Energy-Aware Inference Offloading for DNN-Driven Applications in Mobile Edge Clouds

Zichuan Xu, *Member, IEEE*, Liqian Zhao, Weifa Liang, *Senior Member, IEEE*, Omer F. Rana, *Senior Member, IEEE*, Pan Zhou, *Senior Member, IEEE*, Qiufen Xia, *Member, IEEE*, Wenzheng Xu, *Member, IEEE*, and Guowei Wu

Abstract—With increasing focus on Artificial Intelligence (AI) applications, Deep Neural Networks (DNNs) have been successfully used in a number of application areas. As the number of layers and neurons in DNNs increases rapidly, significant computational resources are needed to execute a learned DNN model. This ever-increasing resource demand of DNNs is currently met by large-scale data centers with state-of-the-art GPUs. However, increasing availability of mobile edge computing and 5G technologies provide new possibilities for DNN-driven AI applications, especially where these application make use of data sets that are distributed in different locations. One fundamental process of a DNN-driven application in mobile edge clouds is the adoption of "inferencing" – the process of executing a pre-trained DNN based on newly generated image and video data from mobile devices. We investigate offloading DNN inference requests in a 5G-enabled mobile edge cloud (MEC), with the aim to admit as many inference requests as possible. We propose exact and approximate solutions to the problem of inference offloading in MECs. We also consider dynamic task offloading for inference requests, and devise an online algorithm that can be adapted in real time. The proposed algorithms are evaluated through large-scale simulations and using a real world test-bed implementation. The experimental results demonstrate that the empirical performance of the proposed algorithms outperform their theoretical counterparts and other similar heuristics reported in literature.

Index Terms—Inference offloading, mobile edge clouds, approximation and online algorithms

1 Introduction

The recent advancement of deep neural networks (DNNs) has led to the evolution of new AI applications in resource-limited mobile devices, such as face recognition, natural language processing and virtual/ augmented reality (AR/VR). These DNN-driven mobile applications are computation-intensive and delay-sensitive [4], because they need to perform inferences (learning) in real time. Due to such concerns, DNN-driven mobile applications require extensive GPU and computing resources that are normally met by large-scale data centers with powerful GPUs. However, consider that data centers are located far from users, executing the DNNs of mobile AI applications in remote data centers may incur prohibitive delays, thereby violating the latency requirements of certain mobile AI applications. Executing all layers of a DNN in a user equipment (UE) itself is impossible, because the numbers of layers of DNNs increases to achieve high identification accuracy and each UE is constrained by both energy and computation capacities [45]. Therefore, each UE may be able to partially execute a DNN [44], [48] and the rest execution of the DNN is distributed to other computing nodes of the network.

Mobile edge computing (MEC) has been envisioned as a promising technology [14] to meet stringent delay requirements of various applications by deploying computing capabilities and services in close proximity to end users. MEC also releases UEs from heavy DNN executions by enabling distributed inferences for DNN-driven AI applications. For example, MEC enables the distributed executions of a federated learning process [57]. Each UE can analyze its sensitive data via a federated learning process, and the analysis results are then sent to a location for aggregation to obtain a trained model. In this way, the UE does not need to analyze all its data and can offload the analysis of its insensitive data to nearby cloudlets in an MEC network, thereby saving energy and computing resource in the process.

In this paper, we investigate the problem of offloading DNN-driven AI applications to the cloudlets of an MEC network. Such DNN-driven AI applications usually consist of training and learning processes. Each DNN-driven AI application trains the parameters of its DNN and obtain a trained

- *Zichuan Xu, Liqian Zhao, and Guowei Wu are with the School of Software, Dalian University of Technology, Dalian 116024, China, and also with the Key Laboratory for Ubiquitous Network and Service Software, Liaoning Province, China. E-mail: {z.xu, wgwdut}@dlut.edu.cn, 31917068@mail.dlut.edu.cn.*
- *Weifa Liang is with the Research School of Computer Science, The Australian National University, Canberra, ACT 2601, Australia. E-mail: wliang@cs.anu.edu.au.*
- *Omer F. Rana is with Cardiff University, CF10 3AT Cardiff, United Kingdom. E-mail: RanaOF@cardiff.ac.uk.*
- *Pan Zhou is with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: panzhou@hust.edu.cn.*
- *Qiufen Xia is with the International School of Information Science and Engineering, Dalian University of Technology, Dalian 116024, China , and also with the Key Laboratory for Ubiquitous Network and Service Software, Liaoning Province 116024, China. E-mail: qiufenxia@dlut.edu.cn.*
- *Wenzheng Xu is with the School of Sichuan University, Chengdu 610017, China. E-mail: wenzheng.xu@scu.edu.cn.*

Manuscript received 14 July 2020; revised 22 Sept. 2020; accepted 9 Oct. 2020. Date of publication 20 Oct. 2020; date of current version 10 Nov. 2020. (Corresponding author: Zichuan Xu.) Recommended for acceptance by Y. Yang. Digital Object Identifier no. 10.1109/TPDS.2020.3032443

model. This process is computing-intensive and basically an offline algorithm based on historical training data. We thus consider that the training of DNNs is done in remote clouds with abundant GPU resources, following the typical practices in building AI applications. Instead, considering that the inference of DNN-driven applications is an online process with stringent delay requirements, we consider inference offloading in an MEC network.

Enabling dynamic and distributed offloading of inference requests in MEC networks is challenging. Specifically, conventional computation offloading in the era of cloud computing [9], [16], [32] is significantly different from inference offloading in MEC networks. First, inference requests usually deal with continuously generated data streams, e.g., video frames, by multiple layers of deep neural networks. To guarantee inference accuracy, a DNN typically has many layers to perform (generally) computationally intensive operations. This however makes the execution of a whole DNN in mobile devices impossible. Understanding how to offload different parts of a DNN into the MEC, while each part has its own data input and output, is challenging. Second, in a 5G-enabled MEC, inference offloading also requires considering both the selection of 5G base stations and cloudlets (edge servers) to guarantee real-time inference for DNN applications. Third, both 5G base stations and AI-based inference are energy-intensive [12], [30], [31], [33], minimizing the energy consumption of 5G base stations and cloudlets is a key issue for adopting the technique of 5G multi-cell MEC. Fourth, consider the fact that computing resources in an MEC is highly distributed and each cloudlet has limited resources, we need dynamic and distributed offloading of inference requests of AI applications to the cloudlets of an MEC.

Inference offloading is an emerging topic for DNN-driven AI applications in MEC networks. Although there are studies on computational offloading in MECs, they are fundamentally different from inference offloading in MECs. That is, inference offloading considers complete or partial offloading of a sequence of tasks, to process video/image data streams. However, computational offloading considers independent tasks with small amounts of input data [9], [16]. Also, most studies on computational offloading focus on energy savings in (battery powered) mobile devices, instead of energy savings in 5G base stations and cloudlets of MEC networks. Although there are a few studies that consider workflow tasks [20], the nature of 5G-enabled MEC with each UE connecting to multiple base stations is not considered in these approaches, where the assignment of users to base stations needs to incorporate the offloading process as well.

Unlike the above mentioned studies, we deal with the offloading of DNN inference requests, by considering both data flow processing by DNNs and energy consumption on base stations and cloudlets in an MEC. To the best of our knowledge, we are the first to investigate inference offloading for DNN-driven AI applications in 5G-enabled MECs to promote energy efficiency of both UEs and base stations while meeting stringent delay requirements of applications.

The main contributions of this paper are as follows.

- We formulate the inference offloading problem in a 5G-enabled MEC for DNN-driven AI applications, with the aim to minimize energy consumption of mobile devices, cloudlets and 5G base stations, or maximize the number of admitted inference requests.
- We propose exact and approximate algorithms for the admission of a single inference request, by adopting the randomized rounding technique [40].
- We devise learning-based dynamic inference offloading methods for online inference offloading in a 5G-enabled MEC.
- We evaluate the performance of the proposed algorithms via both extensive simulations and implementations in a real test-bed, demonstrating promising experimental results that can be used to support energy-efficient offloading.

The remainder of the paper is organized as follows. Section 2 summarizes state-of-the-art approaches on computational offloading in MEC networks. Section 3 introduces the system model, notations and problem definitions. Section 4 describes exact and approximate solutions to the inference offloading problem for the admission of a single inference request. Section 5 proposes learning-based methods for dynamic inference offloading requests. Section 6 evaluates the performance of the proposed algorithms, and Section 7 concludes the paper.

2 Related Work

Considering that mobile devices have limited computing resource, many researchers have been investigating the offloading of computing intensive tasks to either remote data centers or mobile edge clouds [2], [6], [8], [10], [17], [18], [19], [38], [49], [50], [52], [53], [55], [59]. In the following, we summarize the related literature in two categories: (1) conventional task/computation offloading in remote clouds and MECs, and (2) the emerging inference offloading in 5G-enabled MECs.

In the era of 3G and 4G communications, there are many studies on task/computation offloading [2], [6], [27], [28], [49]. The objectives of those studies are mainly to improve the energy efficiency of UEs, by offloading user tasks to remote clouds. For example, Altamimi *et al.* [2] investigated energy-efficient task offloading from smartphones to a remote cloud, with the aim to minimize the energy consumption of mobile devices, by considering the communication costs. Xia *et al.* [49] studied location-aware task offloading in a two-tiered mobile cloud environment, by devising an efficient online algorithm that enables fair share the use of cloudlets by consuming the same proportional of their mobile device energy, while keeping their individual service level agreements. Chen *et al.* [6] focused on the problem of computation offloading in mobile cloud computing environments, by formulating the problem into a decentralized computation offloading decision making problem among mobile users. Li *et al.* [27], [28] designed offloading strategies for both IoT applications and mobile crowdsensing applications in edge computing by leveraging deep learning and reinforcement learning techniques.

With the development of 5G ultra-dense networks, new attention has been focusing on computation offloading in 5G environments [8], [10], [38], [55]. However, none of these studies has ever considered the inference offloading for

DNN-driven AI applications. For example, Chen *et al.* [8] studied the task offloading in a software-defined ultra-dense edge network, with the objective of minimizing the delay while saving the battery life of user's equipment. Xu *et al.* [51] investigated the problem of joint service caching and task offloading for ultra-dense edge clouds, by proposing efficient online algorithms for dynamic service caching and offloading based on Lyapunov optimization and Gibbs sampling.

Neural networks and deep neural networks computations are gaining increasingly popularity to enable AI in mobile devices [3], [15], [21], [22], [24], [58]. The execution of such DNN-driven AI applications requires huge amounts of computation resource powered by state-of-the-art DNN accelerators, such as GPUs, to guarantee the performance of AI applications. Therefore, a UE with limited computing resource may not be able to run DNN-driven inference. Offloading inferences to remote clouds is the existing solution for most AI applications, by offloading their requests in terms of Machine Learning (ML) queries to the remote clouds [21]. This needs expensive bandwidth reservation from mobile devices to data centers within the core network; otherwise the delay of inference requests may be prohibitively long. Most studies focus on DNN partitioning. For example, Shi *et al.* [42] investigated the impact of DNN partitioning with the consideration of inference latency performance and the privacy risks in edge computing. Li *et al.* [29] proposed an on-demand deep learning model inference framework across UEs and edge clouds, and accelerates inference by DNN partitioning and DNN right-sizing. DDNN in [44] embeds the BranchyNet [43] and maps sections of a DNN onto a distributed computing hierarchy which performs feedforward inference over deep networks that distributed deployment geographically rather than processed in parallel over servers-switch connected in mobile edge cloud. These studies however do not consider energy-efficient offloading of online inference requests in 5G-enabled MECs.

There are several studies on inference offloading in 5G-enabled MECs. Most of these studies aim to enable DNN computation in resource-constrained mobile devices, by partitioning DNNs horizontally or vertically into different sub networks [15], [21], [24]. However, none of them considered energy-efficiency of both UEs and base stations in a 5G-enabled MEC. For example, Jeong [21] *et al.* considered the problem of incremental offloading of neural networks, by partitioning a DNN into different subtasks for offloading to the edge cloud. Similarly, NeuroSurgeon [24] uses a DNN partitioning scheme to enable collaborative DNN execution. An efficient prediction method for energy consumption of each layer of DNN is proposed. Based on the prediction results, each DNN is partitioned into two partitions, i.e., local and remote partitions. Hu [15] *et al.* observed that the amount of inter-layer data transfer significantly varies compared with the input data. They proposed efficient algorithms to minimize the overall delay of processing each frame of a video.

3 Preliminaries

In this section we first introduce the system model, context and notation used in this work. We then provide a more detailed description of the problem being considered.

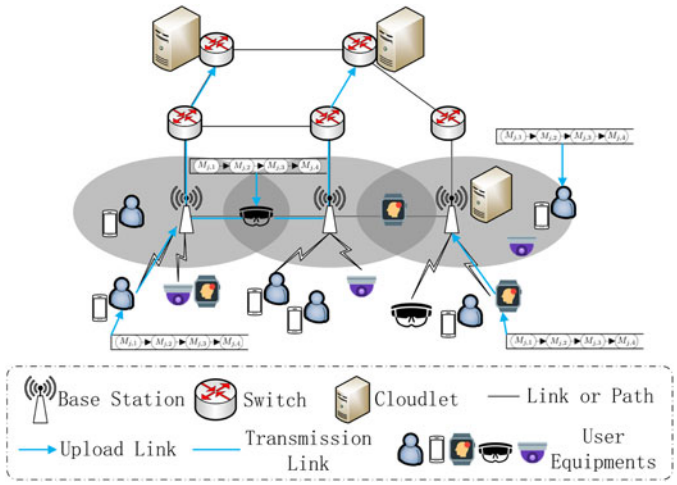


Fig. 1. An example of the 5G-enabled MEC G.

3.1 System Model

We consider a 5G-enabled MEC $G = (\mathcal{BS} \cup \mathcal{CL} \cup V, E)$ with a set $\mathcal{BS}$ of 5G small-cell base stations (BS), a set $\mathcal{CL}$ of cloudlets, a set V of switches, and a set E of links that interconnect base stations, cloudlets, and switches. In a 5G environment, an ultra-dense base station deployment approach is used (e.g., 40 - 50 BSs per km^2). Each base station BS_l has a transmission range, enabling user equipments (UEs) within its range to be connected. Cloudlets are deployed in the 5G-enabled MEC to provide computing capability. Some of them may be attached to base stations while others may be attached to switches of locations such as shopping malls and museums. Each cloudlet $CL_i \in \mathcal{CL}$ has a certain amount of computing resource to execute inference requests of AI applications. The processing units of each cloudlet can be a CPU, GPU, VPU, an accelerator such as an Intel Neural Compute Stick [5], [11], [34] or an FPGA [23], [35]. Denote by $C(CL_i)$ the computation capacity of cloudlet CL_i. Considering that the GPU and computing resource is usually virtualized to Virtual Machines (VMs) or containers, it is clear that the maximum number of VMs that can be provided by each cloudlet is limited. Following many existing studies [1], [13], [46], we thus assume that each task is executed in a single VM or container. This means that the computation capacity is represented by the maximum number of user tasks that can be executed by the cloudlet. E is a set of links (or paths) that provide connection between base stations, cloudlets, and switches. Let $e \in E$ be a link in E. Transmitting data along link e incurs a transmission delay, and denote by d_e the delay in transmitting a unit amount of data along e. Let B_e be the bandwidth of link e. Fig. 1 shows an example of the 5G-enabled MEC.

3.2 Inference Offloading and Partitioning

Since each DNN has multiple layers and matrices that are passed between layers, we consider that each DNN can include different partitions [21], [24], according to the communication workloads and energy consumptions of different layers. We assume that an inference request based on a DNN consists of a sequence of tasks with one or multiple layers for each task, as shown in Fig. 2. Note that partitioning each DNN request into a set of tasks depends on the

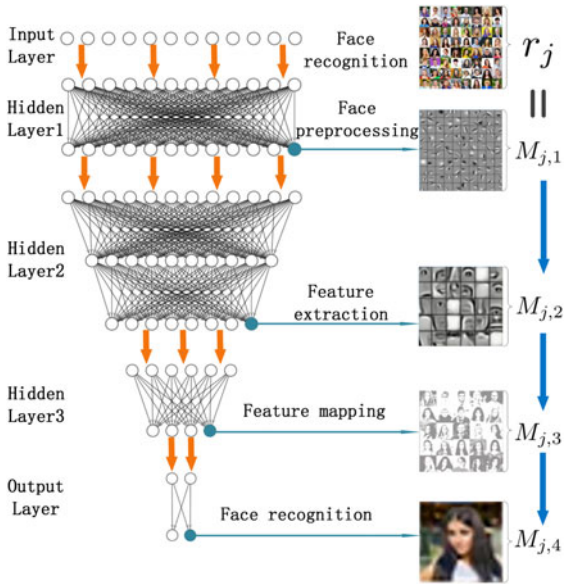


Fig. 2. An example of an inference request.

specific architecture of the DNN. Considering that there are various DNNs for different AI applications, we do not focus on the partitioning of a specific DNN into a sequence of tasks, for the sake of not losing generality. Also, there are already excellent studies on partitioning specific DNNs into different tasks [15], [21], [24]; we thus consider the partitioning of DNNs out of the scope of this paper.

Each task in the inference request takes the output matrix of its predecessor in the sequence as its input matrix and produces its output matrix as its successor task. Let r_j be a user request of an application with a sequence of tasks, denoted by $\{M_{j,1}, \ldots, M_{j,k}, \ldots M_{j,K_j}\}$, where K_j is a positive integer and $1 \leq k \leq K_j$. We consider that each inference request requires processing a data stream. For example, in a surveillance system, a stream of images must be processed to detect abnormalities or possible intrusions. Example application scenarios include vision-enabled airport security checkpoint surveillance [47], which tracks the movement of passengers and carry-on bags, continuously maintaining association between bags and passengers. In this scenario, stable video feeds from a camera network are processed in real time. Let D_j be the original data size that needs to be processed of an inference request r_j.

When an inference request arrives, it may be offloaded to cloudlets in the MEC, because its UE may not have enough resource or energy to guarantee the timely execution of the request. We thus need to decide whether to offload its tasks, depending on the capability of the UE and the resource status of the MEC. If yes, it is necessary to determine which cloudlet will be used to execute which tasks. Considering that a cloudlet in a 5G environment may not have adequate computing resources to accommodate all tasks of a specific request, we allow the requests can be partitioned into multiple cloudlets for execution.

3.3 Latency Model

As an inference request based on DNNs can be divided into different tasks, with each task requiring a number of parameter updates (based on the layers in a DNN), the execution time of each task depends on the execution time of each layer and the number of available computing resource(s) in each cloudlet. For an inference request r_j, the processing latency of each layer in its DNN is proportional to the size of its input matrix. Recall that D_j is the input data of request r_j. Since different DNNs may use different activation functions for their neurons, the input matrix of each layer may be different, depending on its initial input data D_j. However, the ratio of the input matrix of each layer and the original data size is usually bounded and given. Let α_k be a ratio of task $M_{j,k}$. It is known that $\alpha_1 = 1$. Therefore, the size of the input matrix of task $M_{j,k}$ is $\alpha_k \cdot D_j$. Note that the value of α_k can be obtained in the training of the model based on historical information of inference requests.

Following many existing studies [7], [21], [56], the latency $\delta_{j,k,i}$ of executing task $M_{j,k}$ in cloudlet CL_i can be calculated by

$$\delta_{j,k,i} = data\ size \times processing\ rate = (\alpha_k \cdot D_j) \cdot \beta_i, \tag{1}$$

where β_i is a constant identifying how long cloudlet CL_i takes to process a unit amount of data. Similarly, the latency $\delta'_{j,k}$ of executing task $M_{j,k}$ locally is

$$\delta'_{j,k} = \alpha_k \cdot D_j \cdot \gamma_j, \tag{2}$$

where γ_j is a constant identifying the latency of processing a unit amount of data locally.

Input matrices may be transferred to the 5G-enabled MEC if the tasks of r_j are assigned to different cloudlets – incurring a data upload latency. Assuming that B_l is the bandwidth of BS_l, the achieved data rate $R_{j,l}$ (bits per second) via the wireless channel of BS_l for r_j is

$$R_{j,l} = B_l \log_2\Big(1 + \frac{P_j^t \cdot h_{j,l}}{\sigma^2 + I_l}\Big), \tag{3}$$

where σ^2 is the noise power of mobile devices and I_l is the inter-cell interference power [8], $h_{j,l}$ is the channel gain between the mobile device of request r_j and the base station, and P_j^t is the transmission power of the mobile device of request r_k.

Denote by $\eta_{j,k,l}$ the upload latency of transmitting the input matrix needed by task $M_{j,k}$ of r_j via BS_l, then,

$$\eta_{j,k,l} = \frac{\alpha_k \cdot D_j}{R_{j,l}}. \tag{4}$$

Let $\hat{D}_j$ be the delay requirement of inference request r_j. The total delay experienced by r_j should be no greater than $\hat{D}_j$. It must be mentioned that our latency model can be easily extended to include other factors, such as kernel types and heterogeneity of base stations. Recall that we consider that each DNN request consisting of a sequence of tasks. Each task processes partial of a DNN and forwards its processing results to its successor in the sequence. Thus, the output of the last task of an inference request is the final results of implementing the inference request. Considering that we consider AI applications, such as object identification or attribute recognition, the final results usually are not large. We thus do not consider the transmission latency of the final results.

3.4 Energy Consumption Models of Mobile Devices and Cloudlets

We consider the energy consumptions of mobile devices, cloudlets and 5G base stations. The energy consumption of a DNN application determines the accuracy of results obtained – therefore, it impacts the quality of experience observed by a user. Significant amount of energy consumption of an application may not be desirable to most users, due to incurring a high processing delay and a corresponding energy cost. Also, the high energy consumption of 5G base stations pushes up the operational cost incurred by service providers.

Energy Consumption of Mobile Devices. Mobile devices consume energy for executing tasks associated with an inference request. Specifically, if the task is executed at mobile devices where tasks are generated, the energy consumed is mainly for running the task in mobile devices. If the task is executed at a cloudlet remotely, the energy consumption consists of the amount of energy for transferring the task and its required data to the cloudlet, the amount of energy of a mobile device waiting for the results of its offloaded tasks, and the amount of energy for receiving the results.

Recall that a task $M_{j,k}$ of an inference request r_j takes time $\delta'_{j,k}$ locally in the mobile device. Denote by $W_{j,k}$ the number of instructions of task $M_{j,k}$ that needs to be executed. Denote by P_j the processing power of the mobile device executing an inference request r_j. The energy consumption of executing task $M_{j,k}$ locally in the mobile device for request r_j is

$$e^{local}_{j,k} = \delta'_{j,k} \cdot P_j. \tag{5}$$

The energy consumption $e^{ofd}_{j,k,l}$ of the mobile device for offloading task $M_{j,k}$ to cloudlet CL_i via BS_l consists of the amounts of energy consumed on outwards data transmissions and receiving result from the K_jth task of r_j. Then,

$$e^{ofd}_{j,k,l} = \eta_{j,k,l} \cdot P^t_l + \frac{\alpha_{K_j} \cdot D_j}{R_{j,l}} \cdot P^t_l, \tag{6}$$

where P^t_l is the data transmission power of the mobile device executing r_j and α_j is the ratio of the original data and result data of the inference request r_j.

Energy Consumption of GPU-Based Cloudlets. According to [33], the energy consumption of 5G base stations is mainly due to the additional computing power attached to them. Since we consider DNN applications, we focus on the energy consumption due to the DNN processing units at base stations. We thus consider that the energy consumptions of 5G base stations are due to the processing of DNN tasks in them. The energy consumption of a DNN processing unit consists of the energy consumption of different architectural components of the unit. Since we assume each CL_i have multiple DNN processing units, we focus on a high-level energy consumption model of a cluster of DNN processing units. Following existing studies [12], [31], the energy consumption of a processing unit is proportional to the rate of task accessing on the unit and its maximum power consumption. Denote by $e^{rmt}_{j,k,i}$ the energy consumed due to executing task $M_{j,k}$ of request r_j in cloudlet CL_i, which consists of the energy consumption of its GPU unit, idle power, and leakage power, that is,

$$e^{rmt}_{j,k,i} = \delta_{j,k,i} \cdot \Big(\frac{\xi_i \cdot W_{j,k}}{\delta_{j,k,i}} P^{max}_i + P^{idle}_i + P^{leak}_i \Big), \tag{7}$$

where ξ_i is used to calculate the access rate of GPU units as shown in [12], P^{max}_i is the maximum power of all GPU units of cloudlet CL_i, P^{idle}_i is the idle power, and P^{leak}_i is the leakage power of the GPU units at cloudlet CL_i.

3.5 Problem Definitions

Given a 5G-enabled MEC G, a set of inference requests $R = \{r_j \mid 1 \le j \le |R|\}$, and a sequence of tasks $M_{j,1}, \ldots, M_{j,k}, \ldots M_{j,K_j}$ for each request r_j, we assume that the total resource capacity of the 5G-enabled MEC G is larger than the aggregate resource demand of all inference requests in R. Specifically, in a micro-scope, we are given an inference request of a UE, and need to decide which of its tasks should be offloaded. While on a macro-scope, if we are given multiple inference requests, which requests should be admitted. Therefore, we consider the following two optimization problems on inference offloading in the 5G-enabled MEC: (1) the inference offloading for a single inference request, and (2) the inference offloading for multiple requests.

The *inference offloading problem for a single inference request* is to offload the tasks of each request r_j to the cloudlets in G by not only selecting the tasks of r_j to be offloaded but also identifying which base stations and cloudlets in G must perform the offloading. The objective of the problem is to minimize the total energy consumption of the mobile device and involved cloudlets for processing r_j, subject to the processing capacity on each cloudlet and the delay requirement of r_j.

In real scenarios, the arrival of future inference requests are not known in advance. Such uncertainty creates significant difficulties in admitting inference requests. Assuming that inference requests arrive at the system sequentially without knowledge of future arrivals, the *online inference offloading problem for multiple requests* is to decide whether to admit each incoming request immediately. The objective is to admit as many inference requests in R as possible while minimizing the total energy consumption of mobile devices for the admitted inference requests and cloudlets, subject to the processing capacity on each cloudlet.

A special case of the above optimization problems with all tasks of each inference request having no dependance and without latency requirements of users can be reduced to a bin packing problem that is NP-Hard. Since the special case of the problem is NP-Hard, the original inference offloading problems in an MEC network are NP-Hard as well. Due to NP-hardness of the defined problems, we will propose approximation algorithms with approximation ratios for the above-defined optimization problem, where the approximation ratio is defined as follows. *The approximation ratio:* Given a value $\gamma \ge 1$, a *γ-approximation algorithm* for a minimization problem P_1 is a polynomial time algorithm $\mathcal{A}$ that outputs a solution whose value is no more than γ times the value of an optimal solution for any instance I of P_1, where γ is termed as the approximation ratio of algorithm $\mathcal{A}$.

For clarity, the symbols used in this paper are summarized in Table 1.

TABLE 1
Symbols and Meanings

Symbol	Description
$G = (\mathcal{BS} \cup \mathcal{CL} \cup V, E)$	A mobile edge cloud (MEC) network with a set of base stations, a set of cloudlets, a set of switches and a set of links
$C(CL_i)$	The computation capacity of cloudlet CL_i, and clearly $CL_i \in \mathcal{CL}$
R	A set of inference requests
r_j	A user request of an application
$M_{j,k}$	A set of sequence of tasks of request r_j, and clearly $1 \leq k \leq K_j$
K_j	A positive integer of request r_j which indicates the number of task
D_j	The input data size of request r_j
α_k	The input data ratio of each layer
β_i	Latency for processing a unit amount of data of CL_i
$\delta_{j,k,i}$	Latency for processing task $M_{j,k}$ in CL_i
γ_j	Latency for processing a unit amount of data locally
$\delta'_{j,k}$	Latency for processing task $M_{j,k}$ locally
B_l	The bandwidth of BS_l, and clearly $BS_l \in \mathcal{BS}$
σ^2	The noise power of mobile devices
I_l	The inter-cell interference power
$h_{j,l}$	The channel gain between request r_j and base station BS_l
P_j^t	The transmission power of the mobile device of request r_k
$R_{j,l}$	Data rate of request r_j towards BS_l within wireless channel
$\eta_{j,k,l}$	Upload latency of transmitting the input matrix by task $M_{j,k}$ of r_j via BS_l
D_j	Delay requirement of r_j
$W_{j,k}$	The number of instructions of task $M_{j,k}$
P_j	The processing power of the mobile device for executing r_j
$e_{j,k}^{local}$	Energy consumption of executing task $M_{j,k}$ locally for r_j
P_l^t	The data transmission power of mobile device executing r_j
$e_{j,k,l}^{ofd}$	Energy consumption of mobile device for offloading $M_{j,k}$ to CL_i via BS_l
ξ_i	A parameter to calculate the access rate of GPU units
P_i^{idle}	The idle power of GPU units at CL_i
P_i^{leak}	The leakage power of GPU units at CL_i
$e_{j,k,i}^{rmt}$	Energy consumption of executing $M_{j,k}$ of r_j in CL_i
x_j	The number of tasks of r_j executed locally, and clearly $0 \leq x_j \leq K_j$
y_{jki}	A binary variable that indicates whether $M_{j,k}$ is offloaded to CL_i
z_{jkl}	An indicator variable that determines whether the subtask $M_{j,k}$ for r_j is offloaded via BS_l

4 Inference Offloading for a Single Request

We first propose an exact solution for the problem by formulating an Integer Linear Program (ILP). We then devise an approximation algorithm by adopting the randomized rounding technique [40], based on the relaxation of the ILP solution.

4.1 Exact Solution to the Problem With Given Partitioned Inference Requests

The objective of the inference offloading problem for the admission of a single inference request r_j is to minimize the amount of energy consumed due to executing all tasks of r_j. To achieve this, we partition the inference request r_j into two parts: *a local execution part* and *a remote execution part*. Let x_j be the number of tasks of r_j executed locally in the mobile device of r_j. Tasks $M_{x_j+1}, \ldots, M_{K_j}$ are offloaded while tasks $M_1, \ldots, M_{x_j}$ are executed locally in the mobile device of r_j. Clearly, we have $0 \leq x_j \leq K_j$, where $x_j = 0$ means that all tasks are offloaded to the MEC network while $x_j = K_j$ means that all tasks of r_j are executed locally. Tasks in $\{M_{x_j+1}, \ldots, M_{K_j}\}$ may be executed in different cloudlets since a single cloudlet may not be able to host all the tasks. Denote by y_{jki} a binary variable that indicates whether inference task $M_{j,k}$ is offloaded to cloudlet CL_i for processing. Let z_{jkl} be an indicator variable that determines whether the subtask $M_{j,k}$ of r_j is offloaded via base station BS_l. For the sake of clear presentation, the memory capacity is not considered in this paper. However, the proposed algorithms can be easily extended to consider memory capacities by adding one more constraint on each cloudlet.

Assuming that the number of tasks to be offloaded (i.e., $K_j - x_j$) is given, the problem can be formulated as an integer linear program as follows.

$$\textbf{ILP:} \quad \min \sum_{k=1}^{K_j} \sum_{l=1}^{|\mathcal{BS}|} e_{j,k,l}^{ofd} \cdot z_{jkl} + \sum_{k=1}^{K_j} e_{j,k}^{local} \cdot (1 - \sum_{i=1}^{|\mathcal{CL}|} y_{jki}) + \sum_{k=x_j+1}^{K_j} \sum_{i=1}^{|\mathcal{CL}|} e_{j,k,i}^{rmt} \cdot y_{jki}, \tag{8}$$

subject to,

$$\sum_{i=1}^{|\mathcal{CL}|} \sum_{k=1}^{K_j} y_{jki} = K_j - x_j, \tag{9}$$

$$\sum_{i=1}^{|\mathcal{CL}|} y_{jki} = 1, \quad \forall k \text{ with } x_j + 1 \leq k \leq K_j \tag{10}$$

$$\sum_{l=1}^{|\mathcal{BS}|} z_{jkl} = 1, \quad \forall k = x_j \tag{11}$$

$$\sum_{k=1}^{K_j}\sum_{i=1}^{|\mathcal{CL}|} y_{jki} \leq C(CL_i), \quad \forall CL_i \in \mathcal{CL} \tag{12}$$

$$\sum_{k=1}^{K_j}\sum_{l=1}^{|\mathcal{BS}|} z_{jkl}\left(\eta_{j,k,l}+\frac{\alpha_{K_j}D_j}{R_{j,l}}\right)+\sum_{k=1}^{x_j}\delta'_{j,k} + \sum_{k=x_j+1}^{K_j}\sum_{i=1}^{|\mathcal{CL}|} y_{jki}\delta_{j,k,i} \leq \hat{D}_j, \quad \forall CL_i \in \mathcal{CL} \tag{13}$$

$$\sum_{k=1}^{K_j}\sum_{l=1}^{|\mathcal{BS}|} z_{jkl} = 1 \tag{14}$$

$$y_{jki}, z_{jkl} \in \{0,1\}, \tag{15}$$

where Constraints (9) and (10) say that each of the $K_j - x_j + 1$ tasks has to be offloaded to a cloudlet for processing. Constraints (11) indicate that a base station has to be selected to offload data if there is at least a task of r_j that is to be offloaded to the MEC network. Constraints (12) show that the number of offloaded tasks should not exceed its capacity. It must be mentioned that the sequence of sub-tasks means that the sequence processing of user data traffic. Since each task needs to continuously process data, its subtasks thus need to be executed concurrently. Therefore, Constraints (12) calculate the accumulative computing resource consumption, to make sure the total demand does not exceed the capacity of each base station. Constraints (13) make sure that the delay requirement $\hat{D}_j$ of inference request r_j is met. Constraints (14) mean that the output data of the x_jth data is transferred using the wireless channel of a single base station. It must be mentioned that the value of x_j needs to be determined, which makes the above formulation a quadratic program.

4.2 Randomized Approximation Algorithm

We now propose a randomized approximation algorithm for the problem. The basic idea of the algorithm is to first assume that each inference request is already partitioned; that is x_j is given. Based on the **ILP**, we devise an approximation algorithm by utilizing a randomized rounding technique, considering that a randomized rounding technique can guarantee both the efficiency and optimality of the proposed algorithm with high probability. We then remove the assumption and devise an efficient heuristic that jointly makes request partition and offloading decisions for each inference request.

Inference Offloading Based on Randomized Rounding. Given the value of x_j, we now describe the algorithm. Given the maximum number of tasks of r_j executed locally, we first relax Constraint (15) into

$$0 \leq y_{jki}, z_{jkl} \leq 1. \tag{16}$$

Then, the **ILP** is relaxed into a **LP** with objective (8), subject to Constraints (9), (10), (11), (12), (13), and (16).

It can be seen that an optimal solution to the **LP** can be obtained in polynomial time. It however may not be a feasible solution to the original problem, due to the fraction values of y_{jki}, z_{jkl} and the real value of x_j. To make the solution feasible, we round the obtained fractional solution to a feasible solution in the inference offloading problem for a single inference request. To this end, we utilize a randomized rounding technique. Specifically, for each task $M_{j,k}$ of inference request r_j, we use X_{jki} to denote an i.i.d event that task $M_{j,k}$ is assigned to cloudlet CL_i. We assign service $M_{j,k}$ to cloudlet CL_i with probability $\frac{1}{2}y_{jki}$. Similarly, we use Y_{jl} to represent an i.i.d event that the tasks of inference request r_j are offloaded through base station BS_l.

The detailed algorithm is given in Algorithm 1, which is referred to as `ApproRR`.

Algorithm 1. `ApproRR`

Input: $G = (\mathcal{BS} \cup \mathcal{CL} \cup \mathcal{V}, E)$, an inference request r_j with a sequence of tasks $M_{j,1}, \ldots, M_{j,k}, \ldots, M_{j,K_j}$ that need to be executed either locally in their mobile device or remotely in cloudlets.

Output: An offloading decision for inference request r_j.

1: Relax Constraints (15) into Constraints (16), and obtain an **LP**;
2: Obtain a fraction solution x and y by solving the **LP**;
3: **for** each task $M_{j,k}$ **do**
4: Choose a single cloudlet CL_i for task $M_{j,k}$ by setting $X_{jki} = 1$ with probability $\frac{1}{2}x_{jki}$, no cloudlet is chosen with probability $1 - \frac{1}{2}x_{jki}$;
5: **if** all X_{jki} defines a feasible solution **then**
6: Let $e(X_{jki})$ be the energy associated with the feasible solution X_{jki};
7: **if** $e(X_{jki}) \leq e_{Min}$ **then**
8: $e_{Min} \leftarrow e(X_{jki})$;
9: **else**
10: the solution obtained for a x_j is infeasible;
11: **return** e_{Min};

Joint Request Partition and Offloading. So far we assumed the value of x_j is given. We now remove this assumption, and jointly decide the value of x_j and the offloading locations of $K_j - x_j$ tasks of each inference request r_k.

Each request r_k has the number K_j of tasks. Usually, K_j is a small constant. We thus repeat algorithm `ApproRR` K_j times for each x_j. A solution that can achieve the minimum energy consumption is selected from the solutions of these K_j procedures. The steps of the randomized heuristic are given in Algorithm 2, which is referred to as `HeuRR`.

Algorithm 2. `HeuRR`

Input: $G = (\mathcal{BS} \cup \mathcal{CL} \cup \mathcal{V}, E)$, an inference request r_j with a sequence of tasks $M_{j,1}, \ldots, M_{j,k}, \ldots, M_{j,K_j}$ that need to be executed either locally in their mobile device or remotely in a few cloudlets.

Output: An offloading decision for inference request r_j.

1: $e_{Min} \leftarrow Max\_Value$;
2: **for** $x_j \leftarrow 0, \ldots, K_j$ **do**
3: Invoke algorithm `ApproRR`;
4: Let e_{x_j} be the obtained solution;
5: **if** $e_{x_j} \leq e_{Min}$ **then**
6: $e_{Min} \leftarrow e_{x_j}$;
7: **return** e_{Min};

4.3 Algorithm Analysis

We now analyze the solution feasibility and approximation ratio of Algorithm 1 in the following lemma and theorems.

Lemma 1. *Given a fixed number x_j of tasks that execute locally and assume that $C(CL_i) \geq 12 \ln |\mathcal{CL}|$ and $\gamma_j + \beta_i \cdot \frac{C(CL_i)}{2} \leq \alpha^{max} D_j$, the obtained solution by Algorithm 1 is a feasible solution while the capacity $C(CL_i)$ of each cloudlet and the delay requirement of r_j are violated with probability of $\frac{1}{|\mathcal{CL}|^2}$, where $\alpha^{max} = \max_k \alpha_k$. It must be mentioned that the probability is very small in large networks.*

Proof. Clearly, there will be $(K_j - x_j + 1)$ tasks of r_j that will be offloaded to the MEC network for processing. Each of such tasks is assigned to cloudlet CL_i with probability $\frac{1}{2} y_{jki}$. Each base station will be selected with probability $\frac{1}{2} z_{jkl}$.

We first show that the capacity of a cloudlet is violated with a small probability. Recall that in Algorithm 1, task $M_{j,k}$ is assigned to cloudlet CL_i with probability $\frac{1}{2} y_{jki}$. This means that $X_{jki} = 1$ with probability $\frac{1}{2} y_{jki}$ and $X_{jki} = 0$ with probability $1 - \frac{1}{2} y_{jki}$. Let $X_i = \sum_{k=1}^{K_j} X_{jki}$ and its expectation $E(X_i)$ is

$$E(X_i) = \sum_{k=1}^{K_j} E(X_{li}).$$

To bound $E(X_i)$, we that there is a task $M_{j,k'}$ that is assigned to cloudlet $CL_{i'}$. Considering that all other events are independent, we have for each cloudlet $CL_i \in \mathcal{CL}$ that

$$\begin{aligned} E(X_i \mid X_{jk'i'} = 1) &= \sum_{k=1}^{K_j} E(X_{jki}) \\ &= \sum_{k=1}^{K_j} \frac{1}{2} y_{jki} \leq \frac{C(CL_i)}{2}. \end{aligned} \tag{17}$$

Denote by $Pr[\cdot]$ the probability of an event. The capacity of a cloudlet CL_i is violated only if after taking out one assigned task of r_j, the remaining tasks in this cloudlet are still at least $C(CL_i)$. Calculating the probability that the capacity of each cloudlet is violated is to calculate

$$Pr[X_i \geq C(CL_i) \mid X_{jk'i'} = 1].$$

By a Chernoff bound [36] with $\mu = E(X_i)$ and $\delta = 1$, we have

$$\begin{aligned} &Pr[X_i \geq C(CL_i) \mid X_{jk'i'} = 1] \\ &= Pr[X_i \geq 2E(X_i) \mid X_{jk'i'} = 1] \\ &\leq exp\big(-\frac{E(X_i)}{3}\big) \leq exp\big(-\frac{C(CL_i)}{6}\big) \\ &\leq exp\big(-\frac{12 \ln |\mathcal{CL}|}{6}\big) \\ &= 1/|\mathcal{CL}|^2. \end{aligned} \tag{18}$$

We then calculate the probability of violating the delay requirement $\hat{D}_j$ of inference request r_j. Recall that the experienced delay of request depends on the selected base station and cloudlets. We thus first calculate the expectation of event Y_{jl}. Since each base station is selected with probability $\sum_{k=1}^{K_j} \frac{1}{2} z_{jkl}$ and only one base station is selected, we have

$$E(Y_{jl}) = K_j \cdot z_{jkl}/2. \tag{19}$$

Let Z_j be an event denoting the delay experienced by r_j. We then calculate the expectation of Z_j, i.e.,

$$\begin{aligned} E(Z_j) &= E((\alpha_{x_j} D_j)/R_{j,l} \\ &\quad + \frac{\alpha_{K_j} D_j}{R_{j,l}} + \sum_{k=1}^{x_j} \alpha_k \cdot D_j \cdot \gamma_j + \sum_{k=x_j+1}^{K_j} y_{jki} \alpha_k \cdot D_j \cdot \beta_i), \\ &= E\Big(\frac{\alpha_{x_j} D_j + \alpha_{K_j} D_j}{R_{j,l}}\Big) + E\Big(\sum_{k=1}^{x_j} \alpha_k \cdot D_j \cdot \gamma_j\Big) \\ &\quad + E(\sum_{k=x_j+1}^{K_j} y_{jki} \alpha_k \cdot D_j \cdot \beta_i), \\ &\leq \sum_{l=1}^{|\mathcal{BS}|} \frac{2\alpha^{max} D_j}{R_{j,l}} E(Y_{jl}) + E\left(\left(1 - \sum_{k=1}^{K_j} y_{jki}\right) \cdot \alpha^{max} D_j \cdot \gamma_j\right) \\ &\quad + E(\sum_{k=x_j+1}^{K_j} y_{jki} \alpha_k \cdot D_j \cdot \beta_i), \text{since} \alpha^{max} = \max_k \alpha_k \\ &= \sum_{l=1}^{|\mathcal{BS}|} \frac{\alpha^{max} D_j}{R_{j,l}} K_j \cdot z_{jkl} + \alpha^{max} D_j \cdot \gamma_j \\ &\quad - \alpha^{min} D_j \cdot \gamma_j \cdot E(X_i) + \alpha^{max} \cdot D_j \cdot \beta_i \cdot E(X_i), \\ &\leq \frac{\alpha^{max} D_j K_j}{R_{j,l}} + \alpha^{max} D_j \cdot \gamma_j \\ &\quad + (\alpha^{max} - \alpha^{min}) \cdot D_j \cdot (\beta_i - \gamma_j) \cdot E(X_i) \\ &\leq \frac{\alpha^{max} D_j K_j}{R_{j,l}} + \alpha^{max} D_j \cdot \gamma_j \\ &\quad + (\alpha^{max} - \alpha^{min}) \cdot D_j \cdot (\beta_i - \gamma_j) \cdot E(X_i) \\ &\leq \frac{\alpha^{max} D_j K_j}{R_{j,l}} + \alpha^{max} D_j \Big(\gamma_j + \beta_i \cdot \frac{C(CL_i)}{2}\Big) \\ &= \alpha^{max} D_j \Big(\frac{K_j}{R_{j,l}} + \gamma_j + \beta_i \cdot \frac{C(CL_i)}{2}\Big) \\ &\leq \alpha^{max} D_j (1 + \alpha^{max} D_j), \\ &\text{since } \gamma_j + \beta_i \cdot \frac{C(CL_i)}{2} \leq \alpha^{max} D_j \text{ and } K_j \ll R_{j,l}. \end{aligned} \tag{20}$$

By a Chernoff bound [36] with $\mu = E(Z_j)$ and $\delta = 1$, we have

$$\begin{aligned} &Pr[Z_j \geq \hat{D}_j] = Pr[Z_j \geq 2E(Z_j)] \\ &\leq exp\left(-\frac{E(Z_j)}{3}\right) \\ &\leq exp\big(-\frac{\alpha^{max} D_j (1 + \alpha^{max} D_j)}{3}\big). \end{aligned} \tag{21}$$

Recall that we have $\gamma_j + \beta_i \cdot \frac{C(CL_i)}{2} \leq \alpha^{max} D_j$, which means that

$$\alpha^{max} D_j (1 + \alpha^{max} D_j) \geq \frac{(C(CL_i))^2}{2} + \frac{3C(CL_i)}{2} + 2. \tag{22}$$

Combining inequalities (21) and (22), we have

$$\begin{aligned} &Pr[Z_j \geq \hat{D}_j] \\ &\leq exp\left(-\frac{(C(CL_i))^2 + 3C(CL_i) + 4}{6}\right) \\ &\leq exp\left(-\frac{(12\ln|\mathcal{CL}|)^2}{6}\right) \\ &\leq exp\left(-\frac{(12\ln|\mathcal{CL}|)}{6}\right) = \frac{1}{|\mathcal{CL}|^2}. \end{aligned} \tag{23}$$

□

Theorem 1. *Given a 5G-enabled MEC G, an inference request r_k with a sequence of tasks, assume that x_j is given, $M \geq 12\ln|\mathcal{CL}|$ and $K_j \geq 6\ln|\mathcal{CL}|$, the approximation ratio of Algorithm 1 is within twice the optimal solution with a high probability of $(1-\frac{1}{|\mathcal{CL}|^2})$. Note that the probability is very high when the number of of cloudlets grows in the MEC.*

Proof. We now show the approximation ratio of the proposed algorithm. Let OPT be the optimal solution of the **ILP**. Let O be the obtained solution of the randomized approximation algorithm.

Recall that X_{jki} is the event that task $M_{j,k}$ is assigned to cloudlet CL_i. Let $e(X_{jki})$ be the energy associated with event X_{jki}, and its expectation is

$$E(e(X_{jki})) = E(X_{li}) \cdot c_l^{col}(g_i) = (1/2)x_{li} \cdot c_{l,i}. \tag{24}$$

Recall that a solution is feasible only when the capacity of each cloudlet is met, the delay requirement of request r_j is met, and each to-be-offloaded task is offloaded to a single cloudlet. Denote by I the event that whether the obtained solution is feasible. By a union bound of Inequalities (18) and (23), the probability of an infeasible solution is

$$\begin{aligned} Pr[I=0] &= |\mathcal{CL}| \cdot Pr[X_i \geq C(CL_i)] \cdot Pr[Z_j \geq \hat{D}_j] \\ &< |\mathcal{CL}| \cdot \frac{1}{|\mathcal{CL}|^2} \cdot \frac{1}{|\mathcal{CL}|^2} = \frac{1}{|\mathcal{CL}|^3}, \end{aligned} \tag{25}$$

assuming that there are at least 2 cloudlets in the MEC network. This means that the lower bound of $Pr[IF=1]$ is

$$Pr[I=1] = 1 - Pr[I=0] \geq 1 - \frac{1}{|\mathcal{CL}|^3}. \tag{26}$$

Let $\mathcal{E}$ be the total energy consumption of event I, we then have

$$\begin{aligned} E(\mathcal{E}) &= \sum_{k=1}^{K_j}\sum_{l=1}^{|\mathcal{BS}|} Pr[\frac{z_{jkl}}{2} \mid I=1] \cdot e_{j,x_j,l}^{ofd} \\ &+ (K_j + 1 - \sum_{i=1}^{K_j} Pr[\frac{y_{jki}}{2} \mid I=1]) e_{j,k}^{local} \\ &+ \sum_{k=x_j+1}^{K_j}\sum_{i=1}^{|\mathcal{CL}|} Pr[\frac{y_{jki}}{2} \mid I=1] \cdot e_{j,k,i}^{rmt} \\ &\geq \frac{1}{2}(1 - \frac{1}{|\mathcal{CL}|^3}) \cdot e_{min}^{ofd} \\ &+ (K_j + 1 - \left(1 - \frac{1}{|\mathcal{CL}|^3}\right)\frac{C(CL_i)}{2}) e_{min}^{local} \\ &+ \frac{1}{2}\left(1 - \frac{1}{|\mathcal{CL}|^3}\right)(C(CL_i))^2 \cdot e_{min}^{rmt}, \\ &\text{due to inequalities (17) and (26)} \\ &\geq \frac{7}{16} + (K_j + 1 - \frac{7 \cdot C(CL_i)}{16}) + \frac{7}{16}(C(CL_i))^2, \\ &\text{assuming that } e_{min}^{ofd}, e_{min}^{local}, e_{min}^{rmt} \geq 1 \text{ and } |\mathcal{CL}| \geq 2, \\ &\geq \frac{21}{64} + K_j + 1 + \frac{7}{16}(C(CL_i) - \frac{1}{2})^2 \\ &\geq K_j, \text{since } C(CL_i) \geq 12\ln|\mathcal{CL}| \geq 2, \end{aligned} \tag{27, 28}$$

where $e_{min}^{ofd} = \min_{j,x_j,l} e_{j,x_j,l}^{ofd}$, $e_{min}^{local} = \min_{j,k} e_{j,k}^{local}$, and $e_{min}^{rmt} = \min_{j,k,i} e_{j,k,i}^{rmt}$.

To show the approximation ratio of the proposed algorithm with high probability, we calculate the following probability

$$Pr[\mathcal{E} \geq (1+\sigma)OPT], \tag{29}$$

where σ is a constant with $\sigma > 0$.

Recall that the fractional solution to **LP** is a lower bound of the optimal solution. Also, the fractional solution is considered as probabilities in the randomized algorithm. Therefore, by the definition of expectations, we have the fractional solution is the expectation of $E(\mathcal{E})$. We thus have $OPT > E(\mathcal{E})$. This means

$$Pr[\mathcal{E} \geq (1+\sigma)OPT] < Pr[\mathcal{E} \geq (1+\sigma)E(\mathcal{E})], \tag{30}$$

since the property of the upper tail property of the Chernoff bound [36]. Then, apply the Chernoff bound with $\sigma = 1$, we have

$$\begin{aligned} &Pr[\mathcal{E} \geq 2 \cdot OPT] < Pr[\mathcal{E} \geq 2E(\mathcal{E})] \\ &< e^{-\frac{1}{3}E(\mathcal{E})} < e^{-\frac{K_j}{3}}, \text{due to Ineq. (27)}, \\ &< e^{-\frac{6\ln|\mathcal{CL}|}{3}} < \frac{1}{|\mathcal{CL}|^2}. \end{aligned} \tag{31}$$

Therefore, the approximation ratio of the proposed algorithm is 2 with high probability of $(1-\frac{1}{|\mathcal{CL}|^2})$. □

Theorem 2. *Given a 5G-enabled MEC G, an inference request r_k with a sequence of tasks, there exists an efficient heuristic, i.e., `HeuRR`, that delivers a feasible solution for the inference offloading problem for a single inference request.*

Given the feasibility proof in Lemma 1 for algorithm `ApproRR`, showing this theorem is straightforward, omitted.

5 Online Inference Offloading for Multiple Requests

In this section we propose online algorithms for the online inference offloading problem with inference requests arriving one by one without the knowledge of future arrivals.

5.1 Admission Policy Based on Request Deferral

To reduce energy consumption, it is preferable to admit requests with lower energy consumption, however future demand is not known a priori. Admitting requests with more energy consumption now may lead to the rejection of future requests, as shown in Fig. 3. The proposed algorithm therefore defers the admission of a current request, waiting

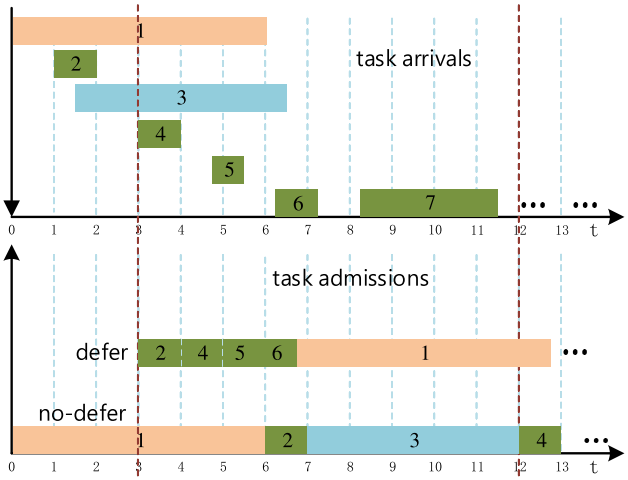


Fig. 3. The motivation of deferring requests. Task 1 has the highest execution latency and the earliest arrival time. If task 1 is admitted on its arrival, the future requests may not be admitted due to the capacity constraint of cloudlets. However, if task 1 is deferred, future tasks 2, 4, 5, and 6 can be executed before the execution of task 1. The deferral of some requests thus increases the system throughput of the system.

for inference requests with a lower energy consumption. This however may lead to a significant violation of the delay requirement of the current inference request.

Considering the afore-mentioned observations, the proposed online algorithm explores the interplay between energy consumption and delay requirements of inference requests. Specifically, we allow the algorithm to defer the assignment of the first t arrived requests. This allows the algorithm to build its request 'expectation'. Then, starting from the $(t+1)$th request, the algorithm selects a request with the least energy consumption for its admission to perform further calculations. Our basic idea is to leverage a reinforcement learning procedure to allow the algorithms dynamically learn whether to defer an inference request. In other words, we aim to design a learning-aided algorithm to dynamically decide whether to admit an inference request or not.

Notice that since we consider online algorithms, we adopt the oblivious adversary model. This adversary knows the algorithm's steps, but does not know the randomized results of the algorithm. This adversary thus randomly generates the arrivals of requests and does not present the requests one by one.

5.2 An Online Optimization Framework for Dynamic Inference Request Admissions

The framework based on request deferral consists of two phases: phase 1 defers the first t requests and builds the expectations of the requests, calculating *pre-offload* decisions for the first t requests by invoking Algorithm 1. The obtained solution is considered as *pre-offloading*, indicating only a decision to offload, without any real offloading action. Let e'_j be the energy consumption due to the implementation of request r_j with $1 \leq j \leq t$. Phase 2 selects the request with minimum energy consumption among all the requests. It then offloads the tasks of the chosen request according to its pre-offloading decisions. However such request admission deferral may cause prohibitive waiting times, which eventually may violate the delay requirements of inference requests. To avoid such cases, in the second phase, if there is a request whose delay requirement will be violated if deferred again, it will be chosen instead of the request with the minimum energy consumption. The details of the proposed online algorithm based on request deferral are shown in Algorithm 3.

Algorithm 3. `Online`

Input: $G = (\mathcal{BS} \cup \mathcal{CL} \cup \mathcal{V}, E)$, a set R of inference requests that arrive into the system one by one without the knowledge of their future arrivals.

Output: An offloading decision for each inference request $r_j \in R$.

$L \leftarrow \emptyset$; /*a set of deferred requests*/
while there is an arrived request r_j **do**
  Find a pre-offloading decision for r_j by invoking Algorithm 1;
  if $|L| < t$ **then**
    Defer the execution of r_j;
  else
    if there is a request $r_{j'} \in L$ whose delay requirement is violated if its is deferred **then**
      Implement $r_{j'}$ according to its pre-offloading decision;
    Find the request in L with the minimum energy consumption and implement it according to its pre-offloading decision;

5.3 An Online Algorithm Based on Reinforcement Learning

So far we proposed an online algorithm with deferred request admissions, by assuming that requests arrive into the system one by one. However, the arrival times of inference requests are not considered in Algorithm 3. In the worst case, the algorithm may wait indefinitely for the arrival of the next inference request, if the arrival time of the next request is far from now. The algorithm needs a smart and adaptive strategy to determine whether it should wait for future requests or not.

To this end, we adopt a Reinforcement Learning (RL) process to make the decision of whether to wait for the next request or select one request from the current arrival list. Specifically, on the implementation of the current inference request r_j, the algorithm observes the system state s_j, and it is asked to choose an action a_j based on its reward. Following the action, the state of the environment transitions its state from s_j to s_{j+1} and the agent receives a reward rwd_j. The state space and the action space of the RL process are defined as follows.

State Space. The state of the system consists of the current arrival list L of requests and their delay requirements. Besides, we also consider the arrival intervals of the arrived requests, where an arrival interval is defined as the time difference between two consecutive requests.

Action Space. The agent of the online algorithm needs to decide whether to wait the arrival of the next request or select one request from the current list L to be executed. Therefore, the action for the agent can be modeled as $\{0, 1\}$, where 0 indicates that the agent wants to wait for the next request while 1 implies that a request from the current list will be selected for admission. If action 0 is selected, the

algorithm will wait until the next request is put into the current list L; otherwise, the algorithm will choose a request from L.

Reward. Recall that the proposed algorithm is to avoid long waiting future requests, we consider that the reward of the agent is defined as the reverse of the average delay experienced by the admitted requests. If the reward of the agent decreases more than a predefined threshold ϑ, this means that the agent may spend more time in waiting for the next request. The agent will thus select a request from the current list, otherwise, the agent will keep waiting the next request.

Detailed steps of the adaptive online algorithm are illustrated in Algorithm 4.

Algorithm 4. `OnlineRL`

Input: $G = (\mathcal{BS} \cup \mathcal{CL} \cup \mathcal{V}, E)$, a set R of inference requests that arrive into the system one by one without the knowledge of their future arrivals, each inference request has a sequence of tasks $M_{j,1}, \ldots, M_{j,k}, \ldots, M_{j,K_j}$ that need to be executed either locally in their mobile device or remotely in a few cloudlets.

Output: An offloading decision for each inference request $r_j \in R$.

1: $L \leftarrow \emptyset$; /*a set of deferred requests*/
2: $WAIT \leftarrow true$;
3: **while** true **do**
4: **if** $WAIT \leftarrow true$ **then**
5: Wait for the next request r_j;
6: **else**
7: Select one request from the current list L;
8: Find a pre-offloading decision for r_j by invoking Algorithm 2;
9: **if** $|L| < t(= c \ln |R|)$ **then**
10: Defer the execution of r_j;
11: **else**
12: **if** there is a request $r_{j'} \in L$ whose delay requirement is violated if its is deferred **then**
13: Implement $r_{j'}$ according to its pre-offloading decision;
14: **else**
15: Find the request in L with the minimum energy consumption and implement it according to its pre-offloading decision;
16: Record the status of the offloaded request and the system status;
17: Calculate the rwd_j as the average delay of offloaded requests;
18: **if** $rwd_{j-1} - rwd_j > \vartheta$ **then**
19: $WAIT = false$;
20: **else**
21: $WAIT = true$;

Theorem 3. *The deferral strategy adopted by Algorithms 3 and 4 deliver a feasible solution to the online inference offloading problem. It has a probability of* $1 - (1 - \zeta \ln(1/\zeta))^{\ln |R|}$ *to identify a request with the minimum energy consumption, where* $\zeta = \frac{c \ln |R|}{|R|}$ *with c being a constant with* $c \geq 1$.

Proof. To show the solution feasibility, we need to show that the computing capacity of each cloudlet and the delay requirement of each admitted request are met. Since Algorithms 3 and 4 invoke Algorithm 2 to process the requests one by one, the computing capacity is met according to Lemma 1. Also, since we check the delay requirement of each request before its deferral, its delay requirement can be met as well.

Recall that the adversary randomly permutes the order of the requests and presents them to the online algorithm one by one. Since there are $|R|$ requests, we thus have a random permutation π of requests. The basic idea of the proposed algorithms is to find a request in π with the minimum energy consumption, and then finding the first $j \geq t+1$ such that the energy consumption of r_j is smaller than the minimum in π.

Now, suppose that r_j is the request with the minimum energy consumption. The proposed algorithms may fail to select r_j, if between the $(t+1)$th request and the $(j-1)$th request there are some requests with less energy consumption than the minimum of the first t requests. When this happens, the minimum of $\pi[1], \ldots, \pi[j-1]$ has to occur in locations between $t+1$ and $j-1$. This means that we do pick the request with the minimum energy consumption if the minimum among the first $j-1$ requests happen to be one of the first t requests.

The probability of picking request r_j with the minimum energy consumption in each trial is

$$\begin{aligned}
&\sum_{j=t+1}^{|R|} Pr[\pi_j \text{ is the minimum and the minimum of } \\
&\qquad \pi[1], \ldots, \pi[j-1] \text{ is in } \pi[1], \ldots, \pi[t]] \\
&= \sum_{j=t+1}^{|R|} \frac{1}{|R|} \cdot \frac{t}{j-1} = \frac{t}{|R|} \Big(\sum_{j=1}^{|R|-1} \frac{1}{j} - \sum_{j=1}^{t} \frac{m}{j}\Big) \\
&\approx \frac{t}{|R|} (\ln |R| - \ln t) = \frac{t}{|R|} \ln \frac{|R|}{t},
\end{aligned} \tag{32}$$

where m is the index of a request in $\pi[1], \ldots, \pi[j-1]$ with the minimum energy consumption. If we perform $\ln |R|$ trials when selecting a request, we can have the probability of selecting a request with the minimum energy consumption promoted to

$$1 - \Big(1 - \frac{t}{|R|} \ln \frac{|R|}{t}\Big)^{\ln |R|}. \tag{33}$$

Assuming that $t = c \ln |R|$, we have the last expression being

$$1 - (1 - \zeta \ln(1/\zeta))^{\ln |R|}, \tag{34}$$

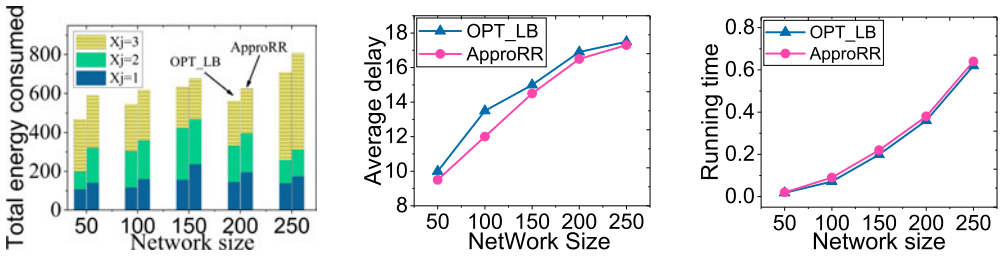
where $\zeta = \frac{c \ln |R|}{|R|}$ and c is a constant with $c \geq 1$. □

6 Experiments

In this section we evaluate the performance of the proposed algorithms by extensive simulations and implementations in a real test-bed.

6.1 Parameter Settings

We consider a multi-cell MEC network by varying from 50 to 250 base stations, where each network topology is generated using GT-ITM [25]. Each cloudlet has a computing power in the range 8,000 to 16,000 MHz [1], [54]. The number of inference tasks that can be executed by each cloudlet



(a) Total energy consumed (W). (b) Average delay (ms). (c) Running time (s).

Fig. 4. The performance of algorithms OPT_LB and ApproRR.

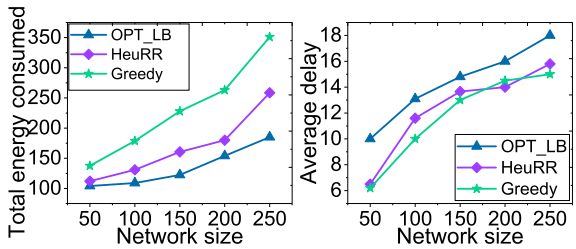
varies from 10 to 50, assuming that each inference task consumes roughly 160 to 1600 MHz computing power. The bandwidth capacity of each base station BS_l is drawn from the range of [10,100] $Mbps$. The maximum number of subtasks of each request is varied from 2 to 5. The input data of each inference request is randomly withdrawn from the range of $[10,\ 100]Mb$ [26]. The average delay requirement by a user of cloudlet is a random value between 10 and 50 milliseconds. The parameters β_i and γ_j on processing unit amount (one KB) of data in cloudlet CL_i and locally are varied from ranges $[0.0001, 0.0005]$ seconds and $[0.005, 0.05]$ seconds, respectively. The parameter α_j is varied from 0.1 to 2, considering that a subtask of a request may output data that is smaller or greater than its input data. The processing and data transmission powers of a mobile device are set to [5, 10] $Watt$ and [10, 50] $Watt$ [33], respectively. The maximum GPU power of each cloudlet is randomly withdrawn from $[60,\ 100]$ $Watt$ and the idle GPU power is 5 $Watt$ [12]. Unless otherwise specified, we will adopt these default settings in our experiments. Each value in the figures is the mean of the results by applying each mentioned algorithm on 15 different topologies of the multi-cell MEC network.

We compare the proposed algorithms ApproRR and HeuRR with the following benchmark algorithms: (1) Greedy: a heuristic that adopts a greedy partition of the DNN of each inference request. Specifically, in Greedy, the number of offloaded subtasks are greedily assigned to the cloudlets that can achieve the minimum data transmission between the UE and the MEC; (2) OPT_LB: since the optimal solution of the proposed solution cannot be obtained in polynomial time, we consider an estimation of the optimal solution by **LP**. Recall that **LP** provides a fractional solution to the problem, its obtained solution thus is a lower bound of the optimal solution. In addition, besides algorithms Greedy, we investigate the performance of algorithms Online and OnlineRL against that of a benchmark that invokes algorithm HeuRR on the arrival of each request. It is referred to as HeuRR_NoDeferral. Similarly, algorithm Greedy is also named as Greedy_NoDeferral for the online problem.

6.2 Performance Evaluation of Algorithms ApproRR and HeuRR

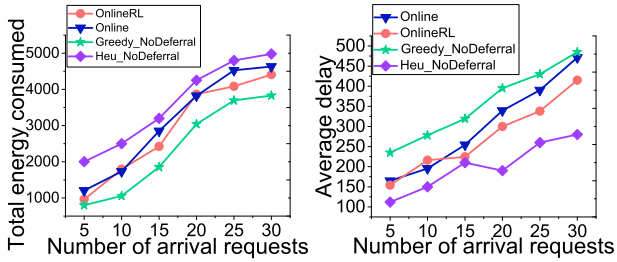
We first evaluate the performance of algorithm ApproRR against OPT_LB and Greedy in terms of the total consumed energy, the average delay of a request, and the running time, by varying the number of tasks executing locally, i.e., x_j, from 1 to 3 and fixing the total subtasks of each request to 3, in networks with sizes varied from 50 to 250. Fig. 4a shows the total amount of energy consumed by all base stations when different numbers of subtasks of an inference request are offloaded to the MEC network for executing. We can see that for each value of x_j our algorithm ApproRR is close to the optimal solution OPT_LB. Namely, when the network size is 250 and $x_j = 1$, the approximation ratio of algorithm ApproRR is 1.25. From Fig. 4a, we can also see that the total consumed energy increases with the growth of x_j. The reason is that a higher value for x_j means more tasks are executed locally in UEs, thereby increasing the energy consumption of UEs. Fig. 4b illustrates the average delay experienced by an inference request. We can see that the delay of the proposed algorithm is much lower than that of algorithm OPT_LB. The rationale behind is that OPT_LB seeks to minimize the energy consumption as long as the delay requirement is met. It thus pushes the limit of the delay constraint to gain more opportunities of minimizing the energy consumption. Instead, algorithm ApproRR is based on the fractional solution of LP via randomized rounding, by considering the value of each decision variable as a probability. This allows the algorithm ApproRR deviate from the action of maximally trading delay for energy. Fig. 4c illustrates that the running time of algorithm ApproRR is roughly equivalent to that of solving linear programs without integer variables. It must be mentioned that obtaining the optimal solution to the **ILP** takes prohibitive long time for large problem sizes.

We then study the performance of algorithm HeuRR against OPT_LB and Greedy in terms of the total consumed energy and the average delay of a request, by fixing the number of subtasks of each request at 3 and varying the network size from 50 to 250. Fig. 5a shows the total consumed energy obtained by OPT_LB is the lowest, because it is a lower bound of the optimal solution OPT. Algorithm HeuRR has a lower total energy consumption than Greedy, since Greedy only



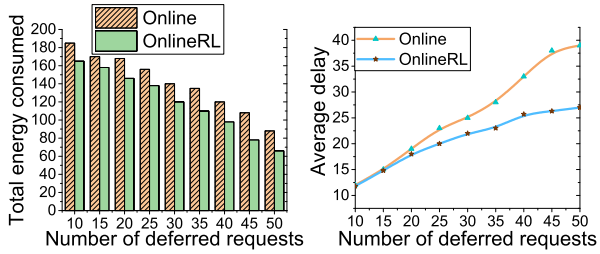
(a) Total energy consumed (W). (b) Average delay(ms).

Fig. 5. The performance of algorithms OPT_LB, HeuRR, and Greedy.



(a) Total energy consumed (W). (b) Average delay(ms).

Fig. 6. The performance of algorithms Online, OnlineRL and Heu_NoDeferral, and Greedy_NoDeferral.



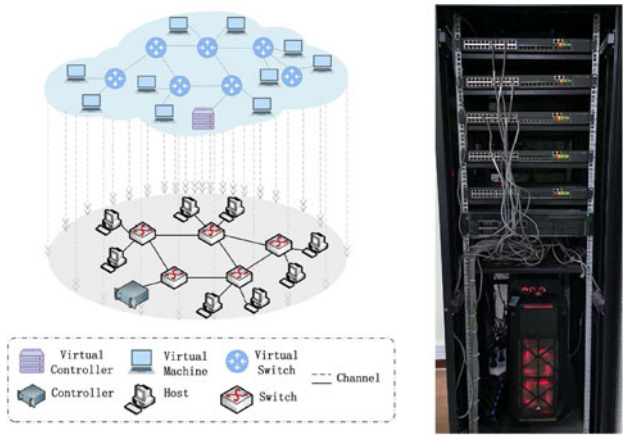
(a) Total energy consumed (W). (b) Average delay(ms).

Fig. 8. The performance of algorithms Online and OnlineRL.

selects the minimum energy consumption due to wireless transmission. Fig. 5b depicts the average delay obtained by algorithms HeuRR, Greedy and OPT_LB, from which we can see that OPT_LB has the longest delay among the three algorithms. On the contrary, algorithm HeuRR has the shortest delay. The reason is that algorithm OPT_LB delivers a fractional solution to the problem, which is a lower bound of OPT. Algorithm HeuRR however divides each fractional value of the fractional solution by 2, which reduces the probability of selection of cloudlets with large delays.

6.3 Performance Evaluation of Algorithms Online and OnlineRL

We then investigate the performance of algorithms Online, OnlineRL, Heu_NoDeferral, and Greedy_NoDeferral by varying the network size from 50 to 250. The arrival interval of two consecutive requests follows the Zipf distribution [39], which means that around 80 percent of arrival intervals are small while 20 percent are large. We can see from Fig. 6a that algorithms Online and OnlineRL have slightly a lower energy consumed than that of algorithm Heu_NoDeferral. The reason is that Online and OnlineRL defer some request admissions when needed; this increases the opportunity of selecting requests with lower energy consumptions. The total energy consumption increases when the number of arrived requests increases from 5 to 25, and it then decreases afterwards when the number of arrived requests is higher than 25. The reason is that more requests normally incur higher energy consumption; however, the MEC network becomes saturated when the number of requests keeps increasing. On the other hand, we can see from Fig. 6b that Online and OnlineRL have slightly higher delays than other algorithms, as they defers requests for larger selection space for lower energy.



(a) The underlay and overlay of the test-bed (b) The physical deployment

Fig. 7. A test-bed for the multi-cell MEC network with an underlay and a large-scale overlay.

6.4 Performance Evaluation of Algorithms Online and OnlineRL in a Real Test-Bed

To testify the applicability of the proposed algorithms in real environments, we finally study the performance of algorithms Online and OnlineRL in a test bed. Specifically, we investigate the impact of the number t of maximally deferred requests on the performance of the algorithms. In the rest, we first describe the settings of our test-bed and then elaborate on the results obtained in the test-bed.

Test-Bed Settings. Recall that we consider an 5G-enabled MEC with a set of cloudlets and switches that interconnect the cloudlets in the mobile access network. Our test-bed thus consists of five hardware switches and several physical servers (i.e., cloudlets). In real environments, there can be many switches in the mobile access network. To approach such settings of large-scale access networks, we adopt a two-layer settings of the test-bed. Specifically, the test-bed has both an underlay with hardware switches and an overlay with virtual switches, as shown in Fig. 7. In particular, each switch of the underlay is a H3C S5560-30S-EI switch, and each of the five servers has an i7-8700 CPU and 16G RAM. We use VXLAN to build the overlay network with a number of Open vSwitch (OVS) [37] nodes and VMs, where Netconf and SNMP protocols are used to manage the switches and the links that interconnect them. The topology of the overlay network is built following the real topology AS1755. Its OVS nodes and VMs are controlled by a Ryu [41] controller. The proposed algorithms are implemented as Ryu applications. All the rest settings are the same as the simulations in the previous subsection.

Results. We evaluate the performance of algorithms Online and OnlineRL by varying t from 10 to 50 with an increase of 5. Figs. 8a and 8b shows the results in terms of the total energy consumed and average delay, from which we can see that the total energy consumed is decreasing with the growth of t and the average delay on the other hand is increasing. The reason is that there will be more requests that are being deferred with the increase of t, leading to the rejection of some requests due to delay violations. This means less requests are admitted by the algorithms, thereby reducing the energy consumed. Since more requests are deferred, the average delay of admitted requests thus is larger for large values of t. In addition, the

average delay of OnlineRL is less than that of Online, due to the OnlineRL can observe the system state for long-term reward and choose whether to wait for the next request, thus, the average delay is reducing.

7 Conclusion

In this paper, we studied the inference offloading problem in a 5G multi-cell MEC network with the aim to minimize the total energy consumption of mobile devices, 5G base stations and cloudlets. We first proposed exact and approximate solutions to the inference offloading problem for the admission of a single inference request, by utilizing a randomized rounding technique. We then devised a learning-based online algorithm for the online inference offloading problem. An offloading deferral approach is also proposed, which aims to minimize energy consumption across a number of submitted inference requests. Experimentally, we finally investigated the performance (and benefits) of the proposed algorithms by extensive simulations. Results show that the proposed algorithms outperform other approaches in literature.

The future works of this study include: (1) in real scenarios, the processing and transmission delays may be uncertain, considering the time-varying network statuses of MEC networks. We thus consider the inference offloading with uncertain information of the MEC network as our first future work, and (2) the age-aware inference offloading for DNN-driven AI applications. Some inference requests may have time-sensitive data that needs to be processed timely. The proposed algorithm can be extended to consider such a scenario by responding to such requests immediately after their arrivals while deferring requests with no time-sensitive data. However, a finer-grained scheduling of such time-sensitive inference requests is desirable, and we consider the design of age-aware scheduling methods in MEC networks as our future work.

Acknowledgments

The authors would like to thank the three anonymous referees and the associate editor for their expertise comments and constructive suggestions, which have helped them improve the quality and presentation of the article greatly. The work by Zichuan Xu and Qiufen Xia was supported in part by the National Natural Science Foundation of China (Grant No. 61802048 and 61802047), the fundamental research funds for the central universities in China (Grant No. DUT19RC(4)035), DUT-RU Co-Research Center of Advanced ICT for Active Life, and the "Xinghai Scholar Program" in Dalian University of Technology, China. The work by Weifa Liang was supported by the Australian Research Council Discovery Project (Grant No. DP200101985). The work of Pan Zhou was supported by the National Natural Science Foundation (Grant No. 61972448).

References

[1] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 668–682, Mar. 2019.

[2] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy cost models of smartphones for task offloading to the cloud," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 3, pp. 384–398, Sep. 2015.

[3] K. Apicharttrisorn, X. Ran, J. Chen, S. V. Krishnamurthy, and A. K. Roy-Chowdhury, "Frugal following: Power thrifty object detection and tracking for mobile augmented reality," in *Proc. 17th Conf. Embedded Netw. Sensor Syst.*, 2019, pp. 96–109.

[4] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 337–368, First Quarter 2014.

[5] R. Chattopadhyay and C. Tham, "Fully and partially distributed incentive mechanism for a mobile edge computing network," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2020.3003079

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

[7] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Hu, "iRAF: A deep reinforcement learning approach for collaborative mobile edge computing IoT networks," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7011–7024, Aug. 2019.

[8] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

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

[10] M. Deng, H. Tian, and X. Lyu, "Adaptive sequential offloading game for multi-cell mobile edge computing," in *Proc. 23rd Int. Conf. Telecommun.*, 2016, pp. 1–5.

[11] C. Gao, A. Rios-Navarro, X. Chen, T. Delbruck and S. Liu, "EdgeDRNN: Enabling low-latency recurrent neural network edge inference," in *Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst.*, 2020, pp. 41–45.

[12] S. Hong and H. Kim, "An integrated GPU power and performance model," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 280–289.

[13] M. D. Hossain *et al.*, "Collaborative task offloading for overloaded mobile edge computing in small-cell networks," in *Proc. Int. Conf. Inf. Netw.*, 2020, pp. 717–722.

[14] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing–A key technology towards 5G," *ETSI, Sophia Antipolis, France*, ETSI White Paper 11, 2015, pp. 1–16.

[15] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive DNN surgery for inference acceleration on the edge," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1423–1431.

[16] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Learning driven computation offloading for asymmetrically informed edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1802–1815, Aug. 2019.

[17] H. Huang and S. Guo, "Proactive failure recovery for NFV in distributed edge computing," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 131–137, May 2019.

[18] H. Huang, S. Guo, W. Liang, K. Wang, and Y. Okabe, "Coflow-like online data acquisition from low-earth-orbit datacenters," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2019.2936202

[19] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1459–1467.

[20] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[21] H. Jeong, H. Lee, C. Shin, and S. Moon, "IONN: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proc. ACM Symp. Cloud Comput.*, 2018, pp. 401–411.

[22] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2018, pp. 253–266.

[23] S. Jiang *et al.*, "Accelerating mobile applications at the network edge with software-programmable FPGAs," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 55–62.

[24] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. 22nd Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2017, pp. 615–629.

[25] GT-ITM, 2000. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[26] J. Li, H. Gao, and T. Lv, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2018, pp. 1–6.

[27] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.
[28] H. Li, K. Ota, and M. Dong, "Deep reinforcement scheduling for mobile crowdsensing in fog computing," *ACM Trans. Internet Technol.*, vol. 19, 2019, Art. no. 21.
[29] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, 2018, pp. 31–36.
[30] J. Leng, T. Hetherington, and A. ElTantawy, "GPUWattch: Enabling energy optimizations in GPGPUs," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, 2013, pp. 487–498.
[31] C. Luo and R. Suda, "A performance and energy consumption analytical model for GPU," in *Proc. IEEE 9th Int. Conf. Dependable Auton. Secure Comput.*, 2011, pp. 658–665.
[32] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
[33] C. X. Mavromoustakis, G. Kormentzas, and G. Mastorakis, "Joint energy and delay-aware scheme for 5G mobile cognitive radio networks," in *Proc. IEEE Global Commun. Conf.*, 2014, pp. 2624–2630.
[34] M. Md, A. Q. Li, and I. Rekleitis, "Deep neural networks: A comparison on different computing platforms," in *Proc. 15th Conf. Comput. Robot Vis.*, 2018, pp 383–389.
[35] S. Mittal, "A survey of FPGA-based accelerators for convolutional neural networks," *Neural Comput. Appl.*, vol. 32, pp. 1109–1139, 2020.
[36] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2005.
[37] Open vSwitch, 2016. [Online]. Available: https://www.openvswitch.org
[38] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 10–18.
[39] D. M. W. Powers, "Applications and explanations of Zipf's Law," in *Proc. Joint Conf. New Methods Lang. Process. Comput. Natural Lang. Learn.*, 1998, pp. 151–160.
[40] P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.
[41] Ryu SDN controller, 2017. [Online]. Available: https://ryu-sdn.org/
[42] C. Shi, L. Chen, C. Shen, L. Song, and J. Xu, "Privacy-aware edge computing based on adaptive DNN partitioning," in *Proc. IEEE Global Commun. Conf.*, 2019, pp. 1–6.
[43] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit.*, 2016, pp. 2464–2469.
[44] 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.*, 2017, pp. 328–339.
[45] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching, and computing in wireless systems: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 7–38, First Quarter 2018.
[46] C. Wang *et al.*, "Joint server assignment and resource management for edge-based MAR system," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2378–2391, Oct. 2020.
[47] Z. Wu and R. J. Radke, "Real-time airport security checkpoint surveillance using a camera network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2011, pp. 25–32.
[48] C. Xia, J. Zhao, H. Cui, and X. Feng, "Characterizing DNN models for edge-cloud computing," in *Proc. IEEE Int. Symp. Workload Characterization*, 2018, pp. 82–83.
[49] Q. Xia, W. Liang, Z. Xu, and B. Zhou, "Online algorithms for location-aware task offloading in two-tiered mobile cloud environments," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput.*, 2014, pp. 109–116.
[50] Q. Xia, Z. Lou, W. Xu, and Z. Xu, "Near-optimal and learning-driven task offloading in a 5G multi-cell mobile edge cloud," *Comput. Netw.*, vol. 176, 2020, Art. no. 107276.
[51] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.
[52] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao, "Task offloading with network function services in a mobile edge-cloud network," *IEEE Trans. Mobile Comput.*, vol. 18, no. 11, pp. 2672–2685, Nov. 2019.
[53] Z. Xu *et al.*, "Learning for exception: Dynamic service caching in 5G-enabled MECs with bursty user demands," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2020.
[54] Z. Xu, L. Zhou, S. Chi-Kin Chau, W. Liang, Q. Xia and P. Zhou, "Collaborate or separate? Distributed service caching in mobile edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2066–2075.
[55] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5G," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6398–6409, Jul. 2018.
[56] Q. Yang, X. Luo, P. Li, T. Miyazaki, and X. Wang, "Computation offloading for fast CNN inference in edge computing," in *Proc. Conf. Res. Adaptive Convergent Syst.*, 2019, pp. 101–106.
[57] Q. Yang, Y. Liu, T. Chen and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, 2019, Art. no. 12.
[58] Z. Zhou, B. Wang, M. Dong, and K. Ota, "Secure and efficient vehicle-to-grid energy trading in cyber physical systems: Integration of blockchain and edge computing," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 50, no. 1, pp. 43–57, Jan. 2020.
[59] S. Zhou, H. Huang, W. Chen, Z. Zheng and S. Guo, "PIRATE: A blockchain-based secure framework of distributed machine learning in 5G networks," *IEEE Netw.*, to be published, doi: 10.1109/MNET.001.1900658

Zichuan Xu (Member, IEEE) received the BSc and ME degrees from the Dalian University of Technology, China, in 2008 and 2011, respectively, all in computer science, and the PhD degree from The Australian National University, Australia, in 2016. From 2016 to 2017, he was a research associate at the Department of Electronic and Electrical Engineering, University College London, United Kingdom. He is currently an associate professor and a PhD advisor in the School of Software, Dalian University of Technology, China. He is also a 'Xinghai Scholar' at the Dalian University of Technology, China. His research interests include cloud computing, mobile edge computing, deep learning, network function virtualization, software-defined networking, routing protocol design for wireless networks, algorithmic game theory, and optimization problems.

Liqian Zhao received the BS degree from the Henan University of Science and Technology, Luoyang, China, in 2019. She is currently working toward the ME degree with the School of Software, Dalian University of Technology, China. Her main research interests are mobile edge computing, network storage, task offloading, deep learning and optimization problems.

Weifa Liang (Senior Member, IEEE) received the BSc degree from Wuhan University, China, in 1984, the ME degree from the University of Science and Technology of China, China, in 1989, all in computer science, and the PhD degree from The Australian National University, Australia, in 1998. He is currently a full professor with the Research School of Computer Science, The Australian National University, Australia. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, mobile edge computing and cloud computing, network function virtualization, software-defined networking, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory.

Omer F. Rana (Senior Member, IEEE) received the BS degree in information systems engineering from Imperial College of Science, Technology and Medicine, London, United Kingdom, the MS degree in microelectronics systems design from the University of Southampton, Southampton, United Kingdom, and the PhD degree in neural computing and parallel architectures from the Imperial College of Science, Technology and Medicine, United Kingdom. He is a professor of performance engineering with Cardiff University, Cardiff, United Kingdom. His current research interests include problem solving environments for computational science and commercial computing, data analysis and management for large-scale computing, and scalability in high performance agent systems.

Pan Zhou (Senior Member, IEEE) received the BS degree in the Advanced Class of the Huazhong University of Science and Technology, China, the MS degree from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2008, respectively, and the PhD degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology (Georgia Tech), Atlanta, Georgia, in 2011. He is currently an associate professor and PhD advisor with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, China. He held honorary degree in his bachelor and Merit Research Award of HUST in his master study. He was a senior technical member at Oracle Inc., USA, during 2011 to 2013, and worked on Hadoop and distributed storage system for big data analytics at Oracle Cloud Platform. He received the "Rising Star in Science and Technology of HUST, in 2017. He is currently an associate editor of the *IEEE Transactions on Network Science and Engineering*. His current research interest includes: security and privacy, big data analytics and machine learning, and information networks.

Qiufen Xia (Member, IEEE) received the BSc and ME degrees from the Dalian University of Technology, China, in 2009 and 2012, respectively, all in computer science, and the PhD degree from The Australian National University, Australia, in 2017. She is currently a lecturer at the Dalian University of Technology, China. Her research interests include mobile cloud computing, query evaluation, big data analytics, big data management in distributed clouds, and cloud computing.

Wenzheng Xu (Member, IEEE) received the BSc, ME, and PhD degrees in computer science from Sun Yat-Sen University, Guangzhou, China, in 2008, 2010, and 2015, respectively. He currently is an associate professor at Sichuan University, China. Also, he was a visitor at both the Australian National University, Australia and the Chinese University of Hong Kong, Hong Kong. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory.

Guowei Wu received the PhD degree from Harbin Engineering University, China, in 2003. He is currently a professor with the School of Software, Dalian University of Technology(DUT), China. His research interests include embedded real-time systems, cyber-physical systems, and smart edge computing. He has published more than 100 papers in journals and conferences.