10), we obtain the inference delay:  $T_i(s_i, \beta_{n,i}^m, \beta_{j,i}^k, p_{n,i}^m, P_{j,i}^k, r_i) = T_i^{device}(s_i) + T_i^{server}(s_i, r_i) + T_i^{tran-i}(s_i, \beta_{n,i}^m, p_{n,i}^m) + T_i^{tran-f}(\beta_{j,i}^k, P_{j,i}^k)$ ; based on (13), (14), (15) and (16), we get the energy consumption:  $E_i(s_i, \beta_{n,i}^m, \beta_{j,i}^k, p_{n,i}^m, P_{j,i}^k, r_i) = E_i^l(s_i) + E_i^t(s_i, \beta_{n,i}^m, p_{n,i}^m) + E_e^t(\beta_{j,i}^k, P_{j,i}^k) + E_e^l(s_i, r_i)$ . Our purpose is to achieve minimum inference delay and energy consumption at the same time, with the variables are the model split strategy  $S^{\mathcal{M}} = s_i^{\mathcal{M}} (\forall i \in \{1, 2, \dots, \mathcal{F}_{\mathcal{M}}\})$ , the subchannel allocation strategy of user  $B_i = \{\beta_{n,i}^m, \beta_{j,i}^k\}$ , the computing resource allocation strategy  $r_i \in [r_{min}, r_{max}]$ , and the transmission power allocation strategy of device  $P_i = \{p_{n,i}^m, P_{j,i}^k\}$ . During these variables,  $S^{\mathcal{M}}$  and  $B_i$  are discrete, while  $P_i$  and  $r_i$  are continuous. Therefore, the problem (P0) that to be solved in this paper can be expressed as:

$$\min \left\{ \sum_{i=1}^U T_i(s_i, B_i, P_i, r_i), \sum_{i=1}^U E_i(s_i, B_i, P_i, r_i) \right\} \quad (18)$$

$$s.t. S^{\mathcal{M}} = s_i^{\mathcal{M}} (\forall i \in \{1, 2, \dots, \mathcal{F}_{\mathcal{M}}\}) \quad (18.a)$$

$$D^{\mathcal{M}} = \{d_1^{\mathcal{M}}, d_2^{\mathcal{M}}, \dots, d_{\mathcal{F}_{\mathcal{M}}}^{\mathcal{M}}\} \quad (18.b)$$

$$B_i \in \{0, 1\} \quad (18.c)$$

$$p_{min} \leq P_i \leq p_{max}, \quad \forall i \in [1, U] \quad (18.d)$$

$$r_{min} \leq r_i \leq r_{max}, \quad \forall i \in [1, U] \quad (18.e)$$

$$\sum_{n=1}^N \sum_{m=1}^M \beta_{n,i}^m = 1, \quad \forall i \in [1, U] \quad (18.f)$$

$$\sum_{j=1}^N \sum_{k=1}^M \beta_{j,i}^k = 1, \quad \forall i \in [1, U] \quad (18.g)$$

In P0, the constraints of (18.a), (18.b), (18.c), (18.d), and (18.e) are easy to understand; the constraints (18.f) and (18.g) mean that each user can select only one edge server and one sub-channel in one time

slot. The P0 is difficult to address, since these two optimal objectives are opposite, as demonstrated in Section I. This indicates that simultaneously finding the minimum energy consumption and minimum inference delay is impossible. Thus, in this paper, we need to find an approach to achieve an optimal tradeoff between these two objectives.

#### IV. OPTIMAL MODEL SPLIT AND RESOURCE ALLOCATION ALGORITHM

In this section, we discuss the optimal solutions for P0. Since P0 is difficult to solve, we propose an Li-GD algorithm for P0. Moreover, the properties of the proposed Li-GD algorithm are also investigated in this section.

##### A. Loop Iteration GD Algorithm

Since the optimization objectives shown in P0 are opposite, we introduce the weight-based approach to construct the utility function for each mobile user that contains both these objectives, which can be expressed as:

$$U_i = \omega_T T_i + \omega_E E_i \quad (19)$$

where  $\omega_T$  and  $\omega_E$  are the weights of inference delay and energy consumption, respectively, and  $\omega_T + \omega_E = 1$ . The weight represents the importance of each optimal objective to users.

Additionally, the  $\omega_T$  and  $\omega_E$  are hyper-parameters, which can be decided by mobile users according to their dynamic QoS requirements. For instance, if the inference delay is more important to mobile user than energy consumption, then the mobile user can set  $\omega_T > \omega_E$ . This approach is flexible and practicable due to the following reasons. First, in practice, the QoS requirements of the same mobile user may change depending on the dynamic environment. For instance, when the mobile devices have sufficient energy, the inference delay may be the primary factor to be considered; otherwise, when the application requires low inference latency, regardless of whether the energy consumption is high or not, the weight of inference delay should be large. Second, the QoS requirements of various mobile users and applications may be different. For instance, the energy supply in user A is greater than that in user B, then the weight of  $\omega_E$  in user A may be smaller than that in user B; if one application has strict restriction on inference delay, then the weight of  $\omega_T$  in this user could be larger than the other users. Thus, this approach can be adjusted according to the dynamic QoS requirements flexibly.

According to (19), the P0 can be expressed as:

$$\min \sum_{i=1}^U U_i \quad (20)$$

Let  $\Gamma = \sum_{i=1}^U U_i$ , based on (8), (12) and (16),  $\Gamma$  can be described as:

$$\begin{aligned} \Gamma &= \sum_{i=1}^U \omega_T T_i(s_i, B_i, P_i, r_i) + \sum_{i=1}^U \omega_E E_i(s_i, B_i, P_i, r_i) \\ &= \sum_{i=1}^U \omega_T \left( \sum_{\delta=1}^{s_i} \frac{f_{l\delta}}{c_i} + \sum_{\delta=s_i+1}^{\mathcal{F}} \frac{f_{e\delta}}{\lambda(r_i) c_{min}} \right. \\ &\quad \left. + \frac{w_{s_i}}{\beta_{n,i}^m \cdot \frac{B_{up}}{M} \log_2(1 + \Upsilon_{n,i}^m)} + \frac{m_i}{\beta_{j,i}^k \cdot \frac{B_{down}}{M} \log_2(1 + \Psi_{j,i}^k)} \right) \end{aligned}$$

$$\begin{aligned}
& + \sum_{i=1}^U \omega_E^i \left( \sum_{\delta=1}^{s_i} \xi_i c_i^2 \varphi_i f_{l_\delta} \right. \\
& + \sum_{\delta=s_i+1}^{\mathcal{F}} \xi_e (\lambda(r_i) c_{min})^2 \varphi_e f_{l_\delta} + p_{n,i}^m \\
& \cdot \frac{w_{s_i}}{\beta_{n,i}^m \cdot \frac{B_{up}}{M} \log_2(1 + \Upsilon_{n,i}^m)} + P_{j,i}^k \\
& \cdot \left. \frac{m_i}{\beta_{j,i}^k \cdot \frac{B_{down}}{M} \log_2(1 + \Psi_{j,i}^k)} \right) \quad (21)
\end{aligned}$$

where  $R_{n,i}^m$  and  $\Phi_{j,i}^k$  are presented in (6) and (9), respectively. In (21), the size of the tasks that calculated on the mobile device and edge server relates to  $s_i$ , i.e.,  $\sum_{\delta=1}^{s_i} f_{l_\delta}$  and  $\sum_{\delta=s_i+1}^{\mathcal{F}} f_{e_\delta}$ , and  $w_{s_i}$  relates to the intermediate data that transmitted between the end device and edge server, which cannot be relaxed as continuous variables. Therefore, we define two variables as follows. For user  $i$ , let  $f_l^{i-\delta} = \sum_{\delta=1}^{s_i} f_{l_\delta}$  and  $s_i \in \{1, 2, \dots, \mathcal{F}\}$ , then  $f_l^{i-1} = f_{l_1}$ ,  $f_l^{i-2} = f_{l_1} + f_{l_2}$ , and so on. Then we can change the variable  $s_i$  to  $f_l^i$ , and  $f_e^i \in \{f_l^{i-1}, f_l^{i-2}, \dots, f_l^{i-\mathcal{F}}\}$ , where  $f_{l_\delta}$  is calculated based on (2). Let  $Z_i = \sum_{\delta=1}^{\mathcal{F}} f_{l_\delta}$  is the size of all layers, then  $f_e^i = Z_i - f_l^i$ . Moreover,  $f_l^i$ ,  $f_e^i$ , and  $w_{s_i}$  are calculated by mobile users in advance and stored in devices with inference model.

We then introduce  $f_l^i$ ,  $f_e^i$ , and  $w_{s_i}$  into (18). For different layers of the model, we can obtain a series of utility functions  $\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_{s_i}, \dots, \Gamma_{\mathcal{F}}\}$ , where  $\Gamma_{s_i}$  can be expressed as:

$$\begin{aligned}
\Gamma_{s_i} = & \sum_{i=1}^U \omega_T^i \left( \frac{f_l^i}{c_i} + \frac{f_e^i}{\lambda(r_i) c_{min}} + \frac{w_{s_i}}{R_{n,i}^m} + \frac{m_i}{\Phi_{j,i}^k} \right) \\
& + \sum_{i=1}^U \omega_E^i \left( \xi_i c_i^2 \varphi_i f_l^i + \xi_e (\lambda(r_i) c_{min})^2 \varphi_e f_e^i + p_{n,i}^m \right. \\
& \cdot \left. \frac{w_{s_i}}{R_{n,i}^m} + P_{j,i}^k \cdot \frac{m_i}{\Phi_{j,i}^k} \right) \quad (22)
\end{aligned}$$

where  $f_l^i$ ,  $f_e^i$ , and  $w_{s_i}$  are already known in advance for each inference model in mobile device.

In (22), there are  $U$  mobile users. For each mobile user, we need to calculate the optimal  $B$ ,  $P$ , and  $r$ . Since the parameters  $P$  and  $r$  are continuous, the variable spaces of  $P$  and  $r$  are large and infinite dimensional. Additionally, since  $B$ ,  $P$ , and  $r$  are all related to  $s_i$ , and  $B$  and  $P$  are close coupled, it is difficult to calculate the optimal value of  $B$ ,  $P$ , and  $r$  separately. Thus, in this study, for finding the optimal tradeoff between inference delay and energy consumption, we introduce the gradient descent approach into our algorithm. To use the gradient descent to address above issue, we first need to prove that the weight function shown in (21) is differentiable which is defined as follows.

**Definition 1 ([43]):** For  $f(x, y, z)$ , if its partial derivative on  $x$ ,  $y$ , and  $z$ , i.e.,  $f'(x, y, z)|_x$ ,  $f'(x, y, z)|_y$ , and  $f'(x, y, z)|_z$ , exist and continue, the  $f(x, y, z)$  is differentiable. Based on Definition 1, we have the conclusion as follows.

**Corollary 1:** When the values of  $f_l^i$ ,  $f_e^i$ , and  $w_{s_i}$  are known in advance, and we loose the constraints of  $\beta_{n,i}^m \in \{0, 1\}$  and

$\beta_{j,i}^k \in \{0, 1\}$  to  $\beta_{n,i}^m \in [0, 1]$  and  $\beta_{j,i}^k \in [0, 1]$ , the utility function shown in (21) is differentiable.

**Proof.** Refer to Appendix A.

The Corollary 1 means that when the values of  $f_l^i$ ,  $f_e^i$ , and  $w_{s_i}$  are known, the GD approach can be used in (22) to find the optimal strategies of  $B$ ,  $P$ , and  $r$ . However, the utility function shown in (22) is only the utility when the model segmentation point is  $s_i$ , there are  $\mathcal{F}$  layers in the inference model, which means that the GD algorithm needs to be repeated  $\mathcal{F}$  times to find the global optimal solutions for  $B$ ,  $P$ , and  $r$ . However, considering the complexity and convergence time of GD approach, repeating the GD approach  $\mathcal{F}$  times will cause serious delay and complexity. Fortunately, for the GD approach, if we can select the initial value carefully, the complexity and convergence time can be significantly reduced. Therefore, in this study, based on the greedy approach, we propose the Loop iteration GD algorithm, which is referred to as Li-GD. The details of the Li-GD algorithm are presented in TABLE I.

The Li-GD algorithm presented in TABLE I is composed of three parts.

1) **(Line2-Line12):** Calculating the optimal strategy when the model segmentation point is in the first layer. For the Li-GD algorithm, since the model segmentation strategy is discrete, we need to calculate the optimal resource allocation strategies layer by layer. For the first layer, its starting values are  $\mathbf{B}^{1(0)} = \{B_1^{1(0)}, \dots, B_U^{1(0)}\}$ ,  $\mathbf{P}^{1(0)} = \{P_1^{1(0)}, \dots, P_U^{1(0)}\}$  and  $\mathbf{r}^{1(0)} = \{r_1^{1(0)}, \dots, r_U^{1(0)}\}$ , where  $B^{1(0)} \in [0, 1]$ ,  $\mathbf{r}^{1(0)} \in [r_{min}, r_{max}]$ , and  $\mathbf{P}^{1(0)} \in [p_{min}, p_{max}]$ . Additionally, they are selected without any information of the final optimal values. Then, the GD algorithm is executed with step size  $\lambda$  and gradient  $-\mathbf{g}_k$ . After  $k$  rounds of iterations, when the threshold of accuracy is reached, the optimal solutions of resource allocation strategy for the first layer are  $\mathbf{B}^{1*} \leftarrow \mathbf{B}^{1(k)}$ ,  $\mathbf{P}^{1*} \leftarrow \mathbf{P}^{1(k)}$ , and  $\mathbf{r}^{1*} \leftarrow \mathbf{r}^{1(k)}$ .

2) **(Line13-Line16):** Calculating the optimal resource strategy for the rest layers. When the optimal resource allocation strategy of the first layer is calculated, then from the second layer, the starting values of this layer are the optimal values of the former layer whose intermediate data size is the closest with this layer. For instance, for the second layer, its starting values are  $\mathbf{B}^{2(0)} = \mathbf{B}^{1*}$ ,  $\mathbf{P}^{2(0)} = \mathbf{P}^{1*}$  and  $\mathbf{r}^{2(0)} = \mathbf{r}^{1*}$ . For the third layer, we compare the intermediate data size of the first and the second layers, if  $|d_1^M - d_3^M| > |d_2^M - d_3^M|$ , then  $\mathbf{B}^{3(0)} = \mathbf{B}^{2*}$ ,  $\mathbf{P}^{3(0)} = \mathbf{P}^{2*}$ , and  $\mathbf{r}^{3(0)} = \mathbf{r}^{2*}$ , otherwise,  $\mathbf{B}^{3(0)} = \mathbf{B}^{1*}$ ,  $\mathbf{P}^{3(0)} = \mathbf{P}^{1*}$ , and  $\mathbf{r}^{3(0)} = \mathbf{r}^{1*}$ , etc. The GD process is the same as that when calculating the optimal strategy for first layer. Therefore, in this stage, the optimal resource allocation strategies for  $\alpha$ th ( $2 \leq \alpha \leq \mathcal{F}$ ) layer are calculated: firstly, calculating  $\mathcal{L}_{\alpha-1}^M = |d_{\alpha-1}^M - d_1^M|$ ,  $\mathcal{L}_{\alpha-2}^M = |d_{\alpha-2}^M - d_2^M|, \dots, \mathcal{L}_{\alpha-(\alpha-1)}^M = |d_{\alpha-(\alpha-1)}^M - d_{\alpha-1}^M|$ ; then, let  $\alpha^* = \arg \min (\mathcal{L}_{\alpha-1}^M, \mathcal{L}_{\alpha-2}^M, \dots, \mathcal{L}_{\alpha-(\alpha-1)}^M)$ ; finally, setting  $\mathbf{B}^{\alpha(0)} = \mathbf{B}^{\alpha*}$ ,  $\mathbf{P}^{\alpha(0)} = \mathbf{P}^{\alpha*}$ , and  $\mathbf{r}^{\alpha(0)} = \mathbf{r}^{\alpha*}$ .

3) **(Line17-Line20):** Finding the final optimal model segmentation strategy and resource allocation strategy. When the optimal resource allocation strategies for all the layers

TABLE I  
THE PROPOSED LI-GD ALGORITHMS

| <b>Algorithm 1:</b> Loop iteration GD algorithm (Li-GD)                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                               |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Input:</b>                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                               |
| Objective function: $\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_{S_i}, \dots, \Gamma_{\mathcal{F}}\}$ ;                                                                                                           |                                                                                                                                                                                                                                                                                                                                               |
| Gradient function: $\nabla = \{\nabla_{B_i} = \frac{\partial \Gamma_{S_i}}{\partial B_i}, \nabla_{P_i} = \frac{\partial \Gamma_{S_i}}{\partial P_i}, \nabla_{r_i} = \frac{\partial \Gamma_{S_i}}{\partial r_i}\}$ ; |                                                                                                                                                                                                                                                                                                                                               |
| Algorithm accuracy: $\varepsilon$ ;                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                               |
| Step size: $\lambda$ ;                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                               |
| <b>Output:</b>                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                               |
| The optimal solution $\mathbf{O}^* = \{\mathbf{B}^*, \mathbf{P}^*, \mathbf{r}^*\}$ ;                                                                                                                                |                                                                                                                                                                                                                                                                                                                                               |
| 1.                                                                                                                                                                                                                  | Let $\mathbf{B}^{j(0)} \in [0, 1]$ , $\mathbf{P}^{j(0)} \in [P_{\min}, P_{\max}]$ , and $\mathbf{r}^{j(0)} \in [r_{\min}, r_{\max}]$ , $\forall i \in [1, U]$ and $\forall j \in [1, \mathcal{F}]$ ;                                                                                                                                          |
| *# Calculating the optimal strategy for the first layer **                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                               |
| 2.                                                                                                                                                                                                                  | If $j = 1$ ;                                                                                                                                                                                                                                                                                                                                  |
| 3.                                                                                                                                                                                                                  | Let $k \leftarrow 0$ , $\mathbf{B}^{j(k)} = \{B_1^{j(k)}, \dots, B_{\mathcal{F}}^{j(k)}\}$ , $\mathbf{P}^{j(k)} = \{P_1^{j(k)}, \dots, P_{\mathcal{F}}^{j(k)}\}$ and $\mathbf{r}^{j(k)} = \{r_1^{j(k)}, \dots, r_{\mathcal{F}}^{j(k)}\}$ ;                                                                                                    |
| 4.                                                                                                                                                                                                                  | Calculating $\Gamma_{S_i}(\mathbf{B}^{j(k)}, \mathbf{P}^{j(k)}, \mathbf{r}^{j(k)})$ ;                                                                                                                                                                                                                                                         |
| 5.                                                                                                                                                                                                                  | Calculating the gradient $\mathbf{g}_k = g(\mathbf{B}^{j(k)}, \mathbf{P}^{j(k)}, \mathbf{r}^{j(k)})$ ;                                                                                                                                                                                                                                        |
| 6.                                                                                                                                                                                                                  | If $\ \mathbf{g}_k\  < \varepsilon$ , then $\mathbf{B}^{j*} \leftarrow \mathbf{B}^{j(k)}$ , $\mathbf{P}^{j*} \leftarrow \mathbf{P}^{j(k)}$ and $\mathbf{r}^{j*} \leftarrow \mathbf{r}^{j(k)}$ ;                                                                                                                                               |
| 7.                                                                                                                                                                                                                  | Otherwise, let $\mathbf{z}_k = -g(\mathbf{B}^{j(k)}, \mathbf{P}^{j(k)}, \mathbf{r}^{j(k)})$ , and let $\mathbf{B}^{j(k+1)} = \mathbf{B}^{j(k)} + \lambda \mathbf{z}_k$ , $\mathbf{P}^{j(k+1)} = \mathbf{P}^{j(k)} + \lambda \mathbf{z}_k$ , and $\mathbf{r}^{j(k+1)} = \mathbf{r}^{j(k)} + \lambda \mathbf{z}_k$ ;                            |
| 8.                                                                                                                                                                                                                  | Calculating $\Gamma_{S_i}(\mathbf{B}^{j(k+1)}, \mathbf{P}^{j(k+1)}, \mathbf{r}^{j(k+1)}) = \Gamma_{S_i}(\mathbf{B}^{j(k)} + \lambda \mathbf{p}_k, \mathbf{P}^{j(k)} + \lambda \mathbf{z}_k, \mathbf{r}^{j(k)} + \lambda \mathbf{p}_k)$ ;                                                                                                      |
| 9.                                                                                                                                                                                                                  | If $\ \Gamma_{S_i}(\mathbf{B}^{j(k+1)}, \mathbf{P}^{j(k+1)}, \mathbf{r}^{j(k+1)}) - \Gamma_{S_i}(\mathbf{B}^{j(k)}, \mathbf{P}^{j(k)}, \mathbf{r}^{j(k)})\  < \varepsilon$ or $\max\{\ \mathbf{B}^{j(k+1)} - \mathbf{B}^{j(k)}\ , \ \mathbf{P}^{j(k+1)} - \mathbf{P}^{j(k)}\ , \ \mathbf{r}^{j(k+1)} - \mathbf{r}^{j(k)}\ \} < \varepsilon$ ; |
| 10.                                                                                                                                                                                                                 | then $\mathbf{B}^{j*} \leftarrow \mathbf{B}^{j(k+1)}$ , $\mathbf{P}^{j*} \leftarrow \mathbf{P}^{j(k+1)}$ , and $\mathbf{r}^{j*} \leftarrow \mathbf{r}^{j(k+1)}$ ;                                                                                                                                                                             |
| 11.                                                                                                                                                                                                                 | otherwise, $k = k + 1$ ;                                                                                                                                                                                                                                                                                                                      |
| 12.                                                                                                                                                                                                                 | end if                                                                                                                                                                                                                                                                                                                                        |
| *# Calculating the optimal strategy of the rest layers **                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                               |
| 13.                                                                                                                                                                                                                 | When $1 < j \leq \mathcal{F}$ ;                                                                                                                                                                                                                                                                                                               |
| *# Loop iteration **                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                               |
| 14.                                                                                                                                                                                                                 | Let $\mathbf{B}^{j+1(0)} = \mathbf{B}^{j*}$ , $\mathbf{P}^{j+1(0)} = \mathbf{P}^{j*}$ , and $\mathbf{r}^{j+1(0)} = \mathbf{r}^{j*}$ , $\forall i \in [1, U]$ and $\forall j \in [1, \mathcal{F}]$ ;                                                                                                                                           |
| 15.                                                                                                                                                                                                                 | repeating step 3 to Step 11;                                                                                                                                                                                                                                                                                                                  |
| 16.                                                                                                                                                                                                                 | $j = j + 1$ ;                                                                                                                                                                                                                                                                                                                                 |
| *# Finding the optimal strategy **                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                               |
| 17.                                                                                                                                                                                                                 | Calculating $\Gamma = \{\Gamma_1(\mathbf{B}^{1*}, \mathbf{P}^{1*}, \mathbf{r}^{1*}), \dots, \Gamma_{\mathcal{F}}(\mathbf{B}^{\mathcal{F}*}, \mathbf{P}^{\mathcal{F}*}, \mathbf{r}^{\mathcal{F}*})\}$ ;                                                                                                                                        |
| 18.                                                                                                                                                                                                                 | $(\mathbf{s}, \mathbf{B}, \mathbf{P}, \mathbf{r}) \leftarrow \arg \min_{\mathbf{s}, \mathbf{B}, \mathbf{P}, \mathbf{r}} \Gamma$ ;                                                                                                                                                                                                             |
| 19.                                                                                                                                                                                                                 | If $B > 0.5 \rightarrow B = 1$ ;                                                                                                                                                                                                                                                                                                              |
| 20.                                                                                                                                                                                                                 | otherwise $B = 0$ .                                                                                                                                                                                                                                                                                                                           |

are calculated, which are  $\mathbf{B}^* = \{\mathbf{B}^{1*}, \mathbf{B}^{2*}, \dots, \mathbf{B}^{\mathcal{F}*}\}$ ,  $\mathbf{P}^* = \{\mathbf{P}^{1*}, \mathbf{P}^{2*}, \dots, \mathbf{P}^{\mathcal{F}*}\}$ , and  $\mathbf{r}^* = \{\mathbf{r}^{1*}, \mathbf{r}^{2*}, \dots, \mathbf{r}^{\mathcal{F}*}\}$ , respectively, then substituting the  $\mathbf{B}^*$ ,  $\mathbf{P}^*$ , and  $\mathbf{r}^*$  into (18) and obtaining  $\mathcal{F}$  utility values  $\mathbf{U}^* = \{U^{1*}, U^{2*}, \dots, U^{\mathcal{F}*}\}$ . Finally, finding the minimum value from  $\mathbf{U}^*$ , and the model split strategy and resource allocation strategy that associated with this utility value are selected as the final optimal strategy. Finally, since the value range of  $\mathbf{B}$  is changed from  $\{0, 1\}$  to  $[0, 1]$ , we give the approximate rule as: if  $B > 0.5$ ,  $B = 1$ ; otherwise  $B = 0$ . The theoretical foundations of the Li-GD approach is that for the GD algorithm, carefully selecting the

start value can decrease the complexity and speed up the convergence greatly [51]. To prove the effectiveness of the proposed Li-GD algorithm, we give the following conclusions.

### B. The Properties of Li-GD Algorithm

In this section, we investigate the properties of Li-GD algorithm, including convergence, complexity, and approximate error. The details are presented below.

**Corollary 2:** The Li-GD algorithm is convergent, and the convergence time is  $K = \frac{\|x^0 - x^*\|_2^2}{2\eta\epsilon}$ , where  $\eta$  is the step size and  $\eta \leq \frac{1}{L}$ ,  $\epsilon$  is the threshold of accuracy.

*Proof.* Refer to Appendix B.

**Corollary 3:** The average complex of the Li-GD algorithm is  $O(X\bar{K}\mathcal{F}Mx^3 \ln^2(x))$ , where  $\bar{K} = \frac{\sum_{i=1}^{\mathcal{F}} K^i}{\mathcal{F}}$ .

*Proof.* Based on Corollary 2, the convergence time of Li-GD approach is  $K = \frac{\|x^0 - x^*\|_2^2}{2\eta\epsilon}$ . In Li-GD algorithm, for each round of iteration, we need to calculate the gradient for  $B_i$  and  $r_i$ , and the number of mobile users is  $U$ . Moreover, according to Appendix B, the gradient calculation of  $B_i$  is:

$$f'(x) = \frac{1}{x^2 \log_2(1 + \frac{1}{x})} \left( \frac{1}{(1+x) \ln 2 \log_2(1 + \frac{1}{x})} - 1 \right) \quad (23)$$

which is much more complex than  $r_i$ . The (23) can be rewritten as:

$$\begin{aligned} f'(x) &= \frac{1}{x^2 \log_2(1 + \frac{1}{x})} \left( \frac{1}{(1+x) \ln 2 \log_2(1 + \frac{1}{x})} - 1 \right) \\ &= \frac{1}{x^2 \log_2(1 + \frac{1}{x})} \cdot \frac{1}{(1+x) \ln 2 \log_2(1 + \frac{1}{x})} \\ &\quad - \frac{1}{x^2 \log_2(1 + \frac{1}{x})} \end{aligned} \quad (24)$$

Since the first term is more complex than the second term, we only consider the first term and let:

$$\begin{aligned} h'(x) &= \frac{1}{x^2 \log_2(1 + \frac{1}{x})} \cdot \frac{1}{(1+x) \ln 2 \log_2(1 + \frac{1}{x})} \\ &= \frac{1}{x^2 (1+x) [\log_2(1 + \frac{1}{x})]^2 \ln 2} = O(x^3 \ln^2(x)) \end{aligned} \quad (25)$$

Moreover, for each mobile user in each round of GD algorithm, the (25) should be calculated. Additionally, the number of layers in the Li-GD algorithm is  $\mathcal{F}$ . Thus, the worst case complexity of the Li-GD algorithm is  $O(XK^*\mathcal{F}Mx^3 \ln^2(x))$ , where  $K^*$  is the maximum convergence time during  $\mathcal{F}$  layers and  $K^* = \max\{K^1, K^2, \dots, K^{\mathcal{F}}\}$ ,  $M$  is the number of subchannels. The average complexity of the Li-GD algorithm is  $O(X\bar{K}\mathcal{F}Mx^3 \ln^2(x))$ , where  $\bar{K} = \frac{\sum_{i=1}^{\mathcal{F}} K^i}{\mathcal{F}}$ . ■

**Corollary 4:** The Li-GD algorithm can accelerate the convergence of GD algorithm while reducing complexity.

*Proof.* From Corollary 2 and Corollary 3, we can conclude that for the fixed precision  $\epsilon$  and step size  $\eta$ , the convergence time  $K$  is correlated to the starting point  $\mathbf{B}^{(0)}$ ,  $\mathbf{P}^{(0)}$ , and  $\mathbf{r}^{(0)}$ , and the complexity is associated with

the convergence time  $K$ . Thus, for reducing convergence time and complexity, the starting point should be selected carefully. Moreover, for the GD algorithm, carefully selecting the start value can decrease the complexity and speed up the convergence greatly [51]. Additionally, in the model split inference, when.

In the traditional GD algorithm, for each round of GD, the starting values are  $\mathbf{B}^{(0)}$ ,  $\mathbf{P}^{(0)}$ , and  $\mathbf{r}^{(0)}$ . Since we do not have any information about the starting values of each round GD process, they always are set to 0. The convergence time is  $K_1 = \max \left\{ \frac{\|\mathbf{B}^{(0)} - \mathbf{B}^*\|_2^2}{2\eta\epsilon}, \frac{\|\mathbf{r}^{(0)} - \mathbf{r}^*\|_2^2}{2\eta\epsilon}, \frac{\|\mathbf{P}^{(0)} - \mathbf{P}^*\|_2^2}{2\eta\epsilon} \right\}$ . However, as introduced in Section IV-A, for the Li-GD algorithm, the starting values are the optimal solutions of the former layer whose intermediate data size is the closest with this layer, which are  $\mathbf{B}^{\alpha(0)} = \mathbf{B}^{\alpha*}$ ,  $\mathbf{P}^{\alpha(0)} = \mathbf{P}^{\alpha*}$ , and  $\mathbf{r}^{\alpha(0)} = \mathbf{r}^{\alpha*}$ , where  $\alpha^* = \arg \max_{\alpha} \left( \mathcal{L}_{\alpha-1}^{\mathcal{M}}, \mathcal{L}_{\alpha-2}^{\mathcal{M}}, \dots, \mathcal{L}_{\alpha-(\alpha-1)}^{\mathcal{M}} \right)$ , i.e.,  $\mathbf{B}^{\alpha*}$ ,  $\mathbf{P}^{\alpha*}$ , and  $\mathbf{r}^{\alpha*}$  are the one of the optimal result of the form  $(\alpha-1)$  layers whose intermediate data size is the closest to  $\alpha$ th layer. Therefore, the convergence time of Li-GD algorithm is  $K_2 = \max \left\{ \frac{\|\mathbf{B}^{\alpha*} - \mathbf{B}^*\|_2^2}{2\eta\epsilon}, \frac{\|\mathbf{r}^{\alpha*} - \mathbf{r}^*\|_2^2}{2\eta\epsilon}, \frac{\|\mathbf{P}^{\alpha*} - \mathbf{P}^*\|_2^2}{2\eta\epsilon} \right\}$ . Since  $|\mathbf{B}^{\alpha*} - \mathbf{B}^*|$ ,  $|\mathbf{P}^{\alpha*} - \mathbf{P}^*|$ , and  $|\mathbf{r}^{\alpha*} - \mathbf{r}^*|$  are much smaller than  $|\mathbf{B}^{(0)} - \mathbf{B}^*|$ ,  $|\mathbf{P}^{(0)} - \mathbf{P}^*|$ , and  $|\mathbf{r}^{(0)} - \mathbf{r}^*|$ , the convergence time is accelerated.

Additionally, there are  $\mathcal{F}$  layers, for the traditional GD algorithm, the total convergence time is  $\mathcal{F}K_1$ . For Li-GD algorithm, the total convergence time is  $K_1 + \sum_{j=2}^{\mathcal{F}} K_2^j$ . Since  $K_2^j$  is much smaller than  $K_1$ , the complexity is reduced. ■

*Corollary 5:* The approximate error  $\varphi$  of Li-GD algorithm is less than  $\frac{\epsilon}{\rho_{\min}(1-B_{\max})\log_2\left(1+\frac{P_{\min}}{\Delta^*+\frac{\alpha P_{\max}}{2}}\right)}$ .

*Proof.* As shown in Corollary 1, the approximation is caused by the approximate of  $\beta_{n,i}^m \in \{0,1\}$  and  $\beta_{j,i}^k \in \{0,1\}$  to  $\beta_{n,i}^m \in [0,1]$  and  $\beta_{j,i}^k \in [0,1]$ . Therefore, according to (7), we have:

$$\partial = \frac{1}{\rho_2(1-B_i^2)\log_2\left(1+\frac{P_2}{\mathbb{R}^*+\mathbb{R}'_{B_i^2}}\right)} - \frac{1}{\rho_1 B_i^1 \log_2\left(1+\frac{P_1}{\mathbb{R}^*+\mathbb{R}'_{B_i^1}}\right)} \quad (26)$$

where  $B_i^2$  means that the value of  $B_i$  is larger than 0.5,  $\rho_2$  is the probability that  $B_i > 0.5$ ;  $B_i^1$  means that the value of  $B_i$  is smaller than 0.5;  $\rho_1$  is the probability that  $B_i < 0.5$ ;  $\mathbb{R}^*$  is the inter-cell interference and intra-cell interference under optimal circumstance;  $\mathbb{R}'_{B_i^2}$  and  $\mathbb{R}'_{B_i^1}$  are the increased intra-cell interference and inter-cell interference under these two circumstances. Additionally,  $0 < \rho_2 < 1$ ,  $0.5 < B_2 < 1$ ,  $0 < \rho_1 < 1$ , and  $0 < B_1 < 0.5$ . Let  $P_2 = \alpha P_1$ , therefore, the (61) equals to:

$$\partial < \frac{1}{\rho_2(1-B_i^2)\log_2\left(1+\frac{P_2}{\mathbb{R}^*+\mathbb{R}'_{B_i^2}}\right)} - \frac{1}{\log_2\left(1+\frac{P_1}{\mathbb{R}^*+\mathbb{R}'_{B_i^1}}\right)} \quad (27)$$

Moreover, for the  $\mathbb{R}'_{B_i^2}$  and  $\mathbb{R}'_{B_i^1}$ , we have:

$$\begin{aligned} \mathbb{R}'_{B_i} &= \rho_2(1-B_i^2)P_2 - \rho_1 B_i^1 P_1 < \frac{\rho_2 P_2 - \rho_1 P_1}{2} \\ &= \frac{P_1(\alpha\rho_2 - \rho_1)}{2} < \frac{P_{\max}(\alpha\rho_2 - \rho_1)}{2} < \frac{\alpha P_{\max}}{2} \end{aligned} \quad (28)$$

Thus, (28) equals to:

$$\begin{aligned} \partial &< \frac{1}{\rho_2(1-B_i^2)\log_2\left(1+\frac{P_2}{\Delta^*+\frac{\alpha P_{\max}}{2}}\right)} \\ &\quad - \frac{1}{\log_2\left(1+\frac{P_1}{\Delta^*+\frac{\alpha P_{\max}}{2}}\right)} \\ &< \frac{1}{\rho_2(1-B_i^2)\log_2\left(1+\frac{P_{\min}}{\Delta^*+\frac{\alpha P_{\max}}{2}}\right)} \\ &< \frac{1}{\rho_{\min}(1-B_i^{\max})\log_2\left(1+\frac{P_{\min}}{\Delta^*+\frac{\alpha P_{\max}}{2}}\right)} \end{aligned} \quad (29)$$

Moreover, the accuracy of GD algorithm is  $\epsilon$ , therefore, the approximate error of Li-GD algorithm is  $\frac{\epsilon}{\rho_{\min}(1-B_i^{\max})\log_2\left(1+\frac{P_{\min}}{\Delta^*+\frac{\alpha P_{\max}}{2}}\right)}$ . ■

Note that the convergence time can be further reduced by optimizing the step size or by using the self-adaptive step size. Moreover, lowering the accuracy can also accelerate the convergence. Therefore, by carefully achieving a tradeoff between accuracy and convergence time also can improve the performance of Li-GD algorithm. However, these are not investigated in this paper.

## V. PERFORMANCE EVALUATION

### A. Experimental Setup

1) *Network and Communication Set:* We deploy 5 APs and 1250 users in the network. The system bandwidth is 10 MHz, which is available for all the APs. In addition, we assume that each subchannel can be accessed by at most 3 devices. The uplink channels are all independent and identically distributed Rayleigh fading channels. Referring to [32], the number of subchannels is 250; the maximum transmission power of device is 25dBm; the circuit power consumption of each edge server is 50dBm; the path loss exponent is 5; the noise power spectral density is -174dbm/Hz; the CPU cycles for 1bit task are  $10^4$  cycles/bit.

2) *Dataset:* We use CIFAR-10 dataset in this paper. The CIFAR-10 dataset consists of 60000  $32 \times 32$  RGB images in 10 classes (from 0 to 9), with 50000 training images and 1000 test images per class.

3) *DNN Benchmarks:* There are many DNN models with different topologies have been proposed recently. For instance, NiN, tiny YOLOv2, VGG16, etc., are the well-known chain topology models; AlexNet, ResNet-18, etc., are the well-known DAG topology models. However, in this paper, we mainly evaluate the performance of the proposed algorithms on chain topology models, i.e., NiN (9 layers), YOLOv2 (17 layers) and VGG16 (24 layers).

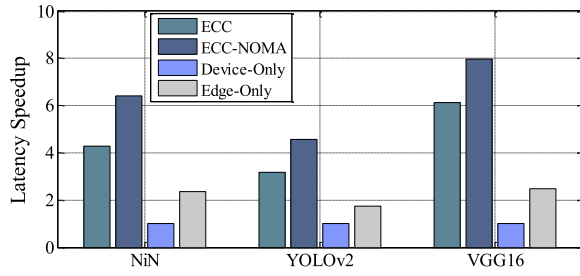


Fig. 3. Latency speedup for different DNN models.

4) *Evaluation Benchmarks*: We compare the proposed algorithms against Device-Only (i.e., executing the entire DNN on the device), Edge-Only (i.e., executing the entire DNN on the edge), IAO [15], DINA [11], Neurosurgeon [38], and DNN surgeon [14]. However, the DNN surgeon, DINA, and DPDS can operate on both chain topology models and DAG topology models. In this paper, since we mainly focus on chain topology models, we only implement these three algorithms on chain topology models (i.e., NiN, YOLOv2, and VGG16).

5) *Evaluation Variables*: In this paper, we use the latency speedup and energy consumption reduction to evaluate the performance of the benchmarks and ECC. The latency speedup means that the times of inference latency reduction compared with the baseline algorithm. For instance, assume that the baseline algorithm is Device-Only, if the ECC algorithm is applied and the inference latency in ECC is 5 times lower than that in Device-Only, then the latency speedup of ECC is 5. The meaning of energy consumption reduction is similar to that of latency speedup. However, in different simulation, the baseline algorithm is different, which will be shown in Section VI.B and Section VI.C.

## B. Performance Evaluation

In this section, we compare the performance of ECC-NOMA (ECC approach with NOMA channel) and ECC-OMA (ECC approach with OMA channel, shorted as ECC in the following sections) with that of the Device-Only, Edge-Only, Neurosurgeon, and DNN surgeon. In this section, we use the Device-Only method as the baseline, i.e., the performance is normalized to the Device-Only method.

1) *Compare the ECC-NOMA and ECC With Device-Only and Edge-Only Approach*: The performance of ECC-NOMA and ECC were compared with Device-Only and Edge-Only strategies. The performance of latency speedup and energy consumption reduction is presented in Fig.3 and Fig.4, respectively.

From Fig.3, it can be seen that compared with the Device-Only approach, both the ECC and Edge-Only approach can reduce the inference latency. For instance, without NOMA, the latency speedup of ECC is 3.1 to 8 times higher than that of the Device-Only approach. This is because the entire inference task is executed on device using the Device-Only approach, in which the computing capability is lower than the edge server. Moreover, the performance of ECC approach is also better than that of the Edge-Only approach. This is because even the computing capability of edge server is better

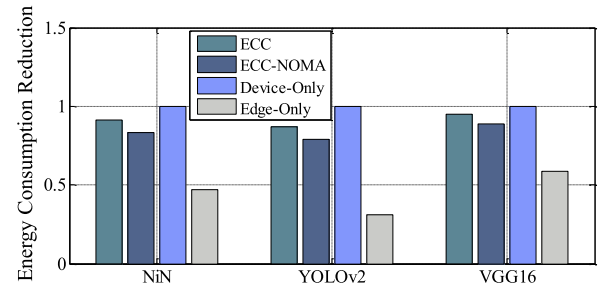


Fig. 4. Energy consumption reduction for different DNN models.

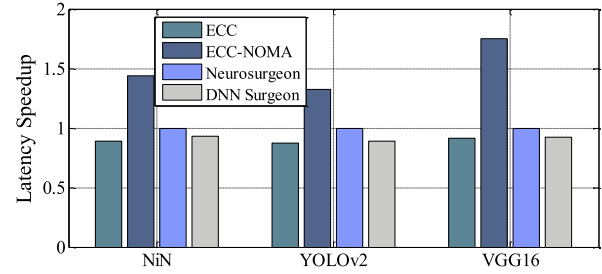


Fig. 5. Latency speedup for different DNN models.

than the device, the large amount of raw data increases the data transmission delay. Moreover, the performance of ECC with NOMA communication channel is better than that without NOMA channel. For instance, in NiN, the latency speedup of ECC without NOMA channel is 4.1 times while it is 6.1 times with NOMA channel. The performance of Device-Only approach is not affected by communication channel since it does not need to transmit data to edge server.

The performance of energy consumption reduction is shown in Fig.4. We can find that the energy consumption in both ECC and Edge-Only approach is larger than that in Device-Only approach. For instance, the energy consumption reduction of ECC is 0.85 to 0.97 times lower than that of the Device-Only approach. The reason is that in ECC and Edge-Only approach, on one hand, there is large amount data transmission between device and edge server, on the other hand, the power for data processing in edge server is higher than that in device. Moreover, considering that the transmission power in the NOMA-based channel is higher than that in OMA channel to avoid intra-cell interference, the energy consumption in ECC-NOMA is higher than that in ECC.

2) *Comparing ECC-NOMA and ECC With Neurosurgeon and DNN Surgeon*: The performances of ECC-NOMA and ECC are compared with Neurosurgeon and DNN surgeon. The performance of the latency speedup and energy consumption reduction is presented in Fig.5 and Fig.6, respectively. In this section, we use the Neurosurgeon as the baseline, i.e., the performance is normalized to the Neurosurgeon algorithm.

From Fig.5, we can see that the latency speedup in ECC and DNN surgeon is similar but slightly worse than that in Neurosurgeon; however, the latency speedup in ECC-NOMA is much better than that in Neurosurgeon. This is because for ECC and DNN surgeon, the energy consumption and the limitation of computing resource in edge server are

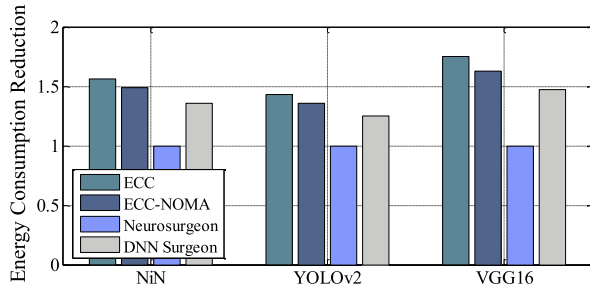


Fig. 6. Energy consumption reduction for different DNN models.

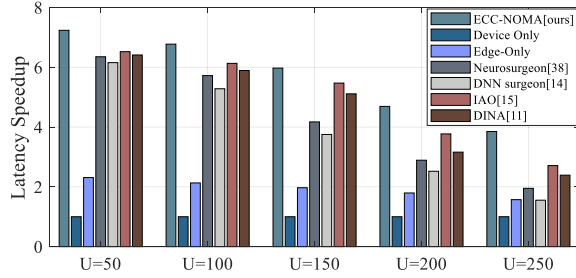


Fig. 7. Latency speedup under different user densities.

considered to find the optimal tradeoff between latency and energy consumption. Therefore, the computing resources that allocated in ECC and DNN surgeon is slightly less than that in Neurosurgeon, which causes a reduction in latency speedup. When considering the NOMA channel in ECC, the data transmission rate is speedup because the spectrum efficiency is improved. Therefore, the performance of ECC-NOMA is better than that of Neurosurgeon.

The performance of energy consumption reduction is shown in Fig.6. We can find that the energy consumption reduction in the ECC is the best. The energy consumption reduction in DNN surgeon is better than that in Neurosurgeon. For instance, the energy consumption reduction in ECC is 1.5 to 1.7 times larger than that of Neurosurgeon, which is 1.3 to 1.49 times in DNN surgeon. This is because that the Neurosurgeon and DNN surgeon do not consider the energy consumption of system, but DNN surgeon considers the resource limitation of edge server; thus, the energy consumption in DNN surgeon is less than that in Neurosurgeon. Moreover, considering that the transmission power in NOMA channel is higher than that in OMA channel to avoid intra-cell interference, the energy consumption in ECC-NOMA is higher than that in ECC, but better than the other algorithms.

### C. Performance Under Different Network Conditions

In this section, we compare the performance of ECC-NOMA [Ours] with Device-Only, Edge-Only, Neurosurgeon [38], DNN surgeon [14], IAO [15], and DINA [11] under different network conditions, including different densities of mobile users (average number of users in each edge server), different number of subchannels, and different workloads. The results are presented in Fig.7 to Fig.12, respectively. In this section, we use the Device-Only method as the baseline, i.e., the performance is normalized to the Device-Only method.

The latency of ECC-NOMA, Device-Only, Edge-Only, Neurosurgeon, DNN surgeon, IAO, and DINA under different

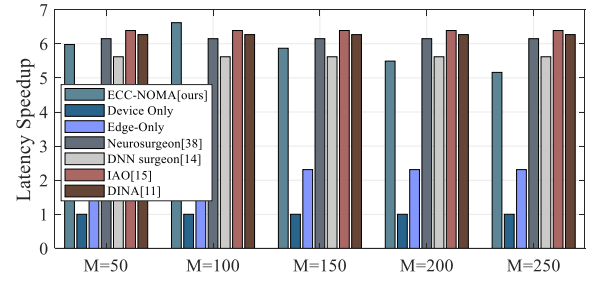


Fig. 8. Latency speedup under different number of subchannels.

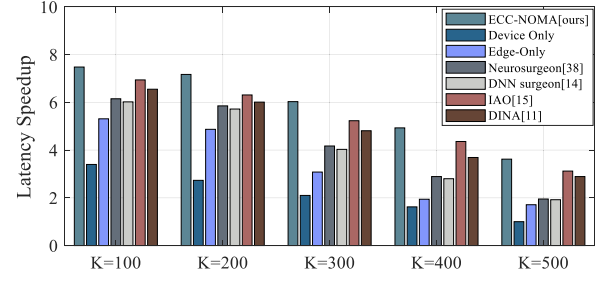


Fig. 9. Latency speedup under different workload.

user densities is presented in Fig.7. We can find that with the increasing of the user density, the latency increases in these algorithms except for Device-Only approach. Since in Device-Only approach, the whole inference model is in the mobile device, it is not affected by the variation of user density. The latency performance of ECC-NOMA is the best. The advantages of ECC becomes obviously with the increasing of the user density. For instance, when  $U = 50$ , the latency speedup of ECC-NOAM and Neurosurgeon is 7.1 and 6.3, respectively; when  $U = 250$ , this becomes 3.85 and 1.9, respectively. The reason is that the NOMA improve the spectrum efficiency, which can guarantee the data transmission rate under crowded network.

The performance of latency speedup under different number of subchannels is presented in Fig.8. We can find that with the increasing of the number of subchannels, the latency speedup of ECC-NOMA increases first then reducing after  $M = 100$ . This is because when the number of subchannels increases, the bandwidth of each subchannel reduces. Even the number of users in each subchannel reduces, if the bandwidth of each subchannel is small enough, the data transmission rate will be reduced seriously. The performance of Edge-Only, Neurosurgeon, DNN surgeon, IAO, and DINA are not affected because they do not use the NOMA channel. Moreover, when  $M = 100$ , the performance of ECC-NOMA is the best. For the other values of  $M$ , the performance of IAO is the best. The reason is the same as that in Fig.7.

The performance of latency speedup under different workloads is presented in Fig.9, where  $k$  means the average number of works in each mobile user. In Fig.9, with the increasing of workloads, the latency in mobile devices increases, too. Therefore, in this Section, we use the latency of mobile devices when  $K = 500$  as the baseline. Therefore, with the increasing of workloads, the latency increases in both these seven algorithms. The latency speedup of ECC-NOMA is better than the other algorithms. Moreover, the advantage

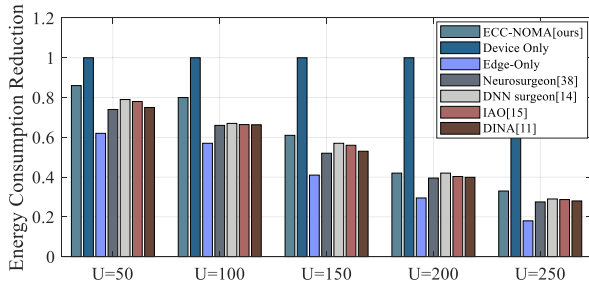


Fig. 10. Energy consumption reduction under different user densities.

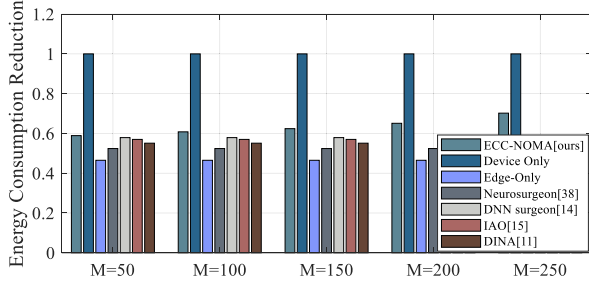


Fig. 11. Energy consumption reduction under different number of subchannels.

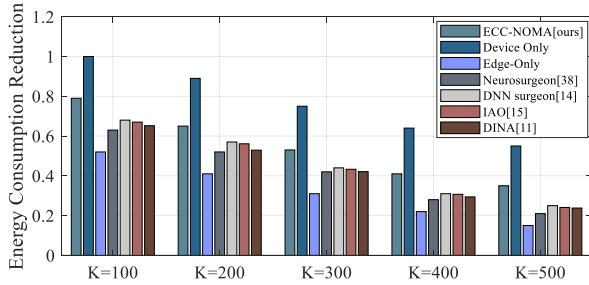


Fig. 12. Energy consumption reduction under different workload.

of ECC-NOMA becomes obviously with the increasing of workload. The reasons are similar with that in Fig.7 and Fig.8.

The energy consumption of ECC-NOMA, Device-Only, Edge-Only, Neurosurgeon, DNN surgeon, IAO, and DINA under different user densities is presented in Fig.10. We can find that with the increasing of the user density, the energy consumption increases in these algorithms except for Device-Only approach. Since in Device-Only approach, the whole inference model is in the mobile device, its energy consumption is not affected by the variation of user density. The energy consumption reduction of ECC-NOMA is the best. The advantage of ECC-NOMA becomes smaller with the increasing of the user density. This is because with the increasing of the user density, more users will share the same subchannel, for maintain high data transmission rate, the transmission power should be improved in NOMA.

The energy consumption reduction under different number of subchannels is presented in Fig.11. We can find that with the increasing of the number of subchannels, the energy consumption of ECC-NOMA reduces slightly. This is because when the number of subchannels increases, the number of mobile users in each subchannel reduces. Then the intra-cell interference reduces. Therefore, the user and edge server can lower their transmission power. However, the increasing of the number of subchannels will cause the increasing of latency,

which will contribute to the increasing of energy consumption. Thus, the energy consumption reduction in ECC-NOMA is slight. The performance of Edge-Only, Neurosurgeon, DNN surgeon, IAO, and DINA are not affected because they do not use the NOMA channel. Moreover, the performance of Device-Only is the best, the reasons are the same with that in Fig.4.

The performance of energy consumption under different workloads is presented in Fig.12. In Fig.12, with the increasing of workloads, the energy consumption in mobile devices increases, too. Therefore, in this figure, we use the latency of mobile devices when  $K = 500$  as the baseline. Therefore, with the increasing of workloads, the energy consumption increases in both these seven algorithms. The energy consumption reduction of ECC-NOMA is better than the other algorithms. Moreover, the advantage of ECC-NOMA becomes obviously with the increasing of workload. The reasons are similar with that in Fig.9.

## VI. CONCLUSION

In this paper, for accelerating the model inference at edge, we integrate the NOMA into split inference in MEC, and propose the effective communication and computing resource allocation algorithm, shorted as ECC. Specifically, when the mobile user has a large model inference task needed to be calculated in the NOMA-based MEC, it will take the energy consumption of both device and edge server and the inference latency into account to find the optimal model split strategy, subchannel allocation strategy, and transmission power allocation strategy. Since the minimum inference delay and energy consumption cannot be satisfied simultaneously, and the variables of sub-channel allocation and model split are discrete, the gradient descent (GD) algorithm is adopted to find the optimal tradeoff between them. Moreover, the loop iteration GD approach (Li-GD) is proposed to reduce the complexity of GD algorithm that caused by the discrete of model segmentation. Additionally, the properties of the proposed algorithms are also investigated, which demonstrate the effectiveness of the proposed algorithms.

## APPENDIX A

Since the values of  $f_l^i$ ,  $f_e^i$ , and  $w_{s_i}$  are already know in advance for every inference model in each mobile device, the values of  $y_1 = \frac{f_l^i}{c_i}$  and  $y_2 = \xi_i c_i^2 \varphi_i f_l^i$  can be calculated easily and nothing to do with the partial derivative on  $B_i$ ,  $P_i$ , and  $r_i$ . Therefore, the partial derivative of  $\Gamma_{s_i}$  on  $B_i$ ,  $P_i$ , and  $r_i$  can be expressed as:

$$\begin{aligned} \Gamma'_{s_i}|_{\beta_{n,i}^m} &= \sum_{i=1}^U \left( \omega_T^i \cdot \frac{w_{s_i}}{R_{n,i}^m} \right) \Big|_{\beta_{n,i}^m} \\ &\quad + \sum_{i=1}^U \left( \omega_E^i \cdot p_{n,i}^m \cdot \frac{w_{s_i}}{R_{n,i}^m} \right) \Big|_{\beta_{n,i}^m} \\ \Gamma'_{s_i}|_{\beta_{j,i}^k} &= \sum_{i=1}^U \left( \omega_T^i \cdot \frac{m_i}{\Phi_{j,i}^k} \right) \Big|_{\beta_{j,i}^k} \end{aligned} \quad (\text{A.1})$$

$$+ \sum_{i=1}^U \left( \omega_E^i \cdot P_{j,i}^k \cdot \frac{m_i}{\Phi_{j,i}^k} \right) \Big|_{\beta_{j,i}^k} \quad (\text{A.2})$$

$$\begin{aligned} \Gamma'_{s_i}|_{p_{n,i}^m} &= \sum_{i=1}^U \left( \omega_T^i \cdot \frac{w_{s_i}}{R_{n,i}^m} \right) \Big|_{p_{n,i}^m} \\ &+ \sum_{i=1}^U \left( \omega_E^i \cdot p_{n,i}^m \cdot \frac{w_{s_i}}{R_{n,i}^m} \right) \Big|_{p_{n,i}^m} \end{aligned} \quad (\text{A.3})$$

$$\begin{aligned} \Gamma'_{s_i}|_{P_{j,i}^k} &= \sum_{i=1}^U \left( \omega_T^i \cdot \frac{m_i}{\Phi_{j,i}^k} \right) \Big|_{P_{j,i}^k} \\ &+ \sum_{i=1}^U \left( \omega_E^i \cdot P_{j,i}^k \cdot \frac{m_i}{\Phi_{j,i}^k} \right) \Big|_{P_{j,i}^k} \end{aligned} \quad (\text{A.4})$$

$$\begin{aligned} \Gamma'_{s_i}|_{r_i} &= \sum_{i=1}^U \left( \omega_T^i \cdot \frac{f_e^i}{\lambda(r_i) c_{min}} \right) \Big|_{r_i} \\ &+ \sum_{i=1}^U \left( \omega_E^i \cdot \xi_e (\lambda(r_i) c_{min})^2 \varphi_e f_e^i \right) \Big|_{r_i} \end{aligned} \quad (\text{A.5})$$

Moreover, according to (A.1), let  $\Delta = \sum_{l=1, l \neq n}^N \sum_{t=1}^U \beta_{l,t}^m \cdot m_t \left| g_{l,t}^m \right|^2 + \sigma^2$ , we have:

$$\begin{aligned} &\sum_{i=1}^U \left( \omega_T^i \cdot \frac{w_{s_i}}{R_{n,i}^m} \right) \Big|_{\beta_{n,i}^m} \\ &= \omega_T^1 \frac{w_{s_i}}{R_{n,1}^m} \Big|_{\beta_{n,i}^m} + \omega_T^2 \frac{w_{s_i}}{R_{n,2}^m} \Big|_{\beta_{n,i}^m} \\ &+ \dots + \omega_T^i \frac{w_{s_i}}{R_{n,i}^m} \Big|_{\beta_{n,i}^m} + \dots + \omega_T^U \frac{w_{s_i}}{R_{n,U}^m} \Big|_{\beta_{n,i}^m} \end{aligned} \quad (\text{A.6})$$

$$\begin{aligned} &\omega_T^i \frac{w_{s_i}}{R_{n,i}^m} \Big|_{\beta_{n,i}^m} \\ &= -\omega_T^i w_{s_i} \\ &\quad \times \frac{\frac{B_{up}}{M} \log_2 \left( 1 + \frac{p_{n,i}^m |h_{n,i}^m|^2}{\sum_{v=1, v \neq i}^N \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right)}{\left\{ \beta_{n,i}^m \frac{B_{up}}{M} \log_2 \left( 1 + \frac{p_{n,i}^m |h_{n,i}^m|^2}{\sum_{v=1, v \neq i}^N \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right) \right\}^2} \\ &= -\omega_T^1 w_{s_i} \\ &\quad \times \frac{1}{(\beta_{n,i}^m)^2 \left\{ \frac{B_{up}}{M} \log_2 \left( 1 + \frac{p_{n,i}^m |h_{n,i}^m|^2}{\sum_{v=1, v \neq i}^N \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right) \right\}^2} \end{aligned} \quad (\text{A.7})$$

$$\begin{aligned} &\sum_{\tau=1, \tau \neq i}^U \left( \omega_T^\tau \cdot \frac{w_{s_i}}{R_{n,\tau}^m} \right) \Big|_{\beta_{n,\tau}^m} \\ &= \omega_T^1 \frac{w_{s_i}}{R_{n,1}^m} \Big|_{\beta_{n,1}^m} \\ &+ \dots + \omega_T^{i-1} \frac{w_{s_i}}{R_{n,i-1}^m} \Big|_{\beta_{n,i-1}^m} + \omega_T^{i+1} \frac{w_{s_i}}{R_{n,i+1}^m} \Big|_{\beta_{n,i+1}^m} \end{aligned}$$

$$\begin{aligned} &+ \dots + \omega_T^U \frac{w_{s_i}}{R_{n,U}^m} \Big|_{\beta_{n,U}^m} \\ &= \sum_{\tau=1, \tau \neq i}^U \frac{\beta_{n,\tau}^m \frac{B_{up}}{M}}{\left( 1 + \frac{p_{n,\tau}^m |h_{n,\tau}^m|^2}{\sum_{v=1, v \neq \tau}^N \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right) \ln 2} \\ &\quad \cdot \frac{p_{n,\tau}^m |h_{n,\tau}^m|^2 \cdot p_{n,i}^m |h_{n,i}^m|^2}{\left( \sum_{v=1, v \neq \tau}^N \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta \right)^2} \end{aligned} \quad (\text{A.8})$$

The calculation of  $\sum_{i=1}^U \left( \omega_E^i \cdot p_{n,i}^m \cdot \frac{w_{s_i}}{R_{n,i}^m} \right) \Big|_{\beta_{n,i}^m}$ ,  $\Gamma'_{s_i}|_{\beta_{j,i}^k}$ ,  $\Gamma'_{s_i}|_{p_{n,i}^m}$ , and  $\Gamma'_{s_i}|_{P_{j,i}^k}$  are similar with that shown in (A.7). For  $\Gamma'_{s_i}|_{\beta_{n,i}^m}$  shown in (A.1) and  $\forall \beta_{n,i}^m \in [0, 1]$ , we have  $x_1 = -\frac{\omega_T^1 w_{s_i}}{(\beta_{n,i}^m)^2 \left\{ \frac{B_{up}}{M} \log_2 \left( 1 + \frac{p_{n,i}^m |h_{n,i}^m|^2}{\sum_{v=1, v \neq i}^N \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right) \right\}^2}$  is continuous,  $x_2 = \sum_{\tau=1, \tau \neq i}^U \frac{\beta_{n,\tau}^m \frac{B_{up}}{M}}{\left( 1 + \frac{p_{n,\tau}^m |h_{n,\tau}^m|^2}{\sum_{v=1, v \neq \tau}^N \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right) \ln 2} \cdot \frac{p_{n,\tau}^m |h_{n,\tau}^m|^2 \cdot p_{n,i}^m |h_{n,i}^m|^2}{\left( \sum_{v=1, v \neq \tau}^N \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta \right)^2}$  is continuous, and  $x_3 = \sum_{i=1}^U \left( \omega_E^i \cdot p_{n,i}^m \cdot \frac{w_{s_i}}{R_{n,i}^m} \right) \Big|_{\beta_{n,i}^m}$  is continuous. Thus, based on the operational rule of continuous function, the  $\Gamma'_{s_i}|_{\beta_{n,i}^m}$  is continuous with  $\forall \beta_{n,i}^m \in [0, 1]$ . Similarly, the  $\Gamma'_{s_i}|_{\beta_{j,i}^k}$ ,  $\Gamma'_{s_i}|_{p_{n,i}^m}$ , and  $\Gamma'_{s_i}|_{P_{j,i}^k}$  are all continuous. Additionally, since  $\lambda(r_i)$  is continuous,  $\Gamma'_{s_i}|_{r_i}$  is continuous with  $\forall r_i \in [r_{min}, r_{max}]$ . Therefore, Corollary 1 is proved. ■

## APPENDIX B

Based on the conclusions in [51], if the differentiable function  $f(x)$  satisfies: 1) L-Lipschitz smooth and 2) convex, the  $f(x)$  is convergent.

Based on (22), let  $\nabla = \sum_{x=1, x \neq j}^N \sum_{y=1}^U \beta_{x,y}^k P_{x,y}^k |G_{x,y}^k|^2 + \sigma^2$ , then the parts of  $\Gamma_{s_i}$  relate to  $B_i$  are:

$$\begin{aligned} \Gamma_{s_i}(B_i) &= \sum_{i=1}^U \omega_T^i \left( \frac{w_{s_i}}{R_{n,i}^m} + \frac{m_i}{\Phi_{j,i}^k} \right) \\ &+ \sum_{i=1}^U \omega_E^i \left( p_{n,i}^m \cdot \frac{w_{s_i}}{R_{n,i}^m} + P_{j,i}^k \cdot \frac{m_i}{\Phi_{j,i}^k} \right) \\ &= \sum_{i=1}^U \omega_T^i \\ &\quad \times \left( \frac{w_{s_i}}{\beta_{n,i}^m \cdot \frac{B_{up}}{M} \log_2 \left( 1 + \frac{p_{n,i}^m |h_{n,i}^m|^2}{\sum_{v=u+1}^U \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right)} \right. \\ &\quad \left. + \frac{m_i}{\beta_{j,i}^k \frac{B_{down}}{M} \log_2 \left( 1 + \frac{|H_{j,i}^k|^2 P_{j,i}^k}{\sum_{q=u+1}^U \beta_{j,q}^k P_{j,q}^k |H_{j,q}^k|^2 + \nabla} \right)} \right) \\ &+ \sum_{i=1}^U \omega_E^i \end{aligned}$$

$$\times \left( p_{n,i}^m \cdot \frac{w_{s_i}}{\beta_{n,i}^m \cdot \frac{B_{up}}{M} \log_2 \left( 1 + \frac{p_{n,i}^m |h_{n,i}^m|^2}{\sum_{v=u+1}^U \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right)} + P_{j,i}^k \cdot \frac{m_i}{\beta_{j,i}^k \cdot \frac{B_{down}}{M} \log_2 \left( 1 + \frac{|H_{j,i}^k|^2 P_{j,i}^k}{\sum_{q=u+1}^U \beta_{j,q}^k P_{j,q}^k |H_{j,q}^k|^2 + \nabla} \right)} \right) \quad (\text{B.1})$$

Let:

$$A = \frac{w_{s_i}}{\beta_{n,i}^m \cdot \frac{B_{up}}{M} \log_2 \left( 1 + \frac{p_{n,i}^m |h_{n,i}^m|^2}{\sum_{v=u+1}^U \beta_{n,v}^m p_{n,v}^m |h_{n,v}^m|^2 + \Delta} \right)} \quad (\text{B.2})$$

$$B = \frac{m_i}{\beta_{j,i}^k \cdot \frac{B_{down}}{M} \log_2 \left( 1 + \frac{|H_{j,i}^k|^2 P_{j,i}^k}{\sum_{q=u+1}^U \beta_{j,q}^k P_{j,q}^k |H_{j,q}^k|^2 + \nabla} \right)} \quad (\text{B.3})$$

For  $\Gamma_{s_i}(B_i)$ ,  $A$  and  $B$  are similar for uplink subchannel allocation  $\beta_{n,i}^m$  and downlink subchannel allocation  $\beta_{j,i}^k$ , which can be simplified as:

$$f(x) = \frac{1}{x \log_2 \left( 1 + \frac{1}{x} \right)} \quad (\text{B.4})$$

And the first-order derivative of (B.4) can be expressed as:

$$f'(x) = \frac{1}{x^2 \log_2 \left( 1 + \frac{1}{x} \right)} \left( \frac{1}{(1+x) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)} - 1 \right) \quad (\text{B.5})$$

*L-Lipschitz smooth:*

Let  $y = kx$  and  $k > 1$ , then we have  $|x - y| = x|k - 1|$  and:

$$\begin{aligned} & |f'(x) - f'(y)| \\ &= \left| \frac{1}{k^2 x^2 \log_2 \left( 1 + \frac{1}{kx} \right)} \left( \frac{1}{(1+kx) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)} - 1 \right) - \frac{1}{x^2 \log_2 \left( 1 + \frac{1}{x} \right)} \left( \frac{1}{(1+x) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)} - 1 \right) \right| \end{aligned} \quad (\text{B.6})$$

Since  $\frac{1}{\log_2 \left( 1 + \frac{1}{x} \right)}$  increases with the increasing of  $x$  and  $\frac{1}{kx}$  decreases with the increasing of  $x$ , we have:

$$\begin{aligned} & \frac{1}{k^2 x^2 \log_2 \left( 1 + \frac{1}{kx} \right)} \left( \frac{1}{(1+kx) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)} - 1 \right) \\ & < \frac{1}{k^2 x^2 \log_2 \left( 1 + \frac{1}{x} \right)} \left( \frac{1}{(1+x) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)} - 1 \right) \end{aligned} \quad (\text{B.7})$$

Firstly, let:

$$\begin{aligned} \varphi &= \frac{\frac{1}{k^2 x^2 \log_2 \left( 1 + \frac{1}{x} \right)}}{\frac{1}{k^2 x^2 \log_2 \left( 1 + \frac{1}{kx} \right)}} = \frac{k^2 x^2 \log_2 \left( 1 + \frac{1}{kx} \right)}{k^2 x^2 \log_2 \left( 1 + \frac{1}{x} \right)} \\ &= \frac{k \log_2 \left( 1 + \frac{1}{kx} \right)}{\log_2 \left( 1 + \frac{1}{x} \right)} \end{aligned} \quad (\text{B.8})$$

Since  $0 < x \leq 1$ , we have:  $\log_2 \left( 1 + \frac{1}{x} \right) > 1$  and  $\log_2 \left( 1 + \frac{1}{kx} \right) > 1$ . Therefore:

$$\theta = \frac{k 2^{\log_2 \left( 1 + \frac{1}{kx} \right)}}{2^{\log_2 \left( 1 + \frac{1}{x} \right)}} = \frac{kx + 1}{1 + x} > 1 \quad (\text{B.9})$$

Thus, (B.9) means  $k \log_2 \left( 1 + \frac{1}{kx} \right) > \log_2 \left( 1 + \frac{1}{x} \right)$ , which also means  $\varphi > 1$ . Moreover, it is obvious that  $\frac{1}{(1+kx) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)} - 1 < \frac{1}{(1+x) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)} - 1$ .

Therefore, (B.6) can be rewritten as:

$$\begin{aligned} & |f'(x) - f'(y)| \\ & \leq \left| \frac{1}{kx^2 \log_2 \left( 1 + \frac{1}{x} \right)} \left( \frac{1}{(1+x) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)} - 1 \right) - \frac{1}{x^2 \log_2 \left( 1 + \frac{1}{x} \right)} \left( \frac{1}{(1+x) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)} - 1 \right) \right| \\ &= \frac{1}{x^2 (1+x) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)} \\ & \quad \times \left| \frac{1}{k \log_2 \left( 1 + \frac{1}{kx} \right)} - \frac{1}{\log_2 \left( 1 + \frac{1}{x} \right)} \right| \\ & \leq \frac{1}{x^2 (1+x) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)} \\ & \quad \times \left| \frac{1}{\log_2 \left( 1 + \frac{1}{kx} \right)} - \frac{1}{\log_2 \left( 1 + \frac{1}{x} \right)} \right| \end{aligned} \quad (\text{B.10})$$

If the (B.10) is no larger than  $L|x - y|$  holds, then the following constraints should be satisfied:

$$\begin{aligned} & \frac{1}{x^2 (1+x) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)} \left| \frac{1}{\log_2 \left( 1 + \frac{1}{kx} \right)} - \frac{1}{\log_2 \left( 1 + \frac{1}{x} \right)} \right| \\ & < Lx|k - 1| = L|x - y| \end{aligned} \quad (\text{B.11})$$

The (B.11) equals to:

$$\begin{aligned} L &> \frac{1}{(k-1)x^3 (1+x) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)} \\ & \quad \times \left| \frac{1}{\log_2 \left( 1 + \frac{1}{kx} \right)} - \frac{1}{\log_2 \left( 1 + \frac{1}{x} \right)} \right| \end{aligned} \quad (\text{B.12})$$

Moreover, it is easy to be proved that  $j(x) = \frac{1}{\log_2 \left( 1 + \frac{1}{kx} \right)} - \frac{1}{\log_2 \left( 1 + \frac{1}{x} \right)}$  is monotone increasing with  $x$ , which is presented as follows.

$$\begin{aligned} j'(x) &= \frac{1}{kx^2 \left( 1 + \frac{1}{kx} \right) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)} \\ & \quad - \frac{1}{x^2 \left( 1 + \frac{1}{x} \right) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)} \end{aligned} \quad (\text{B.13})$$

Let:

$$\begin{aligned} l(x) &= \frac{\frac{1}{kx^2 \left( 1 + \frac{1}{kx} \right) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)}}{\frac{1}{x^2 \left( 1 + \frac{1}{x} \right) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)}} \\ &= \frac{x^2 \left( 1 + \frac{1}{x} \right) \ln 2 \log_2 \left( 1 + \frac{1}{x} \right)^2}{kx^2 \left( 1 + \frac{1}{kx} \right) \ln 2 \log_2 \left( 1 + \frac{1}{kx} \right)^2} \\ &= \frac{(1+x) \log_2 \left( 1 + \frac{1}{x} \right)^2}{(1+kx) \log_2 \left( 1 + \frac{1}{kx} \right)^2} > \frac{\log_2 \left( 1 + \frac{1}{x} \right)^2}{k \log_2 \left( 1 + \frac{1}{kx} \right)^2} \end{aligned}$$

$$\begin{aligned}
&> \frac{(\log_2(\frac{1}{k} + \frac{1}{x}))^2}{k (\log_2(1 + \frac{1}{kx}))^2} \\
&= \frac{(\log_2(k+x) - \log_2(kx))^2}{k (\log_2(1+kx) - \log_2(kx))^2} \\
&= \left( \frac{\log_2(k+x)}{\log_2(1+kx)^{\sqrt{k}}} \right)^2 \quad (\text{B.14})
\end{aligned}$$

Let  $\eta = \sqrt{k} > 1$ , then we have:

$$v(x) = k + x - (1 + kx)^\eta \quad (\text{B.15})$$

According to (B.15), we have:

$$v'(x) = 1 - \eta k (1 + kx)^{\eta-1} < 0 \quad (\text{B.16})$$

Therefore, for  $v(x)$ , when  $x = 0$ , it has the maximum value  $v(0) = k - 1 > 0$ . Thus, we can conclude that  $\frac{\log_2(k+x)}{\log_2(1+kx)^{\sqrt{k}}} > 1$ , which means that  $l(x) > 1$  and  $j'(x) > 0$ .

Thus, for (B.12), we can conclude that:

$$L > \frac{1}{2(k-1)\ln 2} \left| \frac{1}{\log_2(1 + \frac{1}{k})} - 1 \right| \quad (\text{B.17})$$

Since  $k > 1$ ,  $\frac{1}{2(k-1)\ln 2} \left| \frac{1}{\log_2(1 + \frac{1}{k})} - 1 \right| < 1$ . Therefore, according to (B.17), there must exist  $L > 1$  can satisfy the constraint in (B.11).

According to the same method, the  $\Gamma_{s_i}$  relates to  $P_i$  is the same as (B.1). Therefore, the  $\Gamma_{s_i}(P_i)$  can be simplified to four basic functions  $y(x) = \frac{1}{\log_2(1 + \frac{1}{x})}$ ,  $g(x) = \frac{1}{\log_2(1+x)}$ ,  $h(x) = \frac{x}{\log_2(1+x)}$ , and  $z(x) = \frac{x}{\log_2(1 + \frac{1}{x})}$ . The same as the process that shown above, the  $y(x)$ ,  $g(x)$ ,  $h(x)$ , and  $z(x)$  are all L-Lipschitz smooth.

*Convex:*

For the differentiable function  $f(x)$ , if  $f''(x) > 0$ , it is convex [51]. The  $f'(x)$  is presented in (B.5). Thus, let  $L = \frac{1}{x^2 \log_2(1 + \frac{1}{x})}$ ,  $\mathcal{M} = \frac{1}{(1+x)\ln 2 \log_2(1 + \frac{1}{x})} - 1$ , then we have:

$$\mathcal{L}' = \frac{1}{x^3 \log_2(1 + \frac{1}{x})} \left[ \frac{1}{(1+x)\ln 2 \log_2(1 + \frac{1}{x})} - 2 \right] \quad (\text{B.18})$$

$$\mathcal{M}' = \frac{1}{(1+x)^3 \ln 2 \log_2(1 + \frac{1}{x})} \left[ \frac{1}{x \ln 2 \log_2(1 + \frac{1}{x})} - 1 \right] \quad (\text{B.19})$$

Therefore,

$$\begin{aligned}
f''(x) &= \frac{1}{x^3 \log_2(1 + \frac{1}{x})} \left[ \frac{1}{(1+x)\ln 2 \log_2(1 + \frac{1}{x})} - 2 \right] \\
&\cdot \left[ \frac{1}{(1+x)\ln 2 \log_2(1 + \frac{1}{x})} - 1 \right] \\
&+ \frac{1}{(1+x)^3 \ln 2 \log_2(1 + \frac{1}{x})} \\
&\times \left[ \frac{1}{x \ln 2 \log_2(1 + \frac{1}{x})} - 1 \right] \cdot \frac{1}{x^2 \log_2(1 + \frac{1}{x})} \quad (\text{B.20})
\end{aligned}$$

$$\begin{aligned}
&\text{Since } \left[ \frac{1}{(1+x)\ln 2 \log_2(1 + \frac{1}{x})} - 2 \right] < \\
&\left[ \frac{1}{(1+x)\ln 2 \log_2(1 + \frac{1}{x})} - 1 \right], \text{ we have:} \\
&f''(x) > \frac{1}{x^3 \log_2(1 + \frac{1}{x})} \left[ \frac{1}{(1+x)\ln 2 \log_2(1 + \frac{1}{x})} - 2 \right]^2 \\
&+ \frac{1}{(1+x)^3 \ln 2 \log_2(1 + \frac{1}{x})} \\
&\times \left[ \frac{1}{x \ln 2 \log_2(1 + \frac{1}{x})} - 1 \right] \cdot \frac{1}{x^2 \log_2(1 + \frac{1}{x})} \quad (\text{B.21})
\end{aligned}$$

Thus, if  $g(x) = \frac{1}{x \ln 2 \log_2(1 + \frac{1}{x})} - 1 > 0$ , the function  $f(x)$  is convex. Moreover,  $g(x) > 0$  equals to:

$$\frac{1}{x \ln 2 \log_2(1 + \frac{1}{x})} > 1 \quad (\text{B.22})$$

$$\log_2\left(1 + \frac{1}{x}\right) < \frac{1}{x \ln 2} \quad (\text{B.23})$$

Based on the Bottoming formula, we have:

$$\log_2\left(1 + \frac{1}{x}\right) < \log_2 e^{\frac{1}{x}} \quad (\text{B.24})$$

which equals to:

$$\left(1 + \frac{1}{x}\right) < e^{\frac{1}{x}} \quad (\text{B.25})$$

Since the Taylor expansion of  $e^{\frac{1}{x}}$  is:

$$e^{\frac{1}{x}} = \sum_{n=0}^{\infty} \frac{(1/x)^n}{n!} \quad (\text{B.26})$$

which is larger than  $(1 + \frac{1}{x})$ . Thus, the  $f(x)$  is convex. Similar to the L-Lipschitz smooth, the  $y(x)$ ,  $g(x)$ ,  $h(x)$ , and  $z(x)$  are all convex.

Additionally, based on (22), the terms of  $\Gamma_{s_i}$  relate to  $r_i$  are:

$$\begin{aligned}
\Gamma_{s_i}(r_i) &= \sum_{i=1}^U \omega_T^i \left( \frac{f_e^i}{\lambda(r_i) c_{min}} \right) \\
&+ \sum_{i=1}^U \omega_E^i \left( \xi_e (\lambda(r_i) c_{min})^2 \varphi_e f_e^i \right) \quad (\text{B.27})
\end{aligned}$$

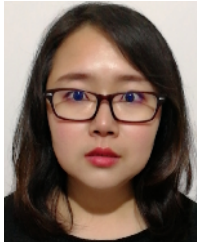
For (B.27), we assume that the  $\lambda(r_i)$  is L-Lipschitz smooth and convex, thus, for  $\Gamma_{s_i}(r_i)$ , it is also L-Lipschitz smooth and convex. Therefore, the Corollary 2 holds. ■

## REFERENCES

- [1] S. Zeng et al., "HFedMS: Heterogeneous federated learning with memorable data semantics in industrial metaverse," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 3055–3069, Mar. 2023.
- [2] M. Xu et al., "Generative AI-empowered simulation for autonomous driving in vehicular mixed reality metaverses," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 5, pp. 1064–1079, Jul. 2023.
- [3] A. Ndikumana, N. H. Tran, D. H. Kim, K. T. Kim, and C. S. Hong, "Deep learning based caching for self-driving cars in multi-access edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2862–2877, May 2021.
- [4] H. Tian et al., "CoPace: Edge computation offloading and caching for self-driving with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13281–13293, Dec. 2021.

- [5] S. Yin et al., "A survey on multimodal large language models," 2023, *arXiv:2306.13549*.
- [6] Y. Huang, X. Qiao, W. Lai, S. Dustdar, J. Zhang, and J. Li, "Enabling DNN acceleration with data and model parallelization over ubiquitous end devices," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 15053–15065, Aug. 2022.
- [7] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2021.
- [8] S. Zhang, S. Zhang, Z. Qian, J. Wu, Y. Jin, and S. Lu, "DeepSlicing: Collaborative and adaptive CNN inference with low latency," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 9, pp. 2175–2187, Sep. 2021.
- [9] F. Dong et al., "Multi-exit DNN inference acceleration based on multi-dimensional optimization for edge intelligence," *IEEE Trans. Mobile Comput.*, 2022.
- [10] C.-Y. Yang, J.-J. Kuo, J.-P. Sheu, and K.-J. Zheng, "Cooperative distributed deep neural network deployment with edge computing," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.
- [11] T. Mohammed, C. Joe-Wong, R. Babbar, and M. D. Francesco, "Distributed inference acceleration with adaptive DNN partitioning and offloading," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 854–863.
- [12] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 565–576, Feb. 2021.
- [13] M. Zhou, B. Zhou, H. Wang, F. Dong, and W. Zhao, "Dynamic path based DNN synergistic inference acceleration in edge computing environment," in *Proc. IEEE 27th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Beijing, China, Dec. 2021, pp. 567–574.
- [14] H. Liang et al., "DNN surgery: Accelerating DNN inference on the edge through layer partitioning," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 3111–3125, Mar. 2023.
- [15] X. Tang, X. Chen, L. Zeng, S. Yu, and L. Chen, "Joint multiuser DNN partitioning and computational resource allocation for collaborative edge intelligence," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9511–9522, Jun. 2021.
- [16] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.
- [17] Z. Ding, P. Fan, and H. V. Poor, "Impact of user pairing on 5G nonorthogonal multiple-access downlink transmissions," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6010–6023, Aug. 2016.
- [18] M. Vaezi, R. Schober, Z. Ding, and H. V. Poor, "Non-orthogonal multiple access: Common myths and critical questions," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 174–180, Oct. 2019.
- [19] Z. Ding, R. Schober, P. Fan, and H. V. Poor, "Simple semi-grant-free transmission strategies assisted by non-orthogonal multiple access," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4464–4478, Jun. 2019.
- [20] Z. Ding, P. Fan, G. K. Karagiannidis, R. Schober, and H. V. Poor, "NOMA assisted wireless caching: Strategies and performance analysis," *IEEE Trans. Commun.*, vol. 66, no. 10, pp. 4854–4876, Oct. 2018.
- [21] A. E. Mostafa, Y. Zhou, and V. W. S. Wong, "Connection density maximization of narrowband IoT systems with NOMA," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4708–4722, Oct. 2019.
- [22] D. Zhai, R. Zhang, L. Cai, and F. R. Yu, "Delay minimization for massive Internet of Things with non-orthogonal multiple access," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 3, pp. 553–566, Jun. 2019.
- [23] A. Kiani and N. Ansari, "Edge computing aware NOMA for 5G networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1299–1306, Apr. 2018.
- [24] L. P. Qian, A. Feng, Y. Huang, Y. Wu, B. Ji, and Z. Shi, "Optimal SIC ordering and computation resource allocation in MEC-aware NOMA NB-IoT networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2806–2816, Apr. 2019.
- [25] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12244–12258, Dec. 2018.
- [26] X. Diao, J. Zheng, Y. Wu, and Y. Cai, "Joint computing resource, power, and channel allocations for D2D-assisted and NOMA-based mobile edge computing," *IEEE Access*, vol. 7, pp. 9243–9257, 2019.
- [27] Z. Song, Y. Liu, and X. Sun, "Joint radio and computational resource allocation for NOMA-based mobile edge computing in heterogeneous networks," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2559–2562, Dec. 2018.
- [28] M. Peng, K. Zhang, J. Jiang, J. Wang, and W. Wang, "Energy-efficient resource assignment and power allocation in heterogeneous cloud radio access networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 11, pp. 5275–5287, Nov. 2015.
- [29] G. Cui et al., "OL-EUA: Online user allocation for NOMA-based mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2295–2306, Apr. 2023.
- [30] P. Lai et al., "Online user and power allocation in dynamic NOMA-based mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6676–6689, Jul. 2022.
- [31] G. Cui et al., "Demand response in NOMA-based mobile edge computing: A two-phase game-theoretical approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1449–1463, Mar. 2023.
- [32] B. Liu, C. Liu, and M. Peng, "Resource allocation for energy-efficient MEC in NOMA-enabled massive IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1015–1027, Apr. 2021.
- [33] S. Rostami, K. Heiska, O. Puchko, K. Leppanen, and M. Valkama, "Novel wake-up scheme for energy-efficient low-latency mobile devices in 5G networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1511–1528, Apr. 2021.
- [34] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.
- [35] Á. M. Groba, P. J. Lobo, and M. Chavarrías, "QoE-aware dual control system to guarantee battery lifetime for mobile video applications," *IEEE Trans. Consum. Electron.*, vol. 65, no. 4, pp. 454–463, Nov. 2019.
- [36] J. Lu et al., "A multi-task oriented framework for mobile computation offloading," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 187–201, Jan. 2022.
- [37] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.
- [38] Y. Kang et al., "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. ACM ASPLOS*, 2017, pp. 615–629.
- [39] E. Cuervo et al., "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, San Francisco, CA, USA, Jun. 2010, pp. 49–62.
- [40] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 328–339.
- [41] J. Mao et al., "MeDNN: A distributed mobile system with enhanced partition and deployment for large-scale DNNs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 751–756.
- [42] N. Jones, "How to stop data centres from gobbling up the world's electricity," *Nature*, vol. 561, no. 7722, pp. 163–166, Sep. 2018.
- [43] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018.
- [44] A. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, Apr. 2015.
- [45] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "JALAD: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2018, pp. 671–678.
- [46] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained Internet-of-Things platforms," in *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveillance (AVSS)*, Aug. 2018, pp. 1–6.
- [47] Z. Huang et al., "Enabling low latency edge intelligence based on multi-exit DNNs in the wild," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 729–739.
- [48] Q. Sun, S. Jin, J. Wang, Y. Zhang, X. Gao, and K. Wong, "Downlink massive distributed antenna systems scheduling," *IET Commun.*, vol. 9, no. 7, pp. 1006–1016, May 2015.
- [49] Y. Saito, Y. Kishiyama, A. Benjebbour, T. Nakamura, A. Li, and K. Higuchi, "Non-orthogonal multiple access (NOMA) for cellular future radio access," in *Proc. IEEE 77th Veh. Technol. Conf. (VTC Spring)*, Jun. 2013, pp. 1–5.

- [50] K. Wang, Y. Liu, Z. Ding, A. Nallanathan, and M. Peng, "User association and power allocation for multi-cell non-orthogonal multiple access networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5284–5298, Nov. 2019.
- [51] S. Bubeck, *Convex Optimization: Algorithms and Complexity*. Norwell, MA, USA: Now Foundations and Trends, 2015.
- [52] W.-Q. Ren et al., "A survey on collaborative DNN inference for edge intelligence," *Mach. Intell. Res.*, vol. 20, no. 3, pp. 370–395, Jun. 2023.
- [53] N. Li, X. Yuan, and Z. Zhang, "CoTree: Region-free and decentralized edge server cooperation," in *Proc. 2nd Workshop Flexible Resource Appl. Manage. Edge*, vol. 2019, Minneapolis, MN, USA, Jul. 2022, pp. 37–43.
- [54] Z. Ning et al., "Mobile edge computing and machine learning in the internet of unmanned aerial vehicles: A survey," *ACM Comput. Surv.*, vol. 56, no. 1, 2023, pp. 13:1–13:31, 2023.



**Xin Yuan** received the B.Eng. degree in control science and engineering from Harbin Institute of Technology (HIT), China, in 2014, and the M.S. and Ph.D. degrees from Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación (ETSIST), Universidad Politécnica de Madrid (UPM), Spain, in 2015 and 2018, respectively. She is currently an Assistant Professor with the School of Ocean Engineering, HIT. Her master's and Ph.D. studies were funded by China Scholarship Council (CSC). Her research interests include

robotic mapping and navigation, simultaneous localization and mapping (SLAM), object recognition, and sonar image processing.



**Ning Li** (Member, IEEE) received the B.S. and M.S. degrees from Zhengzhou University, China, in 2010 and 2014, respectively, and the Ph.D. degree in telematic engineering from Universidad Politécnica de Madrid (UPM), Spain, in 2018. He is currently an Associate Professor with the School of Computer Science and Technology, Harbin Institute of Technology, China. He is also a Post-Doctoral Researcher with The Hong Kong Polytechnic University. He has authored many journal articles (SCI-indexed). His main research interests include

edge computing and edge intelligence.



**Tuo Zhang** received the B.S. degree in civil engineering from Chang'an University. He is currently pursuing the M.S. degree in computer technology with Harbin Institute of Technology, Weihai, under the supervision of Assoc. Prof. Ning Li. His current research interests include edge intelligence, with a focus on the lightweight implementation of the mixture of expert (MoE) models through quantization methods.



**Muqing Li** received the bachelor's degree in computer science from China University of Geosciences (Wuhan). He is currently pursuing the master's degree with the School of Computer Science and Technology, Harbin Institute of Technology, Weihai. He is also under the guidance of Prof. Ning Li with the Network and Information Security Technology Research Center, State Key Laboratory of Computer Information Content Security. His research interests include large language model inference optimization and edge intelligence.



**Yuwen Chen** (Member, IEEE) received the M.S. degree in computer software and theory from Zhengzhou University, Zhengzhou, China, in 2015, and the Ph.D. degree in telematic engineering from the Technical University of Madrid, Madrid, Spain, in 2019. He is currently a Lecturer with the School of Computer Science, Beijing University of Technology, Beijing, China. His research interests include the IoT security and privacy, collaborative learning, model inversion, and gradient inversion.



**Jose Fernan Martinez Ortega** received the B.S. degree in electronic and telecommunications engineering and the Ph.D. degree in telecommunications engineering from Universidad Politécnica de Madrid (UPM), Spain, in 1993 and 2001, respectively. From 1993 to 1996, he was a Technical Responsible for research projects at the national telecommunications company TELECOM, Colombia. He was the Technical Manager in his own company S&H Ltda. He is currently an Associate Professor with the Department of Engineering and Telematic Architectures, UPM. He has participated in several international and European projects. His main interest areas and expertise are ubiquitous computing and the Internet of Things, smart cities, wireless sensor and actuators networks, next-generation telematic networks and services, software engineering and architectures, distributed applications and intermediation platforms (middleware), and high-performance and fault-tolerant systems. He has authored several national and international publications included in the Science Citation Index in his interest areas. He is a technical reviser and the chair of technical national and international events on telematics, as well as a member of different international and scientific committees.



**Song Guo** (Fellow, IEEE) is currently a Chair Professor with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. He has published many papers in top venues with wide impact in these areas and was recognized as a Highly Cited Researcher (Clarivate Web of Science). His research interests include big data, edge AI, mobile computing, and distributed systems. He was a member of the IEEE

ComSoc Board of Governors. He is a fellow of the Canadian Academy of Engineering and IEEE Computer Society. He was a recipient of over a dozen Best Paper Awards from IEEE/ACM conferences, journals, and technical committees. He has also served as chair of organizing and technical committees of many international conferences. He is the Chair of the IEEE Communications Society (ComSoc) Space and Satellite Communications Technical Committee. He is the Editor-in-Chief of IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY. He has served for the IEEE Computer Society on the Fellow Evaluation Committee and has been named on the editorial board of several prestigious international journals, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING. He was an IEEE ComSoc Distinguished Lecturer.